

The Modern Software Delivery Platform[®]

Developer Experience Unleashed

How Harness supercharges your developer experience

Build fast and ship frequently

Next Generation CI/CD

- 4x Faster Builds
- One-click deployments
- Fully automated pipelines
- AI/ML driven deployment verification
- Continuous Delivery & GitOps
- Continuous Integration
- Code Repository
- Infrastructure as Code Management

Integrate security at every step

Shift Left Security

- Aggregate security scans in CI/CD pipelines
- Vulnerability de-duplication & prioritization
- AI/ML driven vulnerability remediation
- SBOM orchestration & SLSA attestation
- Security Testing
Orchestration
- Software Supply
Chain Assurance

Improve resilience and quality

Continuous Resilience

- Automated SLO management
- Chaos engineering for resilience testing
- Quality gates in pipelines
- AI/ML driven change impact analysis
- Chaos
Engineering
- Service Reliability
Management
- Continuous

Release features confidently

Feature Management

Manage flags with GitOps experience

Build Feature Flag pipelines

Manage Flag Tech-Debt

Feature Flags

Accelerate developer onboarding

Platform Engineering

IDP for service catalog & scorecards

Rapid onboarding with golden paths

Built on Open Source - Backstage

Internal
Developer Portal

Drive cost & process efficiency

Cost & Process Optimization

Cloud cost insights

Actionable intelligence across the SDLC

Automatically stop idle resources

OOTB DORA, planning, and execution metrics

Cloud Cost
Management

Software Engineering
Insights

Harness AIDAâ€¢

Leverage AI across the software delivery lifecycle: code, build and test, secure, deploy, manage costs, and ensureÂ resilience.

Engineering excellence platform for the enterprise

Hundreds of DevOps and engineering teams are powered by Harness to become elite performers in velocity, quality, efficiency, and governance.

Citi improves software delivery performance with Harness

Harness CD lets us release each change within minutes of a pull request being merged. Running all the necessary tests and scans during the pipeline gives us the confidence to do this.

Stefanos Piperoglou, Technical Program Manager, Citi

United Airlines Accelerates Deployments by 75% With Harness

By choosing Harness for CI and CD, we were able to give the governance policies to the developers and create the guardrails we needed. Harness gives us a platform rather than just a DevOps tool.

Ratna Devarapalli, Director of IT - Architecture, Platform Engineering & DevOps

Ancestry adds consistency and governance to cut downtime and systems onboarding effort

Harness now empowers Ancestry to implement new features once and then automatically extend those across every pipeline, representing an 80-to-1 reduction in developer effort.

Ken Angell, Principal Architect, Ancestry

Try Harness for free today

Source URL: <https://www.harness.io/products/software-engineering-insights>

Data-driven Engineering

Discover SDLC bottlenecks, assess team productivity, and improve developer experience guided by data and insights.

Assess and boost developer productivity

Proven metrics to track and improve developer productivity

Leverage the Harness Trellis Framework to quickly understand facets of productivity for developers and teams. Trellis' algorithm analyzes over 20 factors gathered from a range of SDLC tools to generate a comprehensive report pinpointing areas where developer productivity can be enhanced.

Discover and fix bottlenecks in your SDLC

Leverage DORA and other frameworks to get actionable insights

Get actionable insights into contributing factors to identify bottlenecks. Analyze across various pivot points to be able to pinpoint bottlenecks quickly. Understand systemic and tactical problems across the entire delivery lifecycle. Realize value quickly using the A widget library included in Software Engineering Insights.

Improve planning and predictability

Streamline execution and improve process adherence

Scorecards and dashboards highlight factors that impact sprint predictability, including anomalies and unplanned work such as missing story points or anti-patterns (such as large stories) that help you assess if new features and changes will be delivered to customers on time. SEI also improves product management and engineering collaboration, enabling best practices for product management hygiene and reducing scope creep.

Articulate engineering investments

Ensure resource allocation supports business demands

Validate that engineering resources are allocated according to business needs. Get a comprehensive breakdown of how time and resources have been invested across different categories (features, bugs, technical debt, etc.), evaluate how that aligns to business plans, and identify wasteful or unplanned engineering work.

COMING SOON

Drive a continuous improvement program

Drive improvements and excellence in a structured way

Driving change and transforming an organization is hard. Data, Insights and KPIs are just one part of the story, but how do engineering leaders change the culture of the team to be more data oriented. Software Engineering Insights can help formulate an improvement plan using Goals and Working Agreements.

Get insights from 40+ tools

Do More. Ship More. Celebrate More

Trusted by DevOps and Developers

Hundreds of DevOps and engineering teams are powered by Harness to become elite performers in velocity, quality, efficiency, and governance.

Broadcom

We've seen incredible improvement in engineering productivity. The technology automates our KPIs and removes bottlenecks, reducing our developers' task burden by 30% in certain areas, enabling them to focus on more rewarding work.

Harness Software Engineering Insights

Product Documentation

Learn how to connect SEI with your existing tech stack and get insights. How to remove bottlenecks and improve planning and sprint hygiene

Product Updates

See our latest feature releases, product improvements and announcements

Blogs

Read on for educational material, technical deep dives, Harness tutorials, and everything in between

Case Studies

Be inspired by success stories from industry leaders

Source URL: <https://www.harness.io/support>

Welcome to Harness Support

Find technical support information in our knowledge base across product docs, blogs, community sites, Harness University, API references, and your support tickets (sign-in required).

Our Mission

We are here to ensure satisfaction and deliver happiness for our end users by enabling and helping them leverage the Harness platform for simplifying their entire Software Delivery Process.

Where are we located?

We use a follow-the-sun model to provide support to our customers around the globe. Our engineers are located in the US, Brazil, Germany and India.

Who can get support

We provide support to all paying customers and prospects during an active Proof of Value (POV). The guaranteed response time is based on whether an advanced support plan was purchased.

How is the support eligibility determined?

The email domain of the individual who creates the support request must match the email domain of an organization with a valid Harness subscription, or active POV.

Trial accounts are not eligible for support without an active Proof of Value trial established with a Harness sales executive.

Can free accounts access Harness Support?

Accounts with free subscriptions are eligible for the community support tier. For more details, see Harness Support definitions.

â

Support escalation

For any support ticket escalation if issue remains outstanding even after escalating in the ticket itself, please reach out to CX Leadership by mailing us at cx-escalation@harness.io.

Service Level

Service Level Harness Support is available 9 to 5 pacific standard time, during Harness workdays. 24x7x365 support is available as part of the premier support package. For details, see support tiers and definitions.

How to contact Harness support

The best way to create a support ticket is from within the Harness platform by clicking the

- A. âHelpâ button
- B. Followed by âSubmit a ticketâ.

Other options are:

1. By sending email to support@harness.io. This is the best option if you have a login issue, or if you are using Harness products on-prem.
2. From the Harness Support portal (<https://support.harness.io/hc/en-us/request>). Notice that a Harness login is required.
3. From your Harness customer slack channel (created by your CSM). Just add the Ticket emoji Â Ä Â Ä Â Ä Â Ä Â Ä Â Ä

Phone and video support

Harness Support does not include in-bound phone or video support. For customers with a premier support plan, the Harness Support engineer, at their discretion, may offer a call to provide faster assistance and to collect information. While the final decision is in the support engineer's hands, customers with Premier support should ask for a call if they feel that it is needed.

If a call is needed, we will send you a link to a scheduling page to pick a time that is best for you to meet. Let us know (via the support ticket) if the options are not compatible with your schedule or time zone, and we'll find another engineer who is compatible with your needs.

Video support options

We support meetings via Zoom or Microsoft Teams. If these are not supported by your organization, it is the customer's responsibility to provide the meeting infrastructure.

Community Support

Harness users without an active subscription or POV can get support by reaching out to the Harness community via Slack. Join Harness Community on Slack

Source URL: <https://www.harness.io/products/infrastructure-as-code-management>

Manage your Infrastructure as Code. End-to-end.

Covering all your Infrastructure as Code Management needs.

Managed entirely on the HarnessÂ Platform

One platform for your developers and cloud engineers to collaborate and manage Infrastructure as Code in a reliable, repeatable way.

Self-service IaC

Better collaboration - shared resources can be managed and used by the same team, avoiding conflict and undetected changes while offering a better review process.

Enable faster adoption of Infrastructure as Code practices.

Reduce time to market - help teams accelerate infrastructure resource changes to decrease deployment cycle times.

Decrease risk

Reduce the occurrence of errors caused by manual processes.

Shift left error detection - help developers and platform engineers find errors earlier in the design.

Complete control through governance. Avoid security breaches and keep all environments safe.

Pipeline for Infrastructure Changes

CI/CD for Infrastructure

âHarness IaCM offers a cutting-edge, sophisticated CI/CDÂ solution tailored to infrastructure changes. Embedding infrastructure automation into the SDLC reduces manual steps and delays

Integrated Pipeline

âSeamlessly incorporate into our comprehensive pipeline platform, enabling streamlined automation and orchestration of infrastructure provisioning and updates.

â

Scale with Templates

Establish reusable automation templates so that your best practices are in place everywhere.

Control costs

Resource visibility - provide complete insight into resource ownership, management, updates, and cost changes.

â

Data-Driven insights - Preview and act on resource changes before they are applied to proactively make cost decisions.

â

Out-of-the-box policies to prevent unexpected cost surges and stay within budget.

Learn more about

Harness Infrastructure as Code Management

Product Documentation

Learn how to use IaCM to manage Infrastructure as code in a self-service approach and design pipelines for complete CI/CD infrastructure automation.

Product Updates

See our latest feature releases, product improvements and announcements

Blogs

Read on for educational material, technical deep dives, Harness tutorials, and everything in between

Case Studies

Be inspired by success stories from industry leaders

Source URL: <https://www.harness.io/products/continuous-delivery>

Worldâs Most âAdvanced CD Platform

Take your applications to production with noÂ scripts.

End-to-end Software Delivery

which tells about all three verticals: Reporting, Optimization, and Governance

Watch full video

Deliver software the HarnessÂ way!

Cloud native deployment without scriptingÂ

Comprehensive GitOps support with native Argo and Flux

Automated verification of failed deployments

Environment aware RBAC, policy as code and full audit trail

Any app anywhere, noÂ scripting

Target any cloud

Deploy to any services of the leading public cloud providers like AWS, GCP, Azure and more

Deployment strategy

Use Canary, Blue Green or Rolling Deployment. No scripting required.

Infrastructure provisioning

Create short lived environments to save cost using a wide range of Infra Provisioners including Terraform, Terragrunt, AWS CDK

GitOps your way

Integrate pipelines with new and existing Argo CD and Flux managed workloads

Centralized control plane for visibility across clusters

Harness Pipelines allow you to quickly integrate build/test/verify with GitOps

Fully automate the path to production: Easily create pull requests, sync application, update application config, and trigger rollbacks from the pipeline

AI-assisted deployment verification

Monitor deployment health

Monitor the health of a deployment by validating metrics and logs from one or many sources

Rollback automation

Automated rollback of problematic releases tied to deployment strategy

Smart notification

Optionally bring in human intervention with smart notifications

Build powerful pipelines

Visual and code editor

Harness provides elegant drag-and-drop and developer friendly as-code experiences that you can toggle between at will. Once you have a great pipeline, save it as a template to allow versioned, parameterized reuse. You donât have to repeat yourself.

Advanced controls

Step into a canary deployment with intelligent rollback. Synchronize key events across services using Barriers to ensure the database is up before service is rolled out. Trigger pipelines on various events. Leverage looping or matrix actions to hit every combination.

Workflow integration

Harness integrates with ticketing systems to automatically raise tickets for approval and proceed when theyâre approved. Customize what happens when an approval is waiting too long or approvals for similar deployments stack up.

Guardrail your deployments

Flexible policies using Open Policy Agent

Give developers permission to edit pipelines while central teams dictate rules like, "All production deployments must have an approval step." With OPA, teams can be empowered while scalable guardrails ensure compliance.

Deployment freeze

Want to prevent deployments during sensitive times? Deployment freezes allow you to block deployments of specified services and environments for a specific window of time.

Enterprise grade role-based access control

Fine grained Role-Based Access Control defines who can access your resources and what actions they can perform.

Audit trails

Harness provides clear and easily accessible audit trails so you know who made a change. With better visibility, you can meet your compliance requirements.

Template library for standardization

Provide users with a process template to follow your enterprise standards and reduce configuration effort.

Flexible policies using Open Policy Agent

Give developers permission to edit pipelines while central teams dictate rules like, "All production deployments must have an approval step." With OPA, teams can be empowered while scalable guardrails ensure compliance.

Deployment freeze

Want to prevent deployments during sensitive times? Deployment freezes allow you to block deployments of specified services and environments for a specific window of time.

Enterprise grade role-based access control

Fine grained Role-Based Access Control defines who can access your resources and what actions they can perform.

Audit trails

Harness provides clear and easily accessible audit trails so you know who made a change. With better visibility, you can meet your compliance requirements.

Template library for standardization

Provide users with a process template to follow your enterprise standards and reduce configuration effort.

Visualize your DevOps data

Harness dashboard provide visibility into your continuous delivery performance. Out of the box overview dashboards provide DORA metrics known to predict high performance in technology teams. Service-level dashboards detail what's where, and bring recent deployments to your fingertips. For those that want to customize, our Looker powered custom dashboards make it easy to craft custom dashboards and automate sending them to interested parties on a schedule or a conditional alert.

More integrations, less scripting

Trusted by DevOps and Developers

Hundreds of DevOps and engineering teams are powered by Harness to become elite performers in velocity, quality, efficiency, and governance.

United

Harness gave us a lot of best practices right out of the box with reusable templates so developers don't need to perform any guesswork to determine how the best pipeline should be built.

Skillsoft

We barely had to train anyone on Harness. Thereâs a big green flag for successful deployments, and if something goes wrong, they can just read the Harness log.

Ulta Beauty

Time-to-market for our eCommerce platform was a huge benefit of Harness. We saved months of time. Months. Harness really came through in a big way for that project.

UWM

In a past position I tried to set up a simple-to-use integration in Jenkins and it took months. We needed a tool that would take minutes.

Ride the wave of Modern Software Delivery

Have a question? We are here to help!

Harness Continuous Delivery and GitOps

Product Documentation

Learn how to automate delivery with Harness Continuous Delivery and GitOps.

Product Updates

See our latest feature releases, product improvements and announcements

Tutorials

Go hands-on to learn key Harness concepts with step-by-step guides.

Case Studies

Be inspired by success stories from industry leaders.

Source URL: <https://www.harness.io/products/continuous-error-tracking>

Developer Observability for Modern Applications

Find and fix issues in minutes (instead of weeks) with full code level visibility and deep context that streamlines remediation efforts across teams â all while keeping your applications up andÂ running.

Complete visibility into errors across Development Pipelines

Release with confidence by finding and fixing errors faster

Minimize Escaped DefectsÂ

Proactively identify new and critical exceptions, and prevent them from negatively impacting customers.

Automate TroubleshootingÂ

Reduce developer toil by eliminating manual troubleshooting of exceptions in development, testing, and production environments.

Trusted by DevOps and Developers

Hundreds of DevOps and engineering teams are powered by Harness to become elite performers in velocity, quality, efficiency, and governance.

Games 24 Seven

We deploy new features and changes frequently, so it's crucial for us to have a tool that provides real-time feedback on application performance. Harness CET has been instrumental in helping us maintain high-quality standards and has greatly simplified our debugging process. We can now confidently deploy code changes, knowing that we have the ability to detect and address issues early on.

Get Started with Continuous Error Tracking

Deliver more reliable software

Source URL: <https://www.harness.io/products/chaos-engineering>

Achieve Continuous Resilience

Discover how your applications stand up to real-world failure scenarios. Gain insights to construct a resilient system that minimizes downtime and saves on costs.

Discover how your applications stand up to real-world failure scenarios.

From Chaos to Resilience within 60 secs

Watch full video

What is Chaos Engineering

Chaos Engineering enables teams to identify weaknesses & potential outages in infrastructures by inducing chaos tests in a controlled way.

Increase System Resilience

Evolve system resilience to make your business predictable

Optimise Cost

Prioritize innovation over incident response, enabling feature creation.

Better Customer Experience

Proactively build resilience to ensure your users do not experience downtime

Faster incident report time

Embed reliability in delivery, promoting successful tech adoption

Why Harness Chaos Engineering?

Free developers from time-consuming, unnecessary processes that slow your work, so you and your team can focus on creating.

Industry's widest coverage on Chaos Faults

Extensive Coverage

Our comprehensive chaos engineering platform offers the most extensive coverage of potential faults in your systems, enabling you to identify and rectify weak points, leading to a better and faster user experience.

Strengthen Systems

With our platform, you gain a deeper understanding of your system's vulnerabilities, allowing you to proactively address potential issues before they impact your users, resulting in improved overall system performance.

Fully automated journey to resilience

Effortless Automation

Our platform provides a fully automated journey to resilience, where you can seamlessly implement chaos engineering practices into

your workflow without the need for extensive manual intervention.

Maximized Efficiency

By automating the chaos engineering process, you save valuable time and resources, allowing your team to focus on developing innovative features and products while ensuring your systems are robust and reliable.

Seamless Chaos Experimentation in Pipelines

Full SDLC Coverage

Integrate chaos experiments effortlessly into your development and deployment pipelines, ensuring that potential issues are identified and addressed at every stage of your software delivery process.

Out of the Box Integrations

Our platform seamlessly integrates with your existing CI/CD pipelines, enabling you to conduct chaos experiments in a controlled manner, guaranteeing that your applications can withstand real-world challenges.

ChaosGuard for easy scaling

Policy Driven

Our platform offers robust security governance measures, ensuring that your chaos engineering practices adhere to the highest security standards, enabling you to scale without compromising on safety.

Controlled Testing

With our security governance features, you can confidently expand your chaos engineering initiatives across your organization, promoting a culture of resilience and innovation while maintaining the utmost data protection and privacy.

Get onboard to the resilience journey today

Discover how your applications stand up to real-world failure scenarios. Gain insights to construct a resilient system that minimizes downtime and saves on costs.

Resources

We aim to build a community to help build a world without downtime

Achieving Continuous Resilience with Harness Chaos Engineering

Introducing the continuous resilience approach. Understand the modern approach to practicing chaos engineering where the efforts to build resilience are inserted into all stages of your software development life cycle through automated chaos experiments.

Maximizing Software Reliability While Minimizing Toil

While most teams strive to automate their reliability tasks, they don't often have the time or resources to minimize the associated toil. This webinar will share practical advice on what can be done to maximize reliability while minimizing the associated toil.

The Chaos Engineering Maturity Model

Businesses are increasingly adopting cloud-native deployments as a means to increase developer velocity. The rapid adoption of the cloud environment, including Kubernetes, has created significant complexity and revealed the inadequacy of traditional systems testing.

The Business Value of Reliability-Driven Software Delivery

Every organization needs to deliver new software features faster. But often, increased speed results in decreased reliability of application services. Without a reliable software delivery strategy, you risk revenue loss and issues with customer retention.

Chaos Experiments in Harness CD Pipelines

Chaos Engineering delivers fantastic benefits to the developers, development teams, and overall DevOps when chaos experiments are automated into the regular CD pipelines. Read this blog to know the benefits and how to use chaos experiments in Harness CD pipelines.

How Chaos Engineering Strengthens Your Disaster Recovery Plan

âHope is not a strategy.â This quote embodies the core philosophy of chaos engineering. We canât just sit around and hope that our business never experiences a costly service disruption. Itâs essential to act now and prepare for the worst by adding chaos engineering to your disaster recovery (DR) testing.

The Top Chaos Engineering Tools

Are you looking for a new chaos engineering tool? We evaluated the top open-source and commercial chaos engineering tools to consider when building your chaos toolkit.

Powered by Open Source Litmus

CNCF Hosted

Chaos experiments that are needed for a quick start in Chaos Engineering.

Chaos Observability

Litmus includes Prometheus metrics that can help measure the impact of chaos on the applications real time.

Chaos Experiments

50+ chaos faults to test system reliability.

Kubernetes Excellence

Experiments function with standard operators, CRDs, and YAML configuration that k8s users are familiar with already.

Source URL: <https://www.harness.io/products/continuous-integration>

Worldâs Fastest CIÂ Platform

Up to 4X faster than other solutions

Deliver software the Harness way!

Free developers from time-consuming, unnecessary processes that slow their work, so you and your team can focus on creating

Code

Any Source Code Manager

Integrate with GitHub, Bitbucket server & cloud, GitLab, Azure Repos and any other git provider.

Any Language

Build, test, and deploy web, mobile, APIs in Java, Go, Node.js, Swift, C#, Python, Ruby and more languages of choice.

Any Platform

Build for multiple architectures, operating systems (Linux, macOS, Windows) on a VM or in a container. Run builds on Harness Cloud or in your self-hosted infrastructure.

Build

Onboard in Seconds

Get started easily with a pipeline customized to your project.

Quickly Convert Your Existing CI/CD Pipelines

Simplify your move from GitHub Actions, GitLab, CircleCI, and other providers with Harness Migration Utility.

Thousands of Integrations

Create your own or tap into the collective wisdom of community plugins by leveraging Drone Plugins, Github Actions and Bitrise Steps in your CI Pipelines.

Enhance Collaboration and Productivity with Templates

Utilize Templates to minimize onboarding time, enforce standardization, and promote collaboration within your team.

800,780

Total Builds

1,793,456

PR Checks

87,098

Total Pipelines

2,388

Total Projects

Test

Save Costs by Running only RelevantÂ Tests

Accelerate test cycles by 80% using ML-based Test Intelligence. By running only relevant tests, you can achieve faster builds, shorter feedback loops, and significant cost savings.

Understand Code Quality with Test Insights

Analyze your tests' performance across the organization to increase efficiency and code quality.

Secure

Proactive Security Feedback

Secure every commit with automatic security scanning built into your pipeline.

Bring in your Preferred Security Scanners

Integrate with popular security scanners for execution of container, SAST, SCA, and DAST security scans, bringing robust security checks into your CI/CD pipelines.

View & Fix Vulnerabilities That Matter

Harness intelligently deduplicates, prioritizes vulnerabilities across all scanners and recommends a fix, helping you focus on the vulnerabilities that matter most.

Publish

Manage Any Artifact

Upload tests reports, logs, code coverage reports and any other build related artifacts to be viewed alongside your build.

Package & Publish Anywhere

Publish artifacts to Artifact Repositories, Container Registries(including Amazon S3, Google Cloud Storage (GCS), JFrog Artifactory, Docker Hub), Apple App Store, Google Play Store.

Deploy

Deploy Anywhere

Choose your platform, deploy your applications. Be it private/public cloud, mobile stores, serverless environments, Kubernetes, containers, VMs, Web Servers, or directly on Linux, Windows, or Mac - we've got you covered.

Progressive Delivery, ML Verification & Auto-Rollbacks

Leverage progressive delivery strategies such as Canary and Blue/Green to deploy changes gradually and with confidence. Harness continuously monitors and automatically rolls back to a previous stable version, ensuring application reliability and minimizing risk.

Govern

Robust governance for your CI/CD. Centralized policy engine, granular templates, RBAC & more!

Standardize with templates and OPA policies

Empower developers with the autonomy to edit pipelines, while central teams create 'golden' templates and enforce essential rules like 'Require container security scanning'. With OPA, we strike a balance between empowering individual teams and maintaining scalable guardrails to ensure compliance across the board.

Streamlined approvals & instant notifications

Simplify your approval processes and stay on top of pipeline events with real-time notifications. Our integration with tools like Jira, Slack, and ServiceNow ensures quick decision-making without slowing down development.

Audit trails

Maintain full visibility and readiness for audits with Harness's detailed audit trails, providing clear records of all actions and changes within your pipelines and other resources.

Enterprise grade role-based access control

Fine grained Role Based Access Control (RBAC) to allow for tailored access management, ensuring each team member has the right level of access to perform their roles effectively while maintaining security protocols.

SLSA L2 compliance without complications

Leverage Software Supply Chain Assurance to achieve SLSA L2 Compliance with ease. Harness automatically crafts SBOMs and attestations, safeguarding the integrity of your artifacts every step of the way.

Your data is our top priority

<p>Harness is committed to data security and complies with GDPR, CCPA, and other privacy laws. Read more about Security and Compliance at Harness.</p>

Standardize with templates and OPA policies

Empower developers with the autonomy to edit pipelines, while central teams create 'golden' templates and enforce essential rules like 'Require container security scanning'. With OPA, we strike a balance between empowering individual teams and maintaining scalable guardrails to ensure compliance across the board.

Streamlined approvals & instant notifications

Simplify your approval processes and stay on top of pipeline events with real-time notifications. Our integration with tools like Jira, Slack, and ServiceNow ensures quick decision-making without slowing down development.

Audit trails

Maintain full visibility and readiness for audits with Harness's detailed audit trails, providing clear records of all actions and changes within your pipelines and other resources.

Enterprise grade role-based access control

Fine grained Role Based Access Control (RBAC) to allow for tailored access management, ensuring each team member has the right level of access to perform their roles effectively while maintaining security protocols.

SLSA L2 compliance without complications

Leverage Software Supply Chain Assurance to achieve SLSA L2 Compliance with ease. Harness automatically crafts SBOMs and attestations, safeguarding the integrity of your artifacts every step of the way.

Your data is our top priority

Harness is committed to data security and complies with GDPR, CCPA, and other privacy laws. Read more about Security and Compliance at Harness.

Insights

Out-of-the-box DORA Metrics & More

Utilize Harness to achieve a clear view of your entire SDLC, promptly spot obstacles for faster software delivery. It offers detailed insights across builds, security flaws, test outcomes, deployments, etc., via comprehensive dashboards.

Plus, you have the freedom to customize and design dashboards tailored to your specific needs.

â

[Learn More ->](#)

Best in class integrations.

We know that teams live and die by their tools.

Harness integrates with, and orchestrates your entire stack.

Trusted by DevOps and Developers

Hundreds of DevOps and engineering teams are powered by Harness to become elite performers in velocity, quality, efficiency, and governance.

United Airlines

By choosing Harness for CI and CD, we were able to give the governance policies to the developers and create the guardrails we needed. Harness gives us a platform rather than just a DevOps tool.

Ancestry

âOur teams were initially skeptical of getting rid of their Jenkins pipelines for Harness, but now they're big fans of the Harness user interface.â

Xactly

âI don't hear people complain anymore about builds. I can't tell you how much better that makes me feel. Those things are hard to quantify, but so much more impactful than the dollars.â

Meltwater

âHarness CI is extremely open and flexible. There's nothing holding you back from building and deploying the way you want. Jenkins was extremely rigid and wouldn't have allowed us to grow the way we have.â

Lessonly by Seismic

âWith Harness, I've seen failures get triaged and solvedâlike completely solvedâwithout us ever being involved, which is super empowering.â

Harness Continuous Integration

Product Documentation

Learn how you can build faster and be more productive with Harness CI

Product Updates

See our latest feature releases, product improvements and announcements

Blogs

Read on for educational material, technical deep dives, Harness tutorials, and everything in between

Case Studies

Be inspired by success stories from industry leaders

Source URL: <https://www.harness.io/solutions/devsecops>

DevSecOps Done Right

Secure Artifacts from Build to Production

Shift security left without friction

Broad Scanner Support

Seamlessly integrate the open-source and commercial security scanners of your choice

One-Click Scanning

Orchestrate SAST, DAST, SCA and container scans throughout your pipeline in a single click

Comprehensive Visibility

Build a clear picture of application code issues early in the development process

Intelligently remediate vulnerabilities with AIDA

Prioritized Vulnerabilities

Automatically deduplicate and prioritize vulnerabilities

AIDA Remediation

Get instant remediation recommendations using AIDA (Artificial Intelligence Development Assistance), allowing developers to fix issues without toil.

Govern Open-Source components with confidence

Comprehensive Visibility

Get complete visibility into the usage of all open-source components in your software and track their deployment status

Enforce Governance

Implement policies to govern the usage of open-source components based on attributes such as Component Name, Version, License, Supplier or PURL

Ensure Artifact Integrity with SLSA Compliance

Generate Provenance

Establish trust by generating provenance in compliance with the SLSA v1.0 specification.

Verify Integrity

Verify SLSA provenance to confirm software integrity and safeguard against tampering before consumption

Build Secure Artifacts with Harness

Shift left and embed security into your supply chain with Harness Security Testing Orchestration (STO) and Software Supply Chain Assurance (SCCA) modules.

ZeroFlucs

To make security easy for our team to integrate into our workflow, we're leveraging the full suite of security tools from Harness — Security Testing Orchestration (STO), Software Supply Chain Assurance (SCCA) and Software Bill-of-Materials (SBOM) so that we know what's in our software, the license implications of it, and what new threats have emerged. With it now bedded in, we can be confident that all work has these practices enforced.

Over 40 scanners and growing

We know that teams live and die by their tools. Harness integrates with, and orchestrates your entire stack.

Source URL: <https://www.harness.io/products/feature-flags>

Release confidently withÂ pipelines

Manage with GitOps

Easy clean-up with flag lifecycle management

Enforce governance with OPA policies and auditÂ trails

Verify reliability with SLOs

Harness Smart Feature Flags

Optimize development velocity and reliability to drive business outcomes your customers require.

Code More, Release Faster

GitOps + Feature Flags

Bringing the power of Git as the source of truth to Feature Flags with a simple yaml configuration

Commit To Release with Pipelines

Harness closes the loop on change management by connecting from CI, to CD, to feature release to post-release monitoring

Immediate Incident Resolution

Architected for disaster-proof resilience, high availability and near-instantaneous performance

Accelerate While ControllingÂ Risk

Control your Customer Experience

Separate release from deploy, empower product and account teams to release new features, schedule when releases should happen and guarantee your customers the best experience possible

Guarantee Best Practices

From naming conventions to flag promotion across environments and blackout windows, write governance policies to enforce your best practices

24/7 Feature Monitoring

Be notified instantly if a flag change correlates to a service anomaly in production

Manage Flag Tech-Debt

Stale Flag Visibility

Bringing the power of Git as the source of truth to Feature Flags with a simple yaml configuration

Fully Customizable Reporting

Build dashboards and scheduled reports to bring visibility for all stakeholders across all teams

Automatically Remove Deprecated Flags Beta

Keep your codebase clean by automatically removing flags that are no longer needed

Open Source SDKs

Optimized for Resiliency, Performance and Security

Testimonials

"Harness Feature Flags has enabled us to implement true CI/CD. We're shortening our commit-to-production time by 3x, releasing features to customers faster and more safely than we could before."

"I wouldn't hesitate for a second to tell somebody how satisfied we are with Harness - they've been a great partner. That we continue to invest in additional solutions just emphasizes how happy we are with the relationship."

With Harness, we have decreased infrastructure cost by 60% with no performance impact as we run over 1 billion flag evaluations per day. This allows us to maintain a price advantage over competitors!

Try Smart Feature Flagging Today

Experience developer automation with GitOps

Source URL: <https://www.harness.io/events>

Events

Join us wherever you are in the world

Discover DevOps innovations with Harness at worldwide events, reshaping the future of modern software delivery.

Featured Event

Lorem ipsum dolor sit amet, consecteturLorem ipsum dolor sit amet, consectetur

Lorem ipsum dolor sit amet, consecteturLorem ipsum dolor sit amet,

DevOps Dining Club

Harness CI/CD DevDays February

Upcoming events

Harness CI/CD DevDays February

In this three hour workshop, we'll teach you how to build a pipeline that promotes artifacts and manifests to multiple environments leveraging a pull request and approval gate.

DevOps Dining Club

Welcome to the DevOps Dining Club, a series of networking events for technology leaders who seek to share and learn with each other in world-class restaurants.

DevSecOps Events with Harness

Building a meaningful DevSecOps practice is likely top of mind for your organization. Join Harness in a series of events around DevSecOps. We'll spend each event networking and diving into important DevSecOps discussion topics.

Past events gallery

Source URL: <https://www.harness.io/public-sector>

Harness Public Sector

Accelerate, optimize, and secure your agency's DevOps processes with the industry's first Intelligent Software Delivery Platform using AI/ML.

Accelerate Speed-to-Mission

- Standardize Multi and Hybrid GovCloud development and deployment workflows
- Low code/no code visual GUI Pipeline Studio speeds onboarding
- Automatically approve your pipeline's progression by integrating it into your ticketing system
- Increase velocity of cloud & microservices migration

Secure and Monitor Pipelines

- Integrate security best practices into your build phase
- Enforce security policy through approved & reusable pipelines
- Automate cATO/RMF through Security Testing Orchestration
- Normalize, deduplicate, and correlate security scanner results

Enforce Policies Across Software Delivery

- Centrally define and monitor policies that are enforced across all delivery pipelines and processes
- Built-in Secrets Management stores encrypted secrets, such as access keys
- Easy compliance audits with UI integration and YAML diff comparison per event
- Verify what specific applications, files, and data an authenticated user has access to

Do More with Less

- Eliminate manual scripting, pipeline maintenance and plug-ins
- Rely on AI/ML verification to automatically roll back failed deployments
- Automated Canary & Blue-Green deployments
- GitOps-as-a-Service provides enterprise control, governance, visibility, and 100% interoperability with the open-source Argo CD approach.

Source URL: <https://www.harness.io/products/software-supply-chain-assurance>

Ensure Artifact Integrity and Open Source Governance

Monitor and control open source components, generate comprehensive Software Bills of Materials (SBOMs) for enhanced visibility, and guarantee software integrity in accordance with SLSA and Executive Order 14028 requirements.

SBOM orchestration and management

SBOM Generation

Automatically generate Software Bill of Materials (SBOMs) with every build

SBOM Attestation

Attest SBOMs to ensure integrity and authenticity

Tool Flexibility

Use your preferred tools to generate SBOMs

Visibility & control over Open Source Software

Comprehensive Visibility

Comprehensive visibility into open-source components used in your artifacts and their deployment status

Policy Enforcement

Define and enforce policies to prevent use of harmful and risky open-source components

License Governance

Govern the usage of open-source licenses in your software

Instantly remediate Zero-Day vulnerabilities

Accurate Assessment

Get instant and accurate view of impacted artifacts and deployments

Notify Owners

Generate tickets and notifications for owners with remediation recommendations

Track Progress

Track the remediation progress & Generate compliance reports

Ensure Software Integrity using SLSA

Generate Provenance

Generate provenance as per SLSA V1.0 specifications for every build

Verify Provenance

Verify provenance before deployment for assurance

Hardened Build

Hardened build system to prevent tampering in your build process

Secure your software supply chain.

Have a question? We are here to help!

Source URL: <https://www.harness.io/products/code-repository>

Code Hosting Powered by Git

Securely host private Git repositories and collaborate on Code with advanced access controls and governance.

Familiar Git experience.

Packed with essential features.

A complete solution for engineering teams of all sizes.

Seamless Developer Experience

Code Repositories integrate seamlessly across software delivery in Harness, including award-winning CI/CD.

Top Notch Governance

Code Confidently: Ensuring every commit meets security standards.

One-click migrations

With a single click, automatically migrate code repositories from GitHub, GitLab, Bitbucket, and more.

Source URL: <https://www.harness.io/products/platform>

One that powers it all

Robust, scalable and intelligent platform that supports the development, deployment, and operation of software applications.

Every step of your SDLC

Internal Developer Portal

Facilitate developer efficiency, simplify and streamline tasks while reducing cognitive load through the Software Catalog. Empower developers with self-service capabilities and evaluate adherence to best practices using Scorecards. With Harness IDP, developers experience fewer disruptions, allowing them to focus on innovation.

Security Testing Orchestration

Shift security left without friction. Seamlessly integrate security scanners and orchestrate tests across your build and deployment pipelines. Enable developers to rapidly remediate vulnerabilities through intelligent prioritization and deduplication.

Software Supply Chain Assurance

Secure your software supply chain end-to-end. Monitor and control open source components and third-party artifacts, generate comprehensive SBOMs for enhanced visibility, and guarantee software integrity in accordance with SLSA and Executive Order 14028 requirements.

Continuous Integration

Optimizing performance through hosted builds, caching, and proprietary Test Intelligence. Designed to be blazing fast, simple, and open to help make developers' lives easier.

Continuous Deployment & GitOps

Eliminate scripting and manual deployments with powerful, easy-to-use pipelines. Empower your teams to deliver new features, faster with AI/ML for automated canary and blue/green deployments, advanced verification, and intelligent rollback.

Feature Flags

Accelerate development while controlling risks. Bring the power of Git combined with easy management of feature flag debt with Smart Feature Flags.

Service Reliability Management

Automate SLO management. Collaborate across your teams to define SLOs and error budgets aligned to developer experience and business goals. Understand the impact of all production changes.

Continuous Error Tracking

Get complete visibility into errors. Find and fix issues in minutes with deep code level visibility and deep context that streamlines debugging and remediation efforts.

Chaos Engineering

Run real-world, controlled experiments that will reveal your systems' weaknesses to avoid disasters, improve customer experience, maintain competitive advantage, accelerate digital transformation, boost developer experience, and achieve Continuous Resilience.

Software Engineering Insights

Drive data driven engineering. Assess engineering team productivity, remove roadblocks, and understand investments to increase

efficiency, improve developer experience, and deliver better customer outcomes.

Cloud Cost Management

Understand the hierarchical breakdown of cloud costs across your organization. Automate cost optimization and governance to save up to 70% on your cloud bill.

Internal Developer Portal

Facilitate developer efficiency, simplify and streamline tasks while reducing cognitive load through the Software Catalog. Empower developers with self-service capabilities and evaluate adherence to best practices using Scorecards. With Harness IDP, developers experience fewer disruptions, allowing them to focus on innovation.

Security Testing Orchestration

Shift security left without friction. Seamlessly integrate security scanners and orchestrate tests across your build and deployment pipelines. Enable developers to rapidly remediate vulnerabilities through intelligent prioritization and deduplication.

Software Supply Chain Assurance

Secure your software supply chain end-to-end. Monitor and control open source components and third-party artifacts, generate comprehensive SBOMs for enhanced visibility, and guarantee software integrity in accordance with SLSA and Executive Order 14028 requirements.

Continuous Integration

Optimizing performance through hosted builds, caching, and proprietary Test Intelligence. Designed to be blazing fast, simple, and open to help make developers' lives easier.

Continuous Deployment & GitOps

Eliminate scripting and manual deployments with powerful, easy-to-use pipelines. Empower your teams to deliver new features, faster with AI/ML for automated canary and blue/green deployments, advanced verification, and intelligent rollback.

Feature Flags

Accelerate development while controlling risks. Bring the power of Git combined with easy management of feature flag debt with Smart Feature Flags.

Service Reliability Management

Automate SLO management. Collaborate across your teams to define SLOs and error budgets aligned to developer experience and business goals. Understand the impact of all production changes.

Continuous Error Tracking

Get complete visibility into errors. Find and fix issues in minutes with deep code level visibility and deep context that streamlines debugging and remediation efforts.

Chaos Engineering

Run real-world, controlled experiments that will reveal your systems' weaknesses to avoid disasters, improve customer experience, maintain competitive advantage, accelerate digital transformation, boost developer experience, and achieve Continuous Resilience.

Software Engineering Insights

Drive data driven engineering. Assess engineering team productivity, remove roadblocks, and understand investments to increase efficiency, improve developer experience, and deliver better customer outcomes.

Cloud Cost Management

Understand the hierarchical breakdown of cloud costs across your organization. Automate cost optimization and governance to save up to 70% on your cloud bill.

Tailoring experiences to fulfill your diverse needs

DevOps

Speed up delivery with control. Use fast app development, test environments, and feature flags for user testing in production. Progressive

delivery, AI/ML verification, and policies as code ensure safe, confident frequent releases.

Our solutions:

DevSecOps

Streamline security integration in the development process, allowing developers to swiftly address vulnerabilities. Bolster governance with complete software supply chain security and detailed SBOM management for full artifact visibility.

Our solutions:

FinOps

Manage Cloud Costs with advanced visibility and automated governance. Achieve granular visibility across teams, simplify cost attribution, and reduce waste through dynamic resource detection and automated fixes.

Our solutions:

Engineering Management

Foster data-driven decisions leveraging Trellis Scores and DORA metrics, unlock agile optimization and right-size your resource investments.

Our solutions:

Harness AI assistant AIDA

Unlock the Power of AI Infused Software Delivery

Enterprise-ready, privacy-first solution

Leverage AI across the entire software delivery life cycle

Automate workflows with AI-driven assistance.

Pillars of Platform

Modular and Flexibility

- Mix and Match to organizational requirements
- Start with one module or adopt the entire platform

Platform Governance + Developer Experience

- Policy As Code
- Custom Dashboards
- RBAC
- Secrets Management
- Out of the Box Template

AI Infused SDLC

- AI Developer Assistant (AIDA)
- Continuous Verification
- Test Intelligence

Open Architecture

Out of the box integrations with hundreds of third party tools.

Best in class integrations.

We know that teams live and die by their tools. Harness integrates with, and orchestrates your entire stack.

**Try Harness
for free today**

Internal Developer Portal

Built for Developers, Crafted by Platform Engineers

Get Organized with Internal Developer Portal

Eliminate cognitive overload by letting developers manage their software effortlessly and access everything at their fingertips.

Provide developer self-service for flows like new service onboarding.

Measure service maturity while enforcing your DevOps and Development best practices.

Service Onboarding

Initiate new projects and services in minutes

Automate standing up your development and delivery toolchains. With out-of-the-box integrations, Harness Internal Development Portal can spin up and wire together repos, agile projects and pipelines.

Build Software Templates with best practices baked in

Your services are not all unique snowflakes. Establish your normal patterns and make them not just repeatable, but repeated through templates.

Create self-service flows and avoid hundreds of tickets

Standing up a new service often results in tickets flowing back and forth across the organization. Eliminate the work management overhead with a process automation.

Software Catalog

A living software registry

Centralize knowledge about services, website, data pipelines in one place. The software registry is powered by Backstage, an Open Source project by CNCF

Establish service ownership with teams

Nothing is more frustrating than needing help with a service or API and not knowing who to go to. The catalog makes ownership clear, saving time and angst.

Track dependencies across services

Who's using this? What might break if we update this API? By tracking dependencies in Harness, IDP teams have more situational awareness and can make changes with confidence.

Scorecards

Measure software maturity

Gain confidence by measuring if a service is really gold standard from a DevOps, Security and Development maturity view. Of course, you get to define what those standards are in your organization.

Drive migrations and adoption of standards

Planning to run a long upgrade of the core framework version or a programming language? Get an overview of the trend - how many applications have migrated and which ones need a push.

Enforce Best Practices

Is the API success rate good? Is the documentation up to date? Does a service use the stable versions of your tech stack? Track

everything using checks, and holding teams accountable for your best practices.

Search

Discover software components, APIs, documentation

Unblock your developers when they get stuck. Ensure they find that particular internal software or documentation which they need to use and move faster.

Extend search to include results from internal tools

You can integrate with your internal knowledge base like Confluence and provide a unified search experience for developers.

Search-driven rather than rumor-driven development

Instead of posting questions on internal forums and depending on tribal knowledge, developers can search for anything about their internal software ecosystem and get up-to-date answers.

Technical Documentation

Technical documentation for a service is a click away

Bring technical documentation closer to the software home page and allow developers to explore, staying in the same context of the software they were looking for.

Based on docs-like-code approach

By treating docs as code and keeping it in the same repository as markdown, developers update docs in same feature Pull Request. Docs never go out of date this way.

Establish feedback loop between readers and writers

Allow readers to report outdated docs directly to the developers. The feedback loop is broken today with existing documentation sites.

Plugins

Harness CI/CD

PagerDuty

Kubernetes

Firehydrant

Jira

Reduce cognitive overload

Integrate with the tools and providers you use to bring the most useful information for developers in the developer portal.

Powered by Backstage.io plugins

Rely on hundreds of plugins built by the open source community to integrate with your git, cloud provider, and monitoring, performance and DevOps tools.

Create your own custom Plugins

Did not find what you are looking for? Build your own plugin and customize your Developer Portal with the information your developers want to access.

Create Your Portal Today

Uncompromising DevOps

Deliver fast, stay in control

Code to release

Rapid feedback

Fast builds and automated pipeline progression drives feedback to developers quickly and economically

Modern deployment automation

Out-of-the-box canary deployments verified by your observability tools

Decouple deploy and release

Leverage native feature flags to control release feature by feature

Release safely

Progressive release

Progressively release with feature flags, validating that changes resonate with users

Release verification

Validate deployments using integrations with leading observability tools, optionally rolling back

Release windows

Block production deployments at sensitive times or days

Wait less. Ship more.

4x faster builds

Powered by fast cloud hardware, optimized caching, and Test Intelligence, Harness builds faster.

Ephemeral environments

Deploy to temporary environments to test more quickly while keeping costs under control.

Hold back a feature, not a release

Stop waiting to release new features because the business isn't ready for some other feature. Decouple with feature flags.

Developer friendly, audit approved

Configuration as code

Pipelines and Flags as code in Git. Clear change audit logs and visibility.

Versioned templates

Versioned Templates at multiple levels to ensure you can reuse best practices without copy-and-paste sprawl.

Policy-as-code

Open Policy Agent (OPA) enables developer control of pipelines while enforcing organizational standards.

Stale feature flags

Automatic detection of stale feature flags, helps teams detect and remediate technical debt.

AI that makes delivery smoother

Test intelligence

Machine learning selects only the tests impacted by a code change driving better feedback faster.

Intelligent rollback

Machine learning selects only the tests impacted by a code change driving better feedback faster.

Explanation of failures

Build and deploy errors are processed through AI to deliver quick explanations and remediation ideas.

Trusted by DevOps and Developers

Hundreds of DevOps and engineering teams are powered by Harness to become elite performers in velocity, quality, efficiency, and governance.

United

By choosing Harness for CI and CD, we were able to give the governance policies to the developers and create the guardrails we needed. Harness gives us a platform rather than just a DevOps tool.

Entur

We chose Harness for the compelling user interface and intuitive workflows. It's a modern, state-of-the-art solution for software delivery.

Ancestry

Harness now empowers Ancestry to implement new features once and then automatically extend those across every pipeline, representing an 80-to-1 reduction in developer effort.

Integrations

We know that teams live and die by their tools. Harness integrates with, and orchestrates your entire stack.

Source URL: <https://www.harness.io/solutions/finops-excellence>

Unleashing Cloud Efficiency and FinOps Mastery

Empowering FinOps, CloudOps and Engineering Teams to Take Control of Cloud Spend

Automated cost optimization

Automate cost optimization across clusters and cloud providers

Reduce costs up to 70% by managing idle non-production resources with Cloud AutoStopping®

Take advantage of cloud provider savings with automated spot orchestration

Take control of your Kubernetes deployments with AI powered cluster autoscaling

Governance-as-code

Create and enforce cloud cost policies and compliance at scale

Stop chasing engineers to take action with automated Cloud Asset Governance

Find and eliminate cloud waste automatically with governance-as-code

AI-powered rule creation keeps asset management simple

Commitment orchestration

Maximize commitment utilization, compute coverage and realized savings

No more wasted capacity, purchase what you need, as you need it

Never miss a contract deadline with automated contract management

Granular cost reporting

Deliver visibility to drive transparency and accountability

Empower your teams with real-time cost visibility, delivered in your business context

All-in-one cost reporting across cloud providers and cloud resources in a single dashboard view

Go deeper into your Kubernetes costs, utilization and efficiency

Trusted by DevOps and Developers

Hundreds of FinOps and engineering teams are powered by Harness to become elite performers in velocity, quality, efficiency, and governance.

H2O.ai

Using Harness Cloud Asset Governance, we've transitioned to a FinOps-as-Code model, enforcing cost governance policies at scale, in real-time. We've uncovered over \$35,000 per month in cost savings during our first 30 days of using the feature, and expect to continue to see more over time.

Trusted by DevOps and Developers

Hundreds of FinOps and engineering teams are powered by Harness to become elite performers in velocity, quality, efficiency, and governance.

Synopsys

"All developers in Synopsys have access to recommendations within Harness CCM. We recommend that developers use this information when deploying new microservices to the cloud. This is what makes this cloud cost management tool a game changer for us as we balance the speed of innovation with its cost."

Carvana

âWe understand now, when we run this environment for five days, this is what the cost is. So now I'm going to run it for three days or I'm going to power it off when I don't need it. Cloud Cost Management helps us immensely.â

Best In Class Open Source Integrations.

We know that teams live and die by their tools. Harness integrates with, and orchestrates your entire stack.

Certification

Harness is proud to be a member of the FinOps Foundation

Source URL: <https://www.harness.io/learn/resource-center>

Resources

Browse and register for e-books, guides, webinars, videos and upcoming events!

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/products/security-testing-orchestration>

Comprehensive Built-In Security Testing Orchestration

Secure your applications at the speed of development

Shift-left security built for your pipelines, designed for developers

Seamlessly integrate security scanners and orchestrate tests anywhere across your build pipelines. Enable developers to rapidly remediate vulnerabilities through intelligent prioritization and deduplication.

Automated CI/CD & security testing

Orchestrate security scans in the pipeline

Easily configure and run AppSec scans with Harness CI/CD stages or in a standalone mode, integrating with any CI/CD tooling

Flexible integrations and scanner support

Natively integrate with over 40 open source and commercial security scanners and create custom integrations to support your scanner of choice. Monitor issues through turnkey integrations with issue tracking systems.

Fast fixes for developers

Rapidly prioritize vulnerabilities

Fix consequential security vulnerabilities and reduce security noise with intelligent organization and deduplication.

Fix fast with AI remediation guidance

Leverage AI enhanced remediation guidance and contextual information to apply the right fixes with minimal triage.

Simplified Vulnerability Management

Single pane of glass

Get centralized visibility into deduplicated security findings based on projects, pipelines or applications of interest.

Grant and manage exemptions

Manage security risk, priorities, and exceptions with time bound two-step exemption management.

Enhanced Governance

Strengthen security posture across your SDLC

Create customized policies with centralized security governance templates powered by OPA and granular RBAC.

Streamline compliance

Enforce mission critical compliance without compromising quality or velocity of software delivery.

Over 40 scanners and growing

Automatically invoke the top security scanners to quickly identify and remediate security vulnerabilities within the layers of your complex applications.

Trusted by DevOps and Developers

Hundreds of DevOps and engineering teams are powered by Harness to become elite performers in velocity, quality, efficiency, and governance.

Using Harness Security Testing Orchestration for a single pipeline, Deluxe identified 170 issues from a scanning vendor, narrowed to nine prioritized problems post-deduplication. The team highlighted a 95% noise reduction, allowing efficient focus on top issues.

[Learn more about](#)

Harness Security Testing Orchestration

Product Documentation

Learn how to connect STO with your existing tech stack and get insights. How to remove bottlenecks and improve planning and sprint hygiene

Product Updates

See our latest feature releases, product improvements and announcements

Blogs

Read on for educational material, technical deep dives, Harness tutorials, and everything in between

Case Studies

Be inspired by success stories from industry leaders

Source URL: <https://www.harness.io/comparison-guide>

Compare DevOps Tools with Harness

There are several DevOps tools to choose from. However, only one was built with you, the end-user, in mind. The Harness CI/CD Platform is a complete end-to-end solution for the modern DevOps team.

Comparison Guide

See how Harness compares to the other DevOps tools in the market.

Harness Continuous Delivery & GitOps

Harness Continuous Integration

Cloud Cost Management

Harness Feature Flags

Security Testing Orchestration

Service Reliability Management

Harness Chaos Engineering

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/solutions/engineering-excellence>

Unlocking Engineering Excellence

Leveraging Data and Intelligence to propel Engineering Outcomes

Optimizing planning, alignment and execution

Improve planning and predictability

Enhance sprint predictability and oversight

Utilize scorecards and dashboards to improve sprint predictability

Identify anomalies and unplanned work, such as missing story points

Unlock Intelligence to enhance product management

Control project scope and delivery

Assess the on-time delivery of new features and changes

Effectively reduce project scope creep to ensure projects stay on track

Align your engineering efforts to business outcomes

Resource allocation analysis

Ensure engineering resources are aligned with business requirements

Get a comprehensive breakdown of resource allocation (features, bugs, debt)

Strategic business focus

Evaluate alignment with business plans and objectives

Identify and mitigate wasteful or unplanned engineering work

Supercharge your execution

Boost developer productivity

Utilize the Harness Trellis Framework for insights into developer productivity

Generate a composite report highlighting opportunities for your team

Unlocking efficiency

Leverage DORA to gain actionable insights into process bottlenecks

Realize quick value through the widget library included in Software Engineering Insights

Trusted by DevOps and Developers

Hundreds of DevOps and engineering teams are powered by Harness to become elite performers in velocity, quality, efficiency, and governance.

Broadcom

We've seen incredible improvement in engineering productivity. The technology automates our KPIs and removes bottlenecks, reducing our developers' task burden by 30% in certain areas, enabling them to focus on more rewarding work.

Source URL: <https://www.harness.io/customers>

Trusted by Hundreds of Businesses

Hundreds of DevOps and engineering teams are powered by Harness to become elite performers in velocity, quality, efficiency, and

governance.

We were facing the prospect of re-writing our deployments from scratch, stringing together a bunch of scripts and CLI stuff, or choosing Harness which ended up saving us \$1.2 million in CI/CD maintenance.

Jeff Green, CTO, Tyler Technologies

Harness CI is extremely open and flexible. Thereâs nothing holding you back from building and deploying the way you want. Jenkins was extremely rigid and wouldnât have allowed us to grow the way we have.

Jim Sheldon, Principle Software Engineer at Meltwater

Harness Feature Flags has enabled us to implement true CI/CD. Weâre shortening our commit-to-production time by 3x, releasing features to customers faster and more safely than we could before.

Sam Hall, Head of Technology, Metrikus

Our environment was heavily under-optimized, but we've seen a 60 to 70% cost savings without any hassle for our infrastructure management teams.

Dheemanth R, CTO, Discover Dollar

Our pipelines were complex and not cloud agnostic. What took engineers days or weeks can now be done in hours with Harness.

Nick Willson, TechOps Lead, GoSpotCheck

Explore Customer Stories

Sensormatic optimizes retail operations with Harness SEI

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Advanced Achieves Cloud Cost Governance Excellence, Saves 33% on Cloud Costs with Harness CCM

Advanced was struggling with cost overruns, and lack of adoption of the previous cost management tool. After implementing Harness Cloud Cost Management, Advanced has seen 33% annualized cost savings to-date.

H2O.ai Saves \$35,000 in First 30 Days with Cloud Asset Governance from Harness Cloud Cost Management

Tyler Technologies reaches \$1.2M annualized cost savings with Harness Cloud Cost Management

Learn how Tyler Technologies reached \$1.2M annualized cost savings with Harness Cloud Cost Management

Burst SMS accelerates application modernization with Harness

Burst SMS has accelerated its application modernization journey with Harness, achieving cost savings, improved reporting, efficient deployments, and positioning itself for future growth.

SheerID Accelerates Deployments by 35X, Eliminates Burden On Developers by Using Harness

Using Harness, SheerID accelerated deployment velocity, increased developer productivityâ allowing more time to focus on innovationâ and successfully implemented canary releases during its transition to a continuous deployment cycle.

Deluxe Corporation Modernizes Deployments and Automates DevSecOps Processes with Harness

When Deluxe was struggling with reduced time-to-market for new features, they turned to harness to automate CI/CD processes and shift application security left.

United Airlines Accelerates Deployments by 75% With Harness

As part of its digital transformation efforts, United moved from monolithic applications to microservices. Legacy tools couldnât cope with this increase in scalability.

Skillsoft Moves to Harness CD to Reduce Frustrations and Workloads While Increasing Velocity

As Skillsoft's software platform expanded to more than 100 microservices, the engineering team looked to Harness to standardize deployments, enable scalability.

Ulta Beauty Speeds Time-to-Market By Easing and Accelerating Deployments

Ulta Beauty sought to improve guest experience by developing its digital store of the future. Harness CD helped achieve this, increasing deployment volumes 50x.

Ancestry Adds Consistency and Governance to Cut Downtime and Systems Onboarding Effort

To hit 99.9% uptime, Ancestry's deployment processes had to change. Find out how Harness helped build a robust, consistent, and governed CI/CD process.

ZeroFlucs Reduces Infrastructure Cost by 60% with Feature Flag-Triggered Optimization

ZeroFlucs increased efficiency and reliability for customers with Harness Feature Flags.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

The Warehouse Group Decreases Lead Time For Changes by 99.1%

Harness CD has helped The Warehouse Group standardize the deployment process and empower their dev squads with easy integration, speed to production, and a range of efficiency features.

UWM Deploys in Minutes Instead of Hours

UWM leverages Harness for self-service Kubernetes deployments and infrastructure creation, reducing deployment time from hours to minutes!

Xactly's Journey from Shadow IT to a Single Pane of Glass CI

With Harness CI, every build from commit to completion takes the same amount of time for Xactly. There is no queueing or build failures. This single consistent performance capability has helped with productivity immensely.

Soulcycle Reduced CI/CD Time & Effort by 80%

Learn how Soulcycle evaluated open-source and Harness for CI/CD and reduced their deployment effort by 80%.

Tyler Technologies Saved \$1.2 Million in CI/CD Maintenance Costs

Learn how Tyler Technologies saved \$1.2 million in CI/CD maintenance costs

Tyler Tech Takes CI/CD to the Next Level and Achieves Unparalleled Velocity With Feature Flags

To Tyler Tech, feature flags were a natural extension of CI/CD. Learn why they chose - and trusted - Harness to provide that capability.

LogMeIn Standardized CI/CD Pipelines Across Teams

Learn how LogMeIn onboards new microservices in just one hour.

Iterable Triples Deployment Speed & Reduces Downtime Risk by 85%

Find out how Iterable deployed faster and more securely.

MakerBot Replaced Spinnaker & Saved \$275,000

Learn how MakerBot replaced Spinnaker and reduced CI/CD costs by \$275,000

Relativity Selects Harness Over Spinnaker for Secure Continuous Delivery

Relativity selects Harness over Spinnaker resulting in massive security improvements.

SMU Reduces Rollback & Deployment Time by Over 80%

SMU becomes a deployment powerhouse by leveraging Harness.

Ruckus Scaled Kubernetes Continuous Delivery With GitOps

Learn how Ruckus scaled Kubernetes Continuous Delivery using Harness GitOps

Single Digits Saves \$108k in Deployment Effort

Single Digits made their migration easy with Harness. See how you can too.

Relativity Reduced Kubernetes Cloud Costs by Millions in 5 Months

Learn how Relativity democratized cloud cost data across engineering and product teams with Harness Cloud Cost Management.

Lavasoft Reduces Developer Toil by 94% Using Drone and Harness

Lavasoft uses both Drone and Harness to create the ideal software delivery environment.

Burst SMS Removes Outage Risk for Their Customers

Harness delivers better DevOps practices to all our customers. Find out how Burst SMS used Harness to reduce outages.

Priceline Accelerates Move to GCP and Canary Deployments Using Harness

Harness helped Priceline quickly and easily build deployment pipelines from scratch for its migration to GCP and GKE. Find out how.

Raisin Runs from Jenkins Nightmare à Saves 525k per Year

Raisin removed their CD maintenance migraine, find out how you can too!

Jobvite Reduces Deployment Time by 90%, Surges Deployment Volumes, and Saves \$500k With Harness

Harness is core to Jobvite's deployment scale and growth, without impacting developers' day-to-day work with tools and process changes. Find out how.

Particle Reduces Human Error in Deployments

Harness was the right out-of the box solution to help Particle deploy software more effectively.

ConsumerTrack Onboards Kubernetes Apps in 3 Hours

ConsumerTrack saves \$580k with Harness.

GoSpotCheck Masters Microservices CI/CD

Learn how GoSpotCheck creates true Continuous Delivery pipelines using Harness.

Meltwater Runs 1200 Pipelines Per Day With Harness CI

Learn how Meltwater builds 1200 artifacts a day.

Metrikus Shortens Commit-to-Production Time by 66%

Metrikus made feature flag management easy with Harness. See how you can too.

Lessonly by Seismic Says Goodbye to Jenkins and Toil With Harness CI Enterprise

Learn how Lessonly by Seismic went from toil with Jenkins to peace of mind with Harness CIE.

Tilting Point Migrates to Kubernetes With One Developer and Harness

Find out how Tilting Point sped up their cloud migration with Harness.

intelliflo Repurposed 2 FTEs + Increased Deployment Velocity 20x

Find out how intelliflo reduced their time spent on Jenkins maintenance.

How Zeotap Increased Deployment Velocity 36x

From Downtime Windows to Deployment Nirvana: Learn How Zeotap Increased Deployment Velocity 36x.

Datastax Selected Harness over Spinnaker for CI/CD

Learn how Datastax upgraded their CI/CD with Harness

Advanced Empowers 700 Engineers With Harness

Advanced saw a 10x return on their investment in 2 months. Find out how.

Linedata Achieves Global Continuous Delivery with Harness

Linedata saved \$500k in developer toil. Find out how you can too.

Vuclip Empowers Developers With Harness

Vuclip found that Harness was the only solution that fully integrated with their GCP, Kubernetes, APM, and Log stack.

Marketron Triples Deployment Frequency & Decreases Deployment Time 75%

Find out how Marketron tripled velocity with Harness.

ABC Fitness Eliminates Delivery Toil

Learn how ABC Fitness drastically reduced the effort spend on software delivery.

Discover Dollar Saves 70% On Their Cloud Bill

Discover Dollar made idle cost management easy with Harness. See how you can too.

Entur Improves Deployment Frequency from Twice a Week to 14 Times a Day, Rollback Time from Two Hours to Five Minutes

As the number of development teams nearly doubled, Entur looked to Harness CD & Feature Flags to gain more control in production, increase deployment frequency.

Lessonly Democratizes Continuous Delivery Using Harness

Find out how Lessonly distributed the deployment workload to its developers.

Choice Hotels Deploys 85% Faster Using Harness

Harness helps us capitalize on the investments we've made in automation tools by streamlining everything into a single pipeline.

Build.com Automated CI/CD Rollbacks to 30 Seconds

Build.com removed most of their verification effort. Find out how you can too.

Carvana Cuts Infrastructure Provisioning Time by 92%, Saves \$1.4 Million With Granular Cloud Cost Management

Harness helped Carvana decrease manual effort and increase scalability during a huge growth period. Then, as Carvana needed to cut costs, Harness helped Carvana build on those automation gains while providing deeper and more granular insights into its cloud infrastructure costs.Â

Campspot Reduces Outage Risk by 78%

Campspot turned to Harness for reliable deployments with less outage risk.

Beachbody Accelerates AWS Cloud Migration

Beachbody avoided the headache associated with cloud migrations by leveraging Harness continuous delivery.

Casting Networks Accelerates Kubernetes Migration by Four Months

Without Harness, Casting Networks would be 4 months behind their Kubernetes migration schedule.

" Our pipelines were complex and not cloud agnostic. What took engineers days or weeks can now be done in hours with Harness.

The Modern Software Delivery Platform®

Need more info? Contact Sales

Source URL: <https://www.harness.io/products/cloud-cost-management>

Control Cloud costs with Intelligent Automation

Automate and save up to 70% on your cloud bill.

Gain insights into detailed unit-level cloud costs.

Automate and save up to 70% on your cloud bill. Gain insights into detailed unit-level cloud costs.

Cost Reporting

Attribute & Manage Costs with Granular Visibility

Simplify hierarchical cost attribution for chargeback and show-back. Allocate shared and complex resource costs. Reuse team, department, BU definitions across reports.

Features: Perspectives and Cost Categories

Cost Optimization

Automated Cost Optimization: Dynamic Idle Resource Detection & Remediation

Automatically shut down idle cloud resources when inactive, bring them back when activity is detected saving up to 70%.

Feature: Cloud AutoStopping™

Cost Governance

Manage Cloud Assets with Governance-As-Code

Automated governance-as-code with cost, security, and compliance policies. Right-size under-utilized resources and clean up unused ones for any cloud, resource, and action. AI-powered policy generation with Harness AIDA.™

Feature: Cloud Asset Governance

Integrations

Trusted by DevOps and Developers

By choosing Harness for CI and CD, we were able to give the governance policies to the developers and create the guardrails we needed. Harness gives us a platform rather than just a DevOps tool.

Synopsys

"All developers in Synopsys have access to recommendations within Harness CCM. We recommend that developers use this information when deploying new microservices to the cloud. This is what makes this cloud cost management tool a game changer for us as we balance the speed of innovation with its cost."

Carvana

"We understand now, when we run this environment for five days, this is what the cost is. So now I'm going to run it for three days or I'm going to power it off when I don't need it. Cloud Cost Management helps us immensely."

Industry Recognition

Get started with smart Cloud Cost Management

Take control of your cloud costs with intelligent automation and granular cost visibility

Source URL: <https://www.harness.io/products/service-reliability-management>

Automated SLO Management

Collaborate across your teams to define SLOs and Error Budgets aligned to user happiness and business goals. Delight your customers by deploying code faster with better reliability.

SLO configuration in minutes

Define SLOs, SLIs and track Error Budget burn rates for all your services across multiple observability data sources. Increase transparency across your organization by collaborating with your development, deployment and reliability teams in a single place.

Change Impact Analysis

Resolve issues faster by understanding the impact of every change related to your deployments, infrastructure, feature flags, chaos experiments, incidents and more on the SLO performance and health of your service.

Automated Governance

Use reliability guardrails in your pipeline templates, along with your SLO data to determine if deployments should proceed. Allow your best developers continue to deliver highly reliable software at high velocity while putting guardrails in place for other developers or sensitive projects.

Configure observability integrations and more

Harness applies AI and ML to observability data to determine if software is reliable.

Try Service Reliability Management today

Deploy faster with better reliability

Source URL: <https://www.harness.io/products/aida>

Harness AIDA™

AI Development Assistant

Unlock the power of AI infused software delivery

- Enterprise-ready, privacy-first solution
- Leverage AI across the entire software delivery life cycle
- Automate workflows with AI-driven assistance

Build and Deployment Troubleshooting

Explain and Fix Vulnerabilities

Auto-Generate Cloud Asset Policies

Build and Deployment troubleshooting

Reduce time to analyze failures

Accelerate debugging and analyzing failures to reduce time to resolution

Streamline development

Instantly identify errors and root causes without parsing thousands of log lines

Supercharge developer productivity

Fast, actionable remediations for build and deployment failures

Explain and fix vulnerabilities

Understand every vulnerability

Leverage AI to quickly analyze vulnerabilities and secure applications

Remediate faster

Auto-generated code suggestions to guide the resolution

Ship secure

Enhance developer autonomy to proactively address security risks

auto-generate cloud asset policies

Generate policies with natural language

Implement cost-savings measures without needing to understand complex code and infrastructure

Explain policies

Generate rules and policies that can be applied directly to cloud assets

Manage savings and democratize policy expertise

Disseminating policy expertise across multiple teams

Harness AIDA™ Roadmap

Onboard Assist

Easily onboard your teams to scale out best practices in software delivery

Generate OPA Policy

Create governance rules in the CI/CD pipeline to ensure risks are managed

Auto-generate Dashboards

Use auto-generated dashboards to measure metrics important for your business

And more...

Capabilities of AIDA infused across software delivery

Resources

Try Harness AIDA™ Today

See what your team could achieve with Harness.

Source URL: <https://www.harness.io/learn/use-cases/self-service>

Self-Service CI/CD

Software delivery enables software changes of all types to reach users safely, quickly, and repeatably. CI/CD pipelines enable predictable routines for developers to deliver their work. Self-service CI/CD provides visibility, features, and access control to allow individuals to deliver their work without depending on someone else.

What is build and test automation?

A build-and-test automation process provides rapid time to release by incorporating development and quality assurance workflows.

How to automate build and test with Harness.

Harness automates the build and test process through easy-to-use Continuous Integration platform features.

Essential Reads

What are the core elements of a CI/CD pipeline? This blog post highlights build and test elements, characteristics of good CI/CD pipelines, and provides examples for you to get started on your build and test automation journey.

Build and artifact-centric quality steps are prudent in a Continuous Integration pipeline; at the same time, testing often requires a deployment and gets handled in the CI/CD pipeline. Learn how to best move your first tests into pipeline.

What is Rollback?

Rollbacks are deployments of a previously known version of an application. They restore services in case of any issues.

How to rollback deployments with Harness.

Rollback a deployment with Harness. Harness initiates a rollback of the most-recent successful deployment onto various platforms, providing rapid, predictable recovery from a failed deployment.

Essential Reads

Internet retailer Build.com migrates from Jenkins to Harness. See how they automated their entire CD process with automatic verification and rollback capabilities using Harness.

Learn how OpenBank (Digital Bank of Santander) empowered engineers using Harness's Continuous Verification. See how they dropped rollback time 96%, from 2-3 hours to 5-10 minutes.

What is Secrets Management?

Secrets management is managing digital authentication credentials, such as passwords, API keys, OAuth tokens, and ssh keys.

How Harness manages Secrets.

Harness includes a built-in secrets management feature that enables the storing of encrypted secrets. With Harness, you can also use third-party secrets managers such as HashiCorp Vault, Azure Key Vault, CyberArk, or AWS Secrets Manager.

Essential Reads

If you've ever posted a private key to your code repository, then you've shared a secret. This three-part series on security will share how to manage modern security solutions for the cloud-native ecosystem.

Secrets management, audit trails, fine-grained RBAC - these are ways to leverage governance into your CI/CD pipelines in order to satisfy compliance requirements and ensure systems are secure. See how governance can prevent you from suffering a cyber attack.

What are Audit Trails?

Audit trails are security logs that record events, changes, and other activities.

How Harness manages Audit Trails.

Audits provide us with answers to who, what, when, and where. Harness helps you with your audit and compliance needs. The audit trails feature provides records of all events and changes to your services and accounts.

Essential Reads

Learn about audit trailing in the context of Continuous Delivery (the who, what, and when of all activity relating to the contents, dependencies, and execution of your deployment pipelines). Read our deep dive article into audit trails at Harness.

Secrets management, audit trails, fine-grained RBAC - these are ways to leverage governance into your CI/CD pipelines in order to satisfy compliance requirements and ensure systems are secure. See how governance can prevent you from suffering a cyber attack.

What is Verification?

In software delivery, verification is a phase in the software development life cycle where software is evaluated to run and function as intended.

How Harness manages Verification.

Harness helps you monitor the health of your deployments by aggregating multiple providers into a single interface, using machine learning to identify anomalies, and performing automatic rollbacks or other actions.

Essential Reads

The main challenge facing deployments is validating the health of newly deployed service instances. With canary analysis, machine learning models learn normal application behavior to identify anomalies that can be flagged in future deployments.

Harness Continuous Verification uses machine learning to auto-verify deployments by analyzing time-series metrics from APM solutions. See an overview of our Continuous Verification dashboard and support.

What are Deployment Strategies?

Deployment strategies are practices used to change or upgrade a running instance of an application.

How Harness manages deployment strategies.

From basic to multi-stage service deployments. Harness supports a variety of deployment strategies, including canary and blue-green deployments.

Essential Reads

Learn about various deployment strategies to release software into production in a way that reduces risk and exposure to customers and the business. Methodologies include blue/green and canary deployments.

Understand your deployment strategies in the context of Kubernetes clusters. This post includes a closer look at the pros/cons of canary and blue/green strategies for Kubernetes deployments.

What is GitOps?

GitOps centers itself around a version control system such as git to control the software delivery lifecycle. Leverage a git repo as a source of truth for changes and easily auto-sync changes to Kubernetes clusters. You can have confidence that changes managed through git are active within a Kubernetes environment.

How Harness manages GitOps.

The Harness GitOps integration allows you to use Git as the single source of truth while maintaining the state of the deployment process in Harness. Opt to sync changes between Harness and your git repository and accelerate your GitOps capabilities.

Essential Reads

Learn how Ruckus Networks, a wireless networking equipment and software company, leveraged Harness GitOps to manage pipeline, service, environment, Helm, and Kubernetes configuration all in one place as code.

The buzz around GitOps has grown given the recent increase in demand for automation. Watch this 30-minute technical webinar to get a high-level overview on instituting GitOps in your org.

What is Change Management?

In software development, change management defines how individuals, teams, and organizations make changes to an application or service.

How change management is implemented with Harness.

Harness helps track and automate your change management process and tools. Harness integrates with Jira and ServiceNow to allow you to audit and control your deployments and pipelines.

Essential Reads

As part of the Continuous Delivery playbook, consider how to provision infrastructure, how to manage change approvals, how to release and rollback when working with CD cross-functional delivery teams.

Learn how change approval fits into automated enterprise governance within delivery pipelines.

Resources

Browse and register for e-books, guides, webinars, videos and upcoming events!

The Modern Software Delivery Platform®

Need more info? Contact Sales

Source URL: <https://www.harness.io/pricing>

Pricing & Plans

- This is some text inside of a div block.
- This is some text inside of a div block.

Pricing FAQ

Our free plan is the fastest and easiest method to start building and deploying with Harness. It is available to customers of all sizes from students, individual developers, startups, mid-size organizations to most demanding enterprise businesses. Best of all, the access doesn't expire, and no credit card is needed unless you choose to upgrade to our Team or Enterprise Plans.

- This is some text inside of a div block.
- This is some text inside of a div block.
- This is some text inside of a div block.
- This is some text inside of a div block.
- This is some text inside of a div block.
- This is some text inside of a div block.
- This is some text inside of a div block.

Pricing FAQ

Our free plan is the fastest and easiest method to start building and deploying with Harness. It is available to customers of all sizes from students, individual developers, startups, mid-size organizations to most demanding enterprise businesses. Best of all, the access doesn't expire, and no credit card is needed unless you choose to upgrade to our Team or Enterprise Plans.

- This is some text inside of a div block.
- This is some text inside of a div block.

Pricing FAQ

Our free plan is the fastest and easiest method to start building and deploying with Harness. It is available to customers of all sizes from students, individual developers, startups, mid-size organizations to most demanding enterprise businesses. Best of all, the access doesn't expire, and no credit card is needed unless you choose to upgrade to our Team or Enterprise Plans.

- This is some text inside of a div block.
- This is some text inside of a div block.

Pricing FAQ

Our free plan is the fastest and easiest method to start building and deploying with Harness. It is available to customers of all sizes from students, individual developers, startups, mid-size organizations to most demanding enterprise businesses. Best of all, the access doesn't expire, and no credit card is needed unless you choose to upgrade to our Team or Enterprise Plans.

- This is some text inside of a div block.
- This is some text inside of a div block.

Pricing FAQ

Our free plan is the fastest and easiest method to start building and deploying with Harness. It is available to customers of all sizes from students, individual developers, startups, mid-size organizations to most demanding enterprise businesses. Best of all, the access doesn't expire, and no credit card is needed unless you choose to upgrade to our Team or Enterprise Plans.

- This is some text inside of a div block.
- This is some text inside of a div block.

Pricing FAQ

Our free plan is the fastest and easiest method to start building and deploying with Harness. It is available to customers of all sizes from students, individual developers, startups, mid-size organizations to most demanding enterprise businesses. Best of all, the access doesn't expire, and no credit card is needed unless you choose to upgrade to our Team or Enterprise Plans.

- This is some text inside of a div block.
- This is some text inside of a div block.

Pricing FAQ

Our free plan is the fastest and easiest method to start building and deploying with Harness. It is available to customers of all sizes from students, individual developers, startups, mid-size organizations to most demanding enterprise businesses. Best of all, the access doesn't expire, and no credit card is needed unless you choose to upgrade to our Team or Enterprise Plans.

- This is some text inside of a div block.
- This is some text inside of a div block.

Jeff Green, CTO, Tyler Technologies

Jim Sheldon, Principle Software Engineer at Meltwater

Sam Hall, Head of Technology, Metrikus

Dheemanth R, CTO, Discover Dollar

Nick Willson, TechOps Lead, GoSpotCheck

Source URL: <https://www.harness.io/blog>

Harness Blog

Read on for educational material, technical deep dives, CI/CD tool comparisons, Harness tutorials, and everything in between.

ArgoCD, Terraform and Harness

Learn how to combine Terraform with ArgoCD to use GitOps for infrastructure and application.

CI/CD for Serverless

Ah âserverlessâ... the promise of running applications without worrying about servers! The term "serverless" often sparks skepticism among those rooted in traditional software development practices. The idea of handing over control over infrastructure may seem daunting, and developers might wonder if serverless is even right for their use case. My goal in this blog is to demystify serverless computing (in particular, serverless deployment) without unnecessary complexity.

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

Discover how Harness Software Engineering Insights (SEI) revolutionizes code quality and productivity. Explore the Trellis Framework, quality metrics, and structured engineering improvements for unparalleled software development. Learn more in our comprehensive guide.

Hygiene in SDLC: A Key to Engineering Efficiency

Explore the importance of hygiene in software development with our blog. Discover how it ensures top-notch software quality and insights. Get practical tips for improvement and join us in fostering a culture of continuous excellence. Prioritize quality and efficiency for success in the software world!

Navigate a New Way: Your Dedicated User Experience

Introducing Nav 2.0, a new design that updates the information hierarchy of our platform to provide a curated user experience for multiple personae.

Push Helm Charts to Container Registries with Harness CI

Learn how to publish OCI Helm charts to a Container registry

January 2024 Product Updates

Harness Product Updates. Learn what has changed in Harness over the previous month.

Harness Acquires Armory Assets

We're excited to announce that Harness has acquired key intellectual property and technology from Armory.

5 Reasons to Attend Chaos Carnival

Harness is excited to host the third annual Chaos Carnival, a FREE two-day virtual conference on March 15-16, 2023. So, have you registered yet? Here are five reasons why you should register today!

How to Contribute to the Harness CLI

Lessons learned from committing to a new project, and to open source in general

How the size of a Pull Request can impact developer experience?

Learn how the size of a Pull Request can significantly impact developer experience, and discover how Harness SEI helps in optimizing PRs to improve code quality, reduce review times, and enhance collaboration for your engineering team.

Retiring Harness CD Community Edition in favor of Gitness

We are retiring the Harness CD Community Edition in favor of Gitness to deliver the true promise of a simple CD pipeline.

December 2023 Product Updates

Harness Product Updates. Learn what has changed in Harness over the previous month.

Webinar: Increased Velocity with Better Control

Field CTO Martin Reynolds joins The Register's Tim Phillips to discuss the challenges of modern software delivery.

Unleashing the Power of Generative AI: Transforming the Inner and Outer Loop Dynamics in Development

GenAI has transformed development, particularly in balancing the creative tasks of the 'inner loop' with the repetitive tasks of the 'outer loop'. Read more about the need for a seamless, AI-enhanced development environment that minimizes bottlenecks and maximizes developer productivity in this blog.

Ephemeral CI environments using ttl.sh and Gitness

Read this blog to learn how Gitness and ttl.sh can be used to create temporary CI environments to expedite development processes.

Getting Started with the React SDK

What is going on with all of my amazing friends in the developer community? In this post we are taking a look at getting the Harness React SDK up and running in your project. This article is a low barrier to entry with the core code to get started. In the example below we are going to take what the React SDK GitHub has to offer and shorten it to get you started with your first feature flag in React.

ShipTalk Podcast - Starting an SRE Practice in a Critical Industry

In this episode of ShipTalk, we get to talk to Dr. Vlad Ukis, Head of R&D for Siemens Healthineers and author of Establishing SRE Foundations.

Unlocking Efficiency: Exploring Churn Rate with Harness Software Engineering Insights (SEI)

Explore the dynamic world of Churn Rate and its role in revolutionizing software delivery with Harness Software Engineering Insights (SEI). Uncover actionable insights, optimize workflows, and navigate the seas of development challenges with precision.

Announcing the Harness SRM Backstage Plugin

We are excited to introduce the latest addition to our suite of Open Source Backstage Plugins - the Harness SRM Backstage Plugin. This plugin is designed to seamlessly integrate with your Backstage Instance as well as with Harness IDP to help with development team's workflow, ensuring that Service Level Objectives (SLOs) and error budgets are not just a metric but a part of your daily development practice.

Using AI To Improve Feature Flags Onboarding

Read about how Harness Feature Flags enhances the developer experience by integrating Harness AIDAâ¢ for customized, AI-powered SDK onboarding. Get tailored code samples specific to your language, version, and framework, ensuring a streamlined and efficient setup process.

Start or Expand Your DevSecOps Education Journey - Introducing STO Developer Certification

Getting certified at the developer level on Harness Security Testing Orchestration is a great milestone in your DevSecOps journey.

Harness Chaos Engineering Faults Landscape

Harness Chaos Engineering, powered by LitmusChaos, provides a wide range of chaos faults that can be used to verify resilience of your system systematically, incrementally and continuously. This article covers the anatomy of a Harness Chaos Engineering experiment and types of chaos faults that the product supports.

Run Ruby Builds Up To 4x Faster Using Harness CI Test Intelligence

Announcing Ruby Test Intelligence preview for Harness CI! Run only the subset of tests relevant to your Ruby code changes and skip the rest.

Simplifying Policy Creation and Management with Harness AIDAâ¢

Open Policy Agent (OPA) Policy Creation will become a whole lot easier with the help of AIDA Policy as Code Assistant. Dive into this blog to learn about how our new feature will revolutionize policy creation with cutting-edge AI technology.

Drone 3.0 - redefining next 10 years of simplicity & innovation in open source CI/CD

Learn about the exciting innovations happening in Drone.

November 2023 Product Updates

Harness Product Updates. Learn what has changed in Harness over the previous month.

Securing CI/CD Images with Cosign and OPA

This blog examines container image security in CI/CD, focusing on tools like Cosign for image signing and Open Policy Agent (OPA) for policy enforcement. It offers a comparative look at various image signing tools, explores methods to ensure image integrity and compliance with organizational standards, and includes practical implementation guidance for Kubernetes, along with a hands-on tutorial.

Bitbucket Server's sunset on February 15th, 2024: A Evolving to streamlined DevSecOps from a disjointed past

Explore a post-Bitbucket era of software delivery with Harness , where AI meets DevSecOps for a superior, forward-thinking software delivery experience.

KubeCon 2023 NA Recap - Developer Experience is Monumental

KubeCon + CloudNativeCon North America has wrapped up, and the buzz it generated is still in the air. With the halls of Chicago's convention center now quiet, those of us who attended are left with a wealth of new insights and ideas.

Architecting for Cost Savings on BigQuery

Cloud infrastructure architecture can have a significant impact on cloud costs. For Google BigQuery, the way you store and access data is a key consideration for both performance and costs. Read how Harness' internal engineering team tackled this challenge.

Trunk Based Development with Feature Flags

Hey all you wonderful developers out there! In this post I am going to take you through creating a feature branch and using the Harness React SDK to implement a flag in your project. This post is an introduction to getting a feature setup using Vite, React, and Sass to change the look of a logo in a navbar.

Increase Developer Velocity with AIDA and Harness CI

See how AIDA helps developers quickly troubleshoot and resolve build failures in Harness CI.

BackstageCon 2023 Recap: Developer Experience Top of Mind

Internal Developer Portals or IDPs are designed to drive DX. The hallmark Open Source IDP, Backstage, has its community coming together at a co-located event at KubeCon North America 2023 for the second year. Harness is proud to support and sponsor the event.

Dreading the upcoming Bitbucket Server support deadline? Try Gitness.

Learn how Gitness can replace Bitbucket Server before Atlassian ends support on February 2nd, 2024

ArgoCon 2023 Recap: GitOps Achieves Escape Velocity

ArgoCon North America 2023 in Chicago brought Kubernetes experts together to discuss the Argo Project's impact on deployment, with Harness spotlighting their Continuous Delivery offering that leverages Argo CD for simplified artifact promotion and custom approval flows.

Harness Doubles Down on GitOps

Harness has added support for Flux and Argo Rollouts.

Why Make Dashboards When AIDA Can Do It For You?

AIDA can now make your dashboards for you with just a chat prompt. Simply ask AIDA what you want to see, and watch your dashboard be built for you one widget at a time.

Road to BackstageCon 2023: A Sneak Peek into an Exciting Lineup & A Recap of 2022!

The Second edition of BackstageCon is round the corner as part of the co-located event in Kubecon and CloudNativeCon North America 2023, this blog post is a glimpse of the talks that are featured in this edition of the event, as well as gives a recap of the 2022 edition of the event.

Lower Infrastructure Costs with Arm; Get there Faster with Harness CI Cloud

The Arm architecture offers significant potential cost and energy savings. Build Arm Docker images five times faster than emulation using Harness CI Cloud.

Got Monorepos Instead of Microservices? This is How Harness IDP Has Got You Covered

Backstage, an open-source developer portal, has become a favorite for many organizations, especially those leaning towards microservices. But what about those still using or transitioning from monorepos?

Adventures at MagnoliaJS

Last week I had the great pleasure to speak at a very intimate developer community event in Jackson, Mississippi. Check out this post to read about my awesome adventure at MagnoliaJS.

Cost Reporting for Chargeback and Showback: Why Choosing Harness Beats Building In-House

Chargeback and Showback are key functions for FinOps teams to provide visibility into and hold engineering teams accountable for cloud costs. Many companies struggle with the decision to develop their own internal solutions or purchase a third party tool. Here are some considerations that can help you drive that decision.

Harness Developer Hub - Ease of Authoring with Git Triggers

Harness Developer Hub [HDH] leverages Git Triggers to deliver developer education to the world. Learn about how HDH gets published.

Migration Made Easy - Automatically Migrate From Drone To Harness CI

Explore how the Harness CI Migration Utility simplifies and accelerates the transition from Drone to Harness, automating the bulk of migration tasks to save you time and effort.

CD and the Modern Software Shop

CD doesn't need to feel overwhelming. Learn how we arrived where we are, and the best practices needed to modernize continuous delivery and deployment in your organization.

Architecting for Cost Savings at Tyler Technologies

Your cloud architecture plays a significant role in your ability to control cloud costs as your business grows. Tyler Technologies realized this, and took action to optimize the organization of the shared architecture their clients relied on in order to maximize Cloud AutoStopping savings.

Why Platform Engineers Trust Harness

Harness is tailored to the needs of Platform Engineers. Recently, Gartner rated highly in a Platform Engineering use case.

Get Actionable Insights with AIDAâ€¢, CD Error AnalyzerÂ

Discover how the AIDA CD Pipeline Error Analyzer is changing the game as an AI tool for engineers. AIDA can identify and help resolve failures in your continuous delivery pipelines. Learn why it works and how it can improve your DevOps process.

Road to KubeCon 2023 - Join Harness at KubeCon NA 2023

Harness will be back at KubeCon NA this year and also participating in Co-Located events. Learn about our latest contributions and predictions.

October 2023 Product Updates

Harness Product Updates. Learn what has changed in Harness over the previous month.

Need for Automation - GitOps at Scale

Think of GitOps like building software: you start with some initial tests, set a strong foundation, then expand with confidence. This blog takes you from early GitOps experiments (Day 0), diving deeper into configurations (Day 1), to navigating the advanced automation challenges (Day 2). Along the way, we'll delve into how Terraform can address the bootstrapping dilemma â setting up the very GitOps tools you want to automate with.

Hate reading docs? Me too. That's why we have Support powered by AIDAâ€¢

Discover how Harness is revolutionizing software delivery with the introduction of AIDAâ€¢s newest capability: Support. Dive into the transformative power of AI-driven customer service and learn how we are infusing AI to create a seamless user experience.

Release Management with Feature Flags

Follow these tips on how to enhance your release management using feature flags.

Deploying to Kubernetes with Gitness

Gitness offers more than just code hosting. In this blog, we'll explore its continuous delivery features and demonstrate how to deploy applications to Kubernetes using Gitness. Along the way, we'll delve into the DevOps cycle, address the concerns associated with using continuous integration tools for deployment tasks, and highlight the advantages of employing dedicated tools like Harness Continuous Delivery for intricate deployments.

Using Feature Flags for Effective Incident Management

Any application is prone to an incident, no matter its resilience. Learn how to use Feature Flags for effective incident management.

The Journey to Using Feature Flags at Their Full Potential

Feature flags may seem straightforward but actually require some strategy to use at their full potential. Follow these 4 steps to get your team mastering feature flags.

Leveraging Feature Flags for Zero-Downtime Database Migrations

Learn how to avoid downtime during your database migration with Harness Feature Flags.

How We Use Our Own Feature Flags at Harness

Let's take a look at how Feature Flags are used by our very own teams here at Harness with a case study on our Customer Success Management team.

Safe Testing in Production with Harness Feature Flags

Testing in production may seem like a risky task, but in reality, it can be the safest way to ensure a working environment. Learn when and how to test in production using Harness Feature Flags.

Elevating Reliability in Software: Harnessing Smart Feature Flags for Velocity and Performance

Software reliability is crucial for a brand's reputation and overall business success in today's digital age. Every software glitch, downtime, or inconsistent user experience could translate to lost customers, negative reviews, and reduced revenue.

Demystifying Trunk-Based Development

Take a journey with me into trunk-based development! We are going to talk about what TBD is, breaking it down (demystifying), and how to get started. TBD is a great way for your dev team to get started branching efficiently.

Delegates and Agents - Onramp to Scale with Harness

By leveraging a Harness Delegate or GitOps Agent, you are able to model commands you would have to maintain locally for remote execution.

Harness Unveils Four Game-Changing Modules

At our {unscripted} conference, Harness introduced 4 new modules: Code Repository, Internal Developer Portal, Infrastructure as Code Management, and Software Supply Chain Assurance.

Harness Announces Engineering Leadership Community to Codify Engineering Excellence Standards

Elevating AIDA: Harness Unveils 7 New Innovative Capabilities

Harness is proud to announce 7 new AIDA features that span across the software delivery lifecycle. These capabilities empower engineers and team members with tools that increase productivity, streamline software delivery processes, and enhance governance.Â

Gitness: Your Ultimate Open Source Development Platform

Gitness is a new alternative to the limited open source version control solutions available with the breadth and depth of features to empower development teams to write and deliver code with safety and speed.

Introducing Harness Infrastructure as Code Management

We are now happy to announce the launch of a new module the provide CI/CD solution for infrastructure changes

Step into Tomorrow: Harness Internal Developer Portal is Ready for the Public

Learn how developers are getting ship done with Harness Internal Developer Portal. This is your exclusive invitation to the future of software development.

Introducing Harness Software Supply Chain Assurance (SSCA)

Boost Your DevSecOps Practice With SLSA-based Artifact Integrity and SBOM-driven Open Source Governance

Harness Releases Gitnessâ Open Source Git Platform

Release of first significant open source developer platform in nearly a decade offers a unified solution for source code management and CI pipelines, improving collaboration, enhancing security, and automating tasks in software development workflows.

Become Harness Certified in CI and CD & GitOps at Any Level

Harness now offers three levels of certifications in both our CI and CD & GitOps modules.

Accelerating Harness CD & GitOps Learning for New Users

To solve the need of new user learning especially for the Harness CD & GitOps Free plan, we are pleased to announce the public preview of a new âGet Startedâ UI that takes the user through a guided learning journey.

Introducing the Harness CLI

We are pleased to announce the public preview launch of the new Harness CLI.

How Quizlet Automates Pull Requests for Cloud Cost Recommendations

Learn how Quizlet empowered their engineers to do more with cloud cost recommendations by creating an automated pull request workflow that increased engineering engagement by 75%, and saved over 40% on their Kubernetes workloads.

AI-Assisted DevOps: My First AIDAâ¢ Experience

Let's discuss AI in DevOps and why it's a game-changer for software delivery teams.

September 2023 Product Updates

Harness Product Updates. Learn what has changed in Harness over the previous month.

Introducing Flag Archiving and Restoration for Smart Feature Flags

You now can archive and restore smart feature flags giving developers more flexibility to manage and address issues with the deletion of a flag.

Harness Announces the 4th Edition of Chaos Engineering Conference focusing on Software Resilience

Save the date for January 24th-25th, 2024, as Harness brings back Chaos Carnival, the annual worldwide conference on Chaos Engineering.

The Impact of Github Copilot on Developer Productivity: A Case Study

There's a Good Chance Your Company's Cloud Costs Are Sky High. Take These 5 Steps to Save Money on Them.

Every year, businesses waste \$180 billion on unnecessary cloud spending. Here's how to fight back.

Trunk-Based vs. Feature-Based Development

Dive into the world of Git-based branching strategies with our easy-to-follow guide on trunk-based and feature-based development. Learn the pros and cons of each, and learn how CI/CD fits in.

HashiCorp's BSL License Change: What this means for Harness CD customers?

Harness Cloud Asset Governance powered by AIDA: The Path to a Well Managed Cloud

Harness Cloud Asset Governance helps customers find and eliminate cloud waste automatically, while also providing a policy engine to ensure cloud resources are in compliance with corporate standards. By leveraging policy-as-code, it automates resource optimization, security, and compliance tasks, freeing your engineers to focus on creating innovative products and services that drive your revenue.

Charting Software Delivery Flight Paths: Intro to GitOps Source Patterns

GitOps offers a declarative approach to deployments, modeling operations as software architecture. Explore the concepts and practical challenges surrounding common GitOps source patterns, and how Harness can inject velocity and governance into whichever model you choose.

Growing the Harness Developer Hub

Learn about new and improved content on the Harness Developer Hub. These updates, based on customer feedback, are aimed to enhance clarity, improve accuracy, and create a structure that better follows the user journey.

Harness joins OpenTF Initiative

Harness is excited to join the OpenTF Initiative, a community-driven fork of the popular Terraform project

Harnessing Next-Gen AI for Secure Software Delivery: A Look at Harness AIDA

Dive into the transformative world of DevOps and DevSecOps, where Harness Security Testing Orchestration leverages the power of GenAI with Google Vertex AI, ensuring a secure and efficient software delivery process.

A/B Testing For Feature Flags, What It Is And What It Shouldn't Be

Feature Flags and A/B testing often go hand in hand - but they probably shouldn't!

Hierarchical Breakdown of Organizational Cloud Costs

Harness Cost Categories offer a way to group cloud costs in ways that are most meaningful for their business and act as the basis for creating and managing Cost Perspectives, which are cost views customized to the needs of your business that empower teams to track and manage costs directly.

Unit Testing vs. Integration Testing

Explore the key differences between integration tests and unit tests in the realm of software development. Learn how these testing methods contribute to building robust and reliable software systems. Discover their unique roles, benefits, and how they integrate into the CI/CD pipeline for delivering top-quality software.

August 2023 Product Updates

Harness Product Updates. Learn what has changed in Harness over the previous month.

Automate Your CI/CD Pipeline Using Triggers

This article talks about how triggers can be used to automate the CI/CD pipeline. It also explains the variety of triggers that can be used to automate the entire CI/CD pipeline.

Advanced Deployment Strategies

This article talks about the different deployment strategies along with the best practices.

Harnessing The Power of Secrets Management

Introducing the Harness Idea Portal: Share Your Insights, Shape Our Future

Today, we're excited to announce that we're changing how we engage with our customers to collect feedback and prioritize our roadmap.

CI/CD Pipeline as Code with Harness

How Organizations Can Reduce DevOps Costs with CI/CD Pipeline Templates

Optimizing Facebook's Proxygen Project Build Time with Harness CI: A Case Study

In this article, we build Facebook's Proxygen project on the Harness CI module. Additionally, we conduct a comprehensive analysis, comparing the build time obtained from Harness CI with that of Github Actions, which happens to be Proxygen's default CI tool.

Easily Migrate Continuous Integration Pipelines to Harness

Explore Harness CI Migration Utility, our latest innovation for effortless CI migration

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/learn/use-cases/spinnaker>

Spinnaker Not Enough?

If you are struggling to deliver your software or to build your software delivery solution, it may be time to consider Software as a Service. Supported SaaS CI/CD solutions take the hassle out of configuration, bugs, and platform support. Harness makes Continuous Delivery easy and provides platform support for hosted, or self-managed instances.

What are the challenges posed by Spinnaker?

Spinnaker was created and designed by Netflix, which is opinionated about where you can deploy but not how you can deploy. Spinnaker focused primarily on the cloud stack that Netflix operated and allowed users/implementations to be left to their own devices on how to deploy, verify, and rollback. Users of the Spinnaker Platform often mention the steep learning curve required to install and administer the platform.

How Harness wins on platform support.

Unlike Spinnaker, Harness is designed to be ubiquitous with platforms to deploy to. Compared to other solutions which are opinionated on where to deploy to, e.g. only Kubernetes, Harness has the ability to deploy anywhere.

Essential Reads

Compare Spinnaker vs. Harness for Continuous Delivery, including their capabilities, verification, test automation, microservices deployment flow control, security, support, and more.

In this post, we discuss the setup and hierarchical differences between Spinnaker and Harness. This will give a detailed understanding of the required effort of setting up one over the other.

What is Open Source?

Open Source or Open Source Software is where software's source code is made available under license. Open Source Software can potentially be open for contributors, thus making it a software development strategy. Licenses typically can enable or restrict the use of source code, derivative works, and distribution.

How Harness supports open source.

Harness loves Open Source. Our engineers contribute to Open Source Projects. With the recent investment into Drone.io, a wildly-popular Open Source Project, Harness is committed to being a good steward of the Open Source Community.

How Harness supports Customer Migrations from Spinnaker.

When investigating a Continuous Delivery platform, it is common to compare Spinnaker to Harness. Several customers have migrated or are in the process of migrating off Spinnaker and to Harness. Our Pre-Sales and Solution Architecture Team has been through the journey with several customers.

Essential Reads

SaaS company Relativity selects Harness over Spinnaker for Continuous Delivery. See why Harness provided the best value and how the Relativity team increased deployment by 30x in 3 months using Harness.

DataStax, builders of the NoSQL data platform on Apache Cassandra, upgraded their CI/CD using Harness. Learn why their challenges with Spinnaker led them to Harness CD, and view the results they achieved for their team of 25+ engineers.

Source URL: <https://www.harness.io/learn/use-cases/public-cloud>

Public Cloud Migration

Harness migrated its own SaaS environment from one public cloud to another in just 76 minutes. Keep your software delivery dynamic and flexible. Harness Continuous Delivery provides a cloud-agnostic abstraction model supporting your deployments whatever the target infrastructure.

What is Public Cloud Migration?

A Public Cloud Migration involves moving data, applications, and other business services into a cloud computing environment such as Amazon Web Services, Google Cloud, and Microsoft Azure.

How Harness supports public cloud migration.

Harness supports public cloud migrations by providing an environment agnostic delivery platform. Utilize infrastructure as code to provision deployment environments.

Essential Reads

Beachbody, a leading provider of fitness/nutrition programs with over 23 million customers, accelerates AWS Cloud migration from 1 year to 3 months using Harness. Their journey involved migrating multiple services to AWS, Kubernetes, and Lambda.

Harness helps Advanced, the UK's third largest provider of business software and services, automate software delivery across its 100+ product teams, 700+ software engineers, 6 data centers and several technology stacks/tools. These include AWS EC2, ECS, EKS and Lambda for public cloud.

What is Cloud Cost Optimization?

Cloud Cost Optimization is the process of reducing the overall cloud cost spend by right-sizing resources, eliminating waste, and identifying mismanaged resources.

How Harness provides Cloud Cost Optimizations.

Harness's Cloud Cost Management is purpose-built to find and eliminate cloud cost overages. Acting as a watchful eye on your cloud infrastructure, coupled with the deployment capabilities of the Harness Platform, Cloud Cost Management can recommend and help implement savings immediately.

Essential Reads

In this case study, learn how Harness saved over \$140k in logging costs by optimizing our cloud usage and logging spend using Cloud Cost Management.

Optimize Kubernetes cluster costs using Harness Cloud Cost Management, which allows you to know exactly when and where costs spike. Receive a quick intro to the Cloud Cost Management product in this blog post.

What are microservices?

Microservices are small, autonomous services that work together.

How microservices work with Harness.

Harness can orchestrate even the most complex Microservice deployments. As applications decompose and the number of services increase, the number of deployments increase. Harness can handle multiple deployment patterns.

Essential Reads

Casting Networks, a cutting-edge entertainment technology company, hit several roadblocks while migrating their legacy monolith applications to Kubernetes microservices. Learn how they quickly onboarded their microservices with Harness.

GoSpotCheck, a company empowering field teams to collect merchandising, sales, and compliance data, used Harness to migrate several applications and microservices between cloud providers and regions in just hours with zero downtime. Learn about their journey.

What is Container Orchestration?

Container orchestrators manage the lifecycle of containers and other computer resources.

How Container Orchestration works with Harness.

Harness can interact with multiple container orchestrators. No matter if deploying to Kubernetes or Amazon ECS, Harness has the ability to be ubiquitous when interacting with a container orchestration. Harness out of the box provides scaffold deployments to several popular orchestrators.

Essential Reads

CI/CD SaaS tools primarily focus on Kubernetes, but have forgotten about ECS customers along the way. Harness understands that many customers are happy with their ECS environments and don't have plans to move to Kubernetes. See how the Harness ECS Experience empowers developers to automate their deployments without having to manage deployment scripts and various failure conditions during deployment.

Learn about the modern-day container orchestrator and why it's become so important to your Kubernetes journey. Also see how Harness provides a powerful orchestration layer working with your Kubernetes investment.

What is Infrastructure as Code?

Infrastructure as Code is the practice of provisioning, managing, and configuring computer infrastructure as such networks, load balancers, and machines.

How Infrastructure as Code relates to Harness.

Harness has the ability to interact and manage with several infrastructure-as-code providers. As part of a modern deployment, infrastructure is provisioned at deployment time. Harness can orchestrate popular infrastructure-as-code providers such as Terraform and CloudFormation.

Essential Reads

DataStax, the company behind the massively scalable, highly-available, cloud-native NoSQL data platform built on Apache Cassandra®, selects Harness over Spinnaker. Dynamic infrastructure provisioning with Harness is just one reason.

Relativity speaks to how Harness integrates with Relativity's entire toolchain consisting of Jenkins for CI, HashiCorp Terraform for Infra provisioning, New Relic for APM, Splunk for Log Analytics, and Prometheus for Infra monitoring.

Source URL: <https://www.harness.io/company/careers>

Join the Team!

Thinking about pursuing a career opportunity at Harness? Join us in taking software delivery into the future. We're always on the lookout for exceptional talent to join our fast growing global team!

We promote a culture of growth and success, for our business as much as our people

We welcome and commemorate the diverse backgrounds and multifaceted lives of our global teams, and we recognize that bringing your whole self to work creates a place where people can thrive.

Harness's mission is to enable the 30 million software developers in the world to deliver code to their users quickly, reliably and efficiently.

Our Core Value

Be Bold

We are pioneers in defining this market space, pushing the boundaries of continuous delivery and beyond.

Get Ship Done

Delivering is in our DNA. We always find a way to get ship done and deliver on time.

Know Your Customer

We invest deeply in understanding and anticipating our customers' needs to make them successful because their success is our success.

Stay Humble

We value humility over egos,
and compassion over self-centeredness.

Build Trust through Transparency

We prioritize transparency and trust our leaders and team members to make the right decisions.

Continuously Improve

We celebrate our wins, learn from our losses, and are not afraid to fail.

Remember the Human

We welcome and commemorate our global teams' diverse backgrounds and multifaceted lives, and recognize that bringing your whole self to work creates an environment where people can thrive.

Celebrate Together

We celebrate milestones, special occasions, and wins together, and recognize the individuals that Get Ship Done. We like who we work with and that makes everyday fun!

Diversity, Equity, Inclusion and Belonging at Harness

At Harness, being yourself is not only embraced but celebrated. We welcome and commemorate the diverse backgrounds and multifaceted lives of our global teams, and we recognize that bringing your whole self to work creates a place where people can thrive. We believe in the power of our people's potential, and we know that if we want them to reach it, we need to create an inclusive environment where people can do their best work. At Harness, we have woven diversity, equity, inclusion, and belonging into the fabric of our company by consistently focusing on increasing cultural awareness by celebrating different cultures and underrepresented groups with a variety of internal events.

Women at Harness

We believe that women bring invaluable qualities into the workplace: communication styles, thought processes, problem-solving approaches, and creativity channels. We strive to foster a female-friendly work environment that will attract and retain the best women in the workforce.

Pride in Tech at Harness

In the efforts of building a diverse workforce, we stand strong with the LGBTQ community and advocating for LGBTQ rights. We are committed to provide a safe, accepting, and welcome workplace.

Life at Harness

We are a company that strives to be the best in the industry, while remaining an enjoyable place to work. We are fortunate to have the opportunity to come together virtually and in-person across the globe.

Life at Harness embraces all our employees with global holiday and milestone celebrations, TGIF off program, and morale boosting activities all while Getting Ship Done!

Celebrating Earth Month

Fireside Chat

Social Hours

Hack Week

At Harness, being yourself is not only embraced but celebrated

Ice Cream Social

We recognize and commemorate milestones, special occasions, and achievements together

Summer Parties and Events

Benefits & Perks

Competitive Salary

Healthcare Benefits

Flexible Time Off (US only)

Employee Referral Bonus

Learning & Development Program

Employee Assistance Program

Employee Recognition Program

Paid Time Off & Parental Leave

Social and Team Building Events

Recharge & Reset Program

Employee Testimonials

Harness is an Equal Opportunity Employer and Values Diversity in All its Forms.

We do not discriminate on the basis of race, religion, color, national origin, gender, sexual orientation, age, marital status, veteran status, or disability status.

Press & news

Partners

Contact us

We're hiring!

We're looking for amazing people to come build with us. Want to join the team?

Source URL: <https://www.harness.io/learn/use-cases/verification>

Deployment Verification

Organizations ensure quality through software verification. To test a software component or system, organizations should ensure that services are installed, configured, and run in their intended test environments. Verification also involves ensuring that services perform their functions within an acceptable time, respond to inputs, and achieve a general result.

What is APM?

APM stands for Application Performance Management. APM tools such as AppDynamics, Datadog, and Dynatrace enable organizations to keep track of application performance and monitor for regression.

How does Harness use APM?

Harness leverages APM Platforms and other markers to determine deployment health. A more complex task such as verifying and validating a canary deployment for promotion is handled in an automated fashion.

Essential Reads

Learn why performance monitoring is critical, and how Harness leverages APM and machine learning for its CI/CD pipelines.

This blog post goes over the two key reasons to use an Application Performance Monitoring (APM) solution from a data science perspective, and how Harness leverages APM in its 24/7 Service Guard solution.

What is Observability?

Observability is the measure of how well internal states of a system can be inferred from the knowledge of the external outputs.

How does Harness use Observability?

Harness leverages Observability Platforms and other markers to determine deployment health. A more complex task such as verifying and validating a canary deployment for promotion is handled in an automated fashion.

Essential Reads

A CI/CD Evangelist presents an overview of Observability for Verification and a summary of how to scale efficient pipelines. Learn how customers use the Harness platform to deliver 80% faster and more securely.

What is observability, and how are APM solutions solving the challenges of observability for modern enterprise applications? Learn more on why observability matters more and more in software delivery.

What are Accelerate Metrics?

The book Accelerate by Nicole Forsgren, Jez Humble, and Gene Kim has four key metrics to measure software team performance. Deployment frequency, lead time for changes, mean time to restore [MTTR], and change failure rate are the four key metrics mentioned.

How Harness uses Accelerate Metrics?

Harness has the ability to show Accelerate Metrics as part of its dashboard capabilities.

Essential Reads

Harness allows DevOps teams/leaders to measure software delivery performance using accelerate metrics. See it in action in this blog post.

Learn why delivery metrics matter in the context of Site Reliability Engineering, and how innovating and delivering features quickly can go hand-in-hand with SRE.

What are Performance Gates?

Gates in software delivery are some sort of criteria that you have to pass before proceeding. Performance gates are designed to ensure that performance expectations, e.g. SLAs/SLOs, are met before fully being deployed into production.

How Harness uses performance gates.

Performance gates can be before, during, or after a deployment. Performance gates can be crucial for making judgment calls if a deployment is going to be or being successful. Harness has native integrations with several APM and Observability tools along with internal markers to determine health.

Essential Reads

With the container and microservices bloom, we deal with larger and larger distributed systems. The topologies continue to grow as we bake in redundancy and performance SLAs.

Kubernetes Deployments

Kubernetes is a container orchestration platform allowing organizations to scale their services and workloads quickly. If you are working with containers or microservices, Kubernetes may be a great use case for you.

What are Kubernetes Deployments?

Kubernetes deployments are container image deployments which target Kubernetes-based environments.

How Harness supports Kubernetes deployments.

Harness has first class support for Kubernetes Deployments. As the Kubernetes ecosystem evolves, Harness makes Kubernetes Deployments simple. The Harness Platform can manage several types of deployments in the ecosystem such as Helm. Enabling canary deployments with automatic verification is core to the Harness Platform.

Essential Reads

Learn how Relativity, a software platform with over 180k users in 40+ countries, reduced their Kubernetes costs across engineering and product teams by millions with Harness Cloud Cost Management.

Tilting Point, an award-winning game publisher, migrates to Kubernetes with just one developer using Harness. See how they saved over \$100k in labor costs and tripled deployment frequency.

What is Helm & Kustomize?

Helm is a package manager for Kubernetes. Similar to Homebrew, RPM, and YUM, Helm deploys charts that define application packages. Kustomize is a tool that customizes Kubernetes manifests files. Kustomize is a configuration management solution for Kubernetes.

How Harness supports Helm and Kustomize.

Harness has native support for Helm and Kustomize based deployments. You can leverage your favorite package and configuration managers as managed resources inside a Harness CI/CD Pipeline.

Essential Reads

Harness recently released support for Helm V3, the latest version of Helm, inside our platform. Let's take a look at leveraging your first Helm V3 deployment.

Read or watch a video to learn how to execute your first Helm deployment with Harness (using Helm V3).

What is Istio à Service Mesh?

Istio is a popular service mesh implementation that provides connection, security, control, and observability to microservices.

How Harness supports Istio Service Mesh.

Harness has native Istio support allowing for seamless canary and blue/green deployments. Harness can manage the traffic-splitting capabilities of Istio, allowing for seamless deployments and for your engineers to not worry about underlying Istio complexities.

Essential Reads

In this first post in a blog series, learn how to carry out your first Istio deployment using Harness. Service Mesh is democratizing networking rules to dev teams, and we're here to help it all happen.

Get your Service Mesh adoption cheat sheet, and learn how Harness can help you perform Istio deployments with ease. Includes a 4-minute video tutorial.

What are Kubernetes Resources à Kubernetes Manifest, CRDs, and Operators?

Kubernetes is a resource-based platform. Typical consumers of the platform will be creating and modifying different Kubernetes Resources. These resources have their lifecycle managed by Kubernetes to some degree. The first point of interaction with the declarative end state of Kubernetes is a manifest, e.g. a deployment.yaml. Other resources such as a Custom Resource Definition/CRD and Operators can extend the Kubernetes platform.

How Harness supports Kubernetes Resources à Manifest, CRDs, and Operators.

Harness has first-class support for Kubernetes Resources. Harness can create scaffolding around Kubernetes Resources removing complexities around crafting your own resource definitions that are purpose made for deployments. Harness can offer granular deployment lifecycle support around different Kubernetes Resources supporting canary and blue/green deployments inside Kubernetes.

Essential Reads

With operators extending and leveraging Controllers and Custom Resource Definitions (CRDs), the possibility is there to have Kubernetes react in very specific ways for your application and application infrastructure. Learn about their role in the future of K8s.

Understand the why, and the how of Kubernetes. This course is geared towards a technical audience that is new to Kubernetes. It's quite a comprehensive introduction, so even those with some exposure will get something out of it.

What are Canary Deployments?

Canary Deployments are a progressive delivery pattern for rolling out releases to a subset of users.

How Harness supports Canary Deployments.

Harness is the defacto platform for Canary Deployments. Canary Deployments can be complex because of the multiple phases and the judgment call of when to promote or rollback a canary. The Harness Platform has smart verification taking away the manual toil in verification and enables seamless Canary Deployments.

Essential Reads

From setup to pipeline execution, building a fully automated canary deployment from scratch in Harness takes just 4 minutes.

Canary deployments are one of the most popular release methods, as it is both an effective and cost-efficient methodology. View an illustrated explanation of how canary deployments can be utilized.

What is GitOps?

Secrets management is managing digital authentication credentials, such as passwords, API keys, OAuth tokens, and ssh keys.

How Harness manages GitOps.

The Harness GitOps integration allows you to use Git as the single source of truth while maintaining the state of the deployment process in Harness. Opt to sync changes between Harness and your git repository and accelerate your GitOps capabilities.

Essential Reads

Learn how Ruckus Networks, a wireless networking equipment and software company, leveraged Harness GitOps to manage pipeline, service, environment, Helm, and Kubernetes configuration all in one place as code.

The buzz around GitOps has grown given the recent increase in demand for automation. Watch this 30-minute technical webinar to get a high-level overview on instituting GitOps in your org.

Source URL: <https://www.harness.io/learn/use-cases/devsecops>

DevSecOps à Compliance & Governance

DevSecOps is short for development, security, and operations, and it is how organizations deliver and make security decisions and actions within their valued deliverables. CI/CD pipelines help organizations with their DevSecOps to continuously integrate security in the software development lifecycle.

What is Security Scanning?

Security scanning, including container and vulnerability scanning, allows you to detect vulnerabilities in deployable artifacts and running applications. Container scanning can refer to scanning the base image or the running container for known vulnerabilities / security exposures. Containers can have several layers all with third party open source powering parts of the container which need to be regularly scanned.

How security scanning works with Harness.

Harness has the ability to call container scanning and vulnerability scanning tools as part of a CI/CD pipeline. Passing a container scan can be a quality gate in a Harness Pipeline Stage before moving on to a deployment. Passing a vulnerability scan can be a quality gate before a build is packaged and deployed.

Essential Reads

SaaS provider LogMeIn found its legacy tool deployments became too complex to manage and standardized its Continuous Delivery with Harness. Since migrating to Harness, they've reduced deployment time, toil, and effort by over 95%.

What is Pipeline Compliance?

Pipeline compliance is the ability for a pipeline to adhere to a certain standard, i.e. conformance, or the ability to have controls in place, i.e. governance.

How Harness addresses pipeline compliance.

Harness can enforce pipeline compliance and also pipeline conformance in several ways. Standardization is a driving factor around compliance providing the guard rails. Harness has the ability to leverage templates and has configuration-as-code which can be managed in a Source Code Management [SCM] solution. With RBAC, controls can support a wide set of users who need to view and users who need edit. Harness can also score conformance to pipeline standards with the Harness Pipeline Governance feature.

Essential Reads

Regulatory and operational compliance is critical in software development. Harness's Pipeline Governance feature will allow you to measure how compliant your pipelines are with your regulatory and operations standards.

In an organization where developers are continuously pushing code to production, managing risks can be difficult. This piece details pipeline compliance under the umbrella of governance, risk management, and compliance (GRC).

What is Secrets Management?

Sensitive properties and passwords should not be stored in plain text. Modern approaches are to store sensitive information as encrypted secrets. To manage the lifecycle of a secret, e.g updates and secret injection, secret management solutions are available.

How Harness manages Secrets.

Harness includes a built-in secrets management feature that enables the storing of encrypted secrets. With Harness, you can also use third-party secrets managers such as HashiCorp Vault, Azure Key Vault, CyberArk, or AWS Secrets Manager.

Essential Reads

If you've ever posted a private key to your code repository, then you've shared a secret. This three-part series on security will share how to manage modern security solutions for the cloud-native ecosystem.

What is Auditing?

Auditing in a technology-sense is the examination of evidence and controls. Having systematic record of why and where a pipeline ran is crucial in an audit.

How Harness addresses Auditing.

Audits provide us with answers to who, what, when, and where. Harness helps you with your audit and compliance needs. The audit trails feature provides records of all events and changes to your services and accounts.

Essential Reads

Learn about audit trailing in the context of Continuous Delivery (the who, what, and when of all activity relating to the contents, dependencies, and execution of your deployment pipelines). Watch our short 3-minute video on how audit trails work in Harness specifically.

Source URL: <https://www.harness.io/learn/use-cases/cloud-cost>

Cloud Cost Management

The flexibility of tools and services within the cloud adds to the required skills of a team. The cloud offers scalability, but developers also need to manage enterprise scale and complexity. Cloud computing is not all that we expect when it comes to cost.

What is Cost Visibility?

From an engineering standpoint, there is a disconnect between what resources an application is taking up in the public cloud and the

actual dollar and cents cost incurred, which usually sits with a finance team. Cost visibility allows the consumer, e.g. the engineer of the cloud resources, to gain a clear and concise view of what they are actually using in their applications.

How Harness provides cloud cost visibility.

Harness Cloud Cost Management is a dedicated product focusing on cloud cost visibility. A challenge with cloud cost visibility is disseminating information to engineers specifically for their work. Answering the question âhow much is my application costingâ is difficult, especially with the multi-tenant or multi-app nature of modern cloud native infrastructure. Harness Cloud Cost Management unlocks cloud cost visibility and solves these problems.

Essential Reads

Introduce Harness Cloud Cost Management to empower engineering and DevOps teams in proactively managing and optimizing cloud costs. See how this move allows finance teams to âshift downâ the responsibility of cloud costs to the people who consume cloud resources.

See how lack of cloud cost visibility challenged software company Relativity, and how they achieved six-figure cost savings using Harness Cloud Cost Management in just 30 days.

What are Container Costs?

Similar to physical machines and VMs, containers have infrastructure costs associated with them. Storage, compute, memory, and networking are all dimensions of costs for containers. Density is key for containers, and optimizing the dimensions allows for higher container density.

How Harness reduces Container Costs.

Harness Cloud Cost Management is a dedicated product focusing on reducing cloud and container costs. Harness Cloud Cost Management can observe trends of containerized workloads, making right-sizing recommendations based on actual usage data. Harness Cloud Cost Management can also correlate usage spikes to deployments/changes, thus getting a clearer picture of how changes impact cost.

Essential Reads

Moving to containers in Kubernetes by itself won't always lower your cloud costs. Read our Cost Management Strategies for Kubernetes eBook and find a plethora of strategies to optimize, rightsize, and downsize your way to a smaller cloud bill.

What are Cost Budgets and Cost Alerting?

Money and cloud infrastructure are not infinite. Organizations at some level must budget for cloud spend. Cost budgets allows for individual teams or departments to take ownership of their actual cloud costs and be alerted if they are exceeding or predicted to exceed the budget. When costs spike in an application, cost alerting can alert the necessary parties much closer to the spike, rather than waiting up to a month for a cloud vendor to provide a bill.

How Harness implements Cost Budgeting and Alerting.

Harness Cloud Cost Management has the ability to alert on cloud costs that are exceeding or forecasted to exceed a particular amount. With the budgeting feature in Cloud Cost Management, you are able to set actual budgets and compare against actual or forecasted costs.

Essential Reads

Cloud storage is a major contributor to cloud costs. This blog explains the basics of cloud storage service and costs and why you should focus on usage optimization.

Harness Cloud Cost Management enhances its product by introducing recommendations on the tuning of container resources, thus showing potential cost savings. Manage cloud spend explicitly and learn how to avoid rocketing consumption.

What is Cloud Cost Optimization?

Cloud Cost Optimization is the process of reducing the overall cloud cost spend by right-sizing resources, eliminating waste, and identifying mismanaged resources.

How Harness provides Cloud Cost Optimizations.

Harnessâs Cloud Cost Management is purpose-built to find and eliminate cloud cost overages. Acting as a watchful eye on your cloud infrastructure, coupled with the deployment capabilities of the Harness Platform, Cloud Cost Management can recommend and help implement savings immediately.

Essential Reads

Introduce Harness Cloud Cost Management to empower engineering and DevOps teams in proactively managing and optimizing cloud costs. See how this move allows finance teams to shift down the responsibility of cloud costs to the people who consume cloud resources.

Source URL: <https://www.harness.io/learn/use-cases/build-and-test-automation>

Build and Test Automation

The main practice in a CI pipeline is to automate the build process. Within a build process, you want to ensure you are integrating and performing unit tests so that builds fail for code that does not meet functional requirements. Build and test automation is about integrating changes early and often. This prevents maintenance hell on both feature and main branches as developers progress on feature development.

What are CI Platforms?

Continuous Integration Platforms are build automation platforms. With the rise of platforms such as Jenkins and Drone, automating language-specific build tools, e.g. Maven, Gradle, NPM, and package formats such as Docker, RPM, AMI, are crucial to any software development organization.

How does Harness leverage CI Platforms?

Harness CI is a best-of-breed Continuous Integration Tool designed to scale with today's modern cloud native needs. The Harness Platform can interact with a multitude of CI platforms and can natively call Jenkins jobs.

Essential Reads

What are the top Continuous Integration tools, and how do they help teams build and test automation? Learn more, and as a bonus, see how Harness automates the build and test process.

Learn how to install, build, and publish your first Harness Continuous Integration (CI) using Drone—all in less than 10 minutes!

What are Source Code Management and Artifact Management?

Harness Continuous Integration and the Harness Platform integrate with multiple SCM providers. Harness CI takes source code and turns it into a build/release. The Harness Platform and Harness CI leverage configuration-as-code, which integrates with SCM providers. Harness Continuous Integration and the Harness Platform also integrate with multiple artifact management providers. Harness CI products artifacts that are published to artifact management providers. The Harness Platform can call a myriad of artifact management providers and formats in a single Harness Pipeline orchestrating the needed artifacts for a successful deployment.

How Source Code Management and Artifact Management are used with Harness.

Harness Continuous Integration and the Harness Platform integrate with multiple SCM providers. Harness CI takes source code and turns it into a build/release. The Harness Platform and Harness CI leverage configuration-as-code, which integrates with SCM providers. Harness Continuous Integration and the Harness Platform also integrate with multiple artifact management providers. Harness CI products artifacts that are published to artifact management providers. The Harness Platform can call a myriad of artifact management providers and formats in a single Harness Pipeline orchestrating the needed artifacts for a successful deployment.

Essential Reads

CI/CD Evangelist Ravi Lachhman presents an overview of everything that happens before the software deployment, as well as a summary of Source Code Management.

Learn the benefits of Source Code Management, also known as Version Control, for developers as they work on different parts of a codebase, collaborate, and deliver new releases.

What is Test Automation?

Testing and quality engineering are important parts of the SDLC. Testing tools which are integrated into source code as part of the build such as JUnit or tests can be externally called against the application, e.g. performance testing tools like Gatling/JMeter. Testing can paint a wide brush and there is lots of coverage between the source code and running application.

How Harness uses Test Automation

Harness Continuous Integration enables automated build and test. With a robust ecosystem and wide variety of plug-ins, integrating modern testing methodologies and new languages are easy. Pipelines e.g. automated build and tests can be configured as code and are declarative in nature expressing goals instead of lengthy scripting.

Essential Reads

In an organization where developers are continuously pushing code to production, managing risks can be difficult. What role do developers play in Governance, risk management, and compliance (GRC)?

Testing in your Continuous Integration pipeline is crucial to modern software development teams. Understand process vs. off process testing, moving tests into the pipeline, and Harness CI.

Source URL: <https://www.harness.io/learn/use-cases/application-modernization>

Application Modernization

Since the beginning with computer software, the evolutionary journey software takes is typically a modernization journey. Re-platforming, re-factoring, re-architecting, and even re-hosting (from Amazonâs 6 Râs) are all part of Application Modernization. Migrating to Microservices is a popular Application Modernization to take on today.

What does it mean to migrate a monolith to a microservice?

The journey of building smaller deployable units has been going on for a while. Going from a monolith to microservices is a journey that, in the end, yields more deployable units. An entire application or platform can be decomposed into smaller functional areas which can independently scale and be deployed.

How Harness supports the migration from monolith to microservice.

As applications are decomposed into smaller pieces e.g .Microservices, deployment complexity increases. The more pieces you have, the more you have to deploy. Harness, with its self-service and convention-based deployments, makes scaling deployments easy. Harness erases complexity around deployment strategy, verification and rollbacks.

Essential Reads

Learn how iHerb, global eCommerce leader for natural healthcare products, migrated to the cloud with Kubernetes and Harness and reduced deployment time to <20 minutes, thus eliminating babysitting of deployments.

SaaS provider LogMeIn found its legacy tool deployments became too complex to manage and standardized its Continuous Delivery with Harness. Since migrating to Harness, theyâve reduced deployment time, toil, and effort by over 95%.

What does support for traditional apps mean?

Traditional applications are those not running in a Linux container e.g Docker, Mesos, etc. With the rise of cloud native technologies, many applications are still traditional. As such, supporting only the latest container orchestrators like Kubernetes leaves traditional apps behind.

How Harness provides support for traditional apps.

Harness is ubiquitous where you deploy to. Harness can deploy to almost any infrastructure, ranging from physical servers to serverless providers. Harness has the ability to even orchestrate Application Server deployments such as JBoss and Tomcat.

Essential Reads

SaaS company Relativity selects Harness over Spinnaker for Continuous Delivery. See how Harness was able to support their native Windows services in addition to Azure microservices.

Harness supports every major cloud provider and supports traditional on-premises clouds/DCs. Learn more about our Continuous Delivery model and how customers use Harness to onboard new Kubernetes Microservices.

What are Containers?

Containers, or Linux Containers, are popular packaging and virtualization methods for running multiple isolated systems on a host system. Since containers package all needed system and application dependencies, agility can be achieved by shipping exactly what you need.

How Containers work with Harness.

Harness supports multiple methods of orchestrating containers such as Kubernetes and Amazon ECS. Harness can interact with a variety of container registries from public cloud providers such as ECR/GCR and private registries on your own infrastructure.

Essential Reads

Kicking off our first part of a six-part series, we dive into what a container is and how container technology plays a crucial role in the Kubernetes sphere.

What is Container Orchestration?

Containers are made to die and the purpose of container orchestrators are to maintain a schedule (minimums and maximums) of running containers and resource management by placing containers on the most appropriate resources/nodes.

How Harness works with Container Orchestration.

Harness can interact with multiple container orchestrators. No matter if deploying to Kubernetes or Amazon ECS, Harness has the ability to be ubiquitous when interacting with a container orchestration. Harness out of the box provides scaffold deployments to several popular orchestrators.

Essential Reads

CI/CD SaaS tools primarily focus on Kubernetes, but have forgotten about ECS customers along the way. Harness understands that many customers are happy with their ECS environments and don't have plans to move to Kubernetes. See how the Harness ECS Experience empowers developers to automate their deployments without having to manage deployment scripts and various failure conditions during deployment.

Learn about the modern-day container orchestrator and why it's become so important to your Kubernetes journey. Also see how Harness provides a powerful orchestration layer working with your Kubernetes investment.

Source URL: <https://www.harness.io/security>

Security and Compliance at Harness

At Harness, the security and privacy of customer data, intellectual property, and personal data are top priorities. We maintain a geographically diverse security team dedicated to operating and continuously improving our security and compliance programs.

Get in Touch

If you believe you have discovered a critical security bug or vulnerability, please contact security@harness.io. We'll get back to you within 24 hours or sooner.

If you'd like to participate in our private bug bounty, please reach out with your preferred email address. We request that you do not publicly disclose the issue until we have had a chance to address it.

Compliance and Certifications

Harness takes security seriously, and has implemented a comprehensive security program to protect customer data. Each year, we undergo third-party audits and technical assessments of our security capabilities.

To request a copy of our latest reports as well as access additional information related to our security posture and controls, please visit the Harness Trust Center at trust.harness.io.

Data Privacy

Harness has intentionally minimized the amount of personal data needed to use our platform. In some circumstances, we may require personal data to facilitate your use of the platform, or to improve our websites and services.

Harness is compliant with GDPR, CCPA, and applicable privacy laws.

To understand your privacy rights and how we handle your personal data please review our Privacy Statement.

To manage the use of your privacy information please see Do Not Sell or Share My Personal Information.

To exercise your privacy rights please submit a request at Exercise Your Data Rights.

Security at Harness

How do we protect Harness?

Risk Management

Harness conducts risk assessments on at least an annual basis, and on-demand for significant changes to the environment. The output of the risk assessment is a report identifying and classifying risks, which are reviewed with management and stakeholders and tracked in a risk register. As a complement to the risk assessment process, Harness also conducts annual application business impact assessments

to validate controls and security posture of critical systems.

Vendor Management

Harness maintains a vendor risk management program that includes regular monitoring and assessment of suppliers' ability to comply with security and compliance requirements. The scope of this program includes both business systems and technical assets used for service delivery.

Threat Models

Harness conducts risk-based threat modeling for critical application features and components, including new features and modules.

Account Protection

All Harness employees use Single Sign On for access to critical business systems, and we've adopted two-factor authentication across our estate wherever possible.

Training and Awareness

When new employees start, one of their first tasks is to attend security and privacy awareness training. We also conduct annual and ongoing security and privacy awareness training for all employees.

Vulnerability Management

We use industry leading SAST, DAST, and SCA tools to discover vulnerabilities in our codebase and images. Findings are handled according to our documented Vulnerability Management policy and procedures.

Penetration Tests

We conduct internal technical security assessments on a regular basis, and track all findings through our vulnerability management process. We also engage with trusted third parties to complete network and application penetration tests at least annually.

Audit Logging

We have audit logs enabled in our environment to identify anomalies, measure efficiency, and demonstrate compliance.

Incident Response

We maintain a dedicated Incident Response function, and keep customers updated on operational incidents through our Status Page.

How do we protect customers?

How does Harness ensure Delegate updates do not contain malicious code?

The Harness Delegate is a service running in your local network or VPC to connect all of your artifact, infrastructure, collaboration, verification and other providers with the Harness Manager.

Harness follows a documented secure SDLC for all development (SaaS and on-prem) to ensure the integrity of software updates distributed to customers. The SDLC includes steps to conduct PR reviews, Static Code Analysis Testing, Dynamic Application Security Testing, regular penetration testing, and risk-based threat modeling for critical components.

Container Hardening

Harness Security reviews each image released to production, and provides a "safe image" for the given deployment. This safe image ensures that third-party dependencies have been procured from trusted resources, and that relevant operating system hardening has been implemented.

How does Harness assess third party dependencies prior to inclusion in our software?

Harness has implemented layered technical controls, including an automated scan integrated into our CI/CD pipeline to enumerate third party security vulnerabilities, manual code review, and hardened production images.

Where are secrets stored?

Harness production secrets are stored using dedicated secret management technologies. Customers can use the built-in Harness Secrets Manager, or integrate an existing third-party solution.

Product Security Features

Encryption in Transit

Data submitted to Harness is encrypted with TLS 1.2 or better over the public internet.

Encryption at Rest

Data stored in Harness SaaS environment is encrypted at rest with AES 256.

Authentication & Authorization

Harness supports both local authentication and integration with your corporate Identity Provider. See our technical documentation for a detailed walkthrough on how to configure SSO. You can enforce Two-Factor Authentication through Harness or your Identity Provider.

Backups

We perform regular backups of our systems and data stored in the Harness platform. Data is encrypted at rest, and access to data stores is restricted by the principle of least privilege.

Business Continuity and Disaster Recovery

Harness maintains a documented BCDR program, which is tested at least annually. Our RPO is 6 hours, and RTO is 4 hours.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/company/about-us>

Get to know Harness

Harness aims to enable every software engineering team in the world to deliver code reliably, efficiently and quickly to their users.

Our Mission

At the core of our mission lies a deep commitment to empowering software engineering teams worldwide. We are dedicated to providing the tools, knowledge, and resources necessary for these teams to excel in their work. Our ultimate goal is to revolutionize the way software is developed, ensuring that every team, regardless of their size or location, can deliver code with utmost reliability, efficiency, and speed.

Harness's mission is to enable the 30 million software developers in the world to deliver code to their users quickly, reliably and efficiently.

Meet our Executive Team

Jyoti Bansal is a serial entrepreneur and technology visionary who believes passionately in software's ability to change the world for the better. He co-founded Harness in 2017 to automate and simplify all software delivery processes, and serves as CEO. In 2018, he co-founded Traceable, the leading API security platform, and venture capital firm Unusual Ventures. Unusual Ventures is reinventing the VC engagement model by providing entrepreneurs with an unprecedented level of services. Unusual closed its third fund in 2022 and currently has over \$1B under management.

Carlos Delatorre is the Chief Revenue Officer (CRO) at Harness, overseeing the sales and go-to-market (GTM) functions. With a proven track record of enterprise sales leadership, he has consistently built high-performance teams that deliver strong results. Prior to Harness, Carlos was head of Sales at several companies, including Navan and MongoDB.

Currently, Carlos is a board member at Yalo and an investor/advisor to companies including Modern Treasury, Vartana, Exafunction, Starburst, and Outreach. He earned his undergraduate degrees from the University of Miami and Troy State University, and holds an MBA from Troy State University.

John Bonney oversees the Harness company's finance and accounting function. Prior to Harness, he oversaw rapid growth as the CFO at Mapbox as well as FinancialForce, the latter quadrupling revenues to over \$100M during his tenure. Prior to that, he was the Division CFO of the Cloud Business unit at SAP where he oversaw over \$1 billion of Cloud revenues across multiple business units, products, and geographies. Bonney joined SAP from Ariba as a senior finance executive who helped drive rapid expansion and ultimately the sale of Ariba to SAP for \$4.3 billion.

Previously, Luan has served as an advisor for companies like PlanGrid, Headhuntr.io, and 88 Inc. He also served as a Guest Lecturer at University of California, Berkeley for the Organization and Management course with the International Diploma Programs. Luan holds

a degree in International Relations from the University of California, Davis. He has written articles on the âWar for Talentâ; and has been featured in several publications including Forbes, The Wall Street Journal, The Guardian, and San Francisco Business Times, specifically on the art and science behind recruiting.

â

Sri is responsible for leading the companyâs global engineering organization, including its R&D teams in United States, Bangalore (India), Latin America, and Europe. Sri previously spent five years scaling Zoom from less than 12 million ARR in the early days to post IPO growth, ultimately leading a team of more than 600 people, launching the companyâs marketplace, and significantly expanding its ecosystem. Sri has also held key engineering leadership roles at Saba, Plantronics, and Cisco.

Gleb Brichko brings over 20 years of B2B marketing experience across software, security, and infrastructure. Most recently, Gleb built and led the global growth and demand marketing team at Rubrik, where he was responsible for leading, planning and executing the company's go-to-market strategy, brand presence, demand generation campaigns and programs, digital and web, partner marketing, field marketing, customer advocacy, strategic events, content, and operations. His prior roles include marketing leadership positions at Nutanix, Fortinet, ForeScout, and Blue Coat Systems. Gleb holds a bachelor's degree in mass communications from UC Berkeley.

Parmeet Chaddha leads post-sales customer success and services function at Harness. With multiple patents to his name, Parmeet brings 30+ years of general management experience in engineering, business development and customer success. Prior to Harness, he led customer success at Securiti.ai. Previously, he held senior executive roles at Google Cloud, Nutanix, EMC, IBM and Oracle. He has been honored as "Top 50 Technology Executives" by InfoWorld and the "Leading Data Consultants for North America, 2022" by the CDO Magazine. Parmeet holds a B.S. and M.S. from Massachusetts Institute of Technology.

Investors

Harness was spun out of BIG Labs, a startup studio designed to solve hard technology problems and build enduring companies. Harness has raised \$425M of venture capital from top-tier investors.

Our offices around the globe

[Press & news](#)

[Partners](#)

[Contact us](#)

Source URL: <https://www.harness.io/learn/use-cases/jenkins>

Scaling Beyond Jenkins

Today, software delivery solutions need to quickly and easily scale alongside organizations and teams advancing their software delivery capabilities. Not every architecture solution is designed to scale with you. Consider what it means to support your organization as application workloads scale horizontally and vertically. Use Harness to quickly deploy artifacts into production and into the hands of your users. Harness integrates with Jenkins to manage your CD process.

What are the challenges posed by Jenkins?

Use cases that surround Continuous Delivery are challenging for a platform like Jenkins, which has been designed for Continuous Integration. Items that are post-build, e.g. deployment, verification, and additional operational steps such as rollbacks, are challenging in Jenkins and need to be heavily scripted out.

Essential Reads

We compare Jenkins and Harness in detail and highlight the pains of scripting in Jenkins. See how you can migrate easily from Jenkins to Harness in minutes to accelerate deployments and solve your CI/CD problems.

In this blog, we map out four reasons why your Jenkins pipeline may be brittle. Learn how plug-ins, jobs/scripts, debugging in Jenkins, and other issues are affecting your CI/CD performance.

What is the evolution of Jenkins?

Jenkins genesis and lineage started as a build server for JAVA projects, then evolved into a build server for multiple languages. The introduction of their pipeline domain specific language [DSL] allowed for scripted steps and manipulations to broaden the capabilities into automation.

How does Harness Self-Service compare to Jenkins?

Harness is purpose-built to enable software delivery with self-service in mind. Challenging items around Continuous Delivery such as verification, canary deployments, and auditing are all automated and convention-based. The Harness Platform has cloud cost efficiency and Continuous Integration portions with the same ease of use and capabilities.

Essentials Reads

Casting Networks, a cutting-edge entertainment technology company, hit several roadblocks while migrating their legacy monolith applications to Kubernetes microservices. Learn how they quickly onboarded their microservices with Harness.

How Harness supports Customer Migrations from Jenkins.

Harness has the ability to natively call Jenkins jobs. As a wrapper around Jenkins jobs, customers (if needed) can start to decompose lengthy deployment scripts in Jenkins and just have Jenkins focus on build-centric steps. By leveraging Harness CI, Jenkins can potentially be removed altogether.

Essentials Reads

Learn how SaaS company Meltwater, after switching to Harness for Continuous Integration, eliminated the problems it faced while using Jenkins. Read about how they built 1200 artifacts a day using Harness CI (Drone).

In this case study, we see how Marketron tripled its deployment frequency and decreased deployment time 75% using Harness. Prior, Marketron had deployed its legacy applications using Jenkins and manual executions.

Source URL: [https://www.harness.io/company/contact-sales?
utm_source=harness_io&utm_medium=cta&utm_campaign=homepage&utm_content=hero](https://www.harness.io/company/contact-sales?utm_source=harness_io&utm_medium=cta&utm_campaign=homepage&utm_content=hero)

The Software Delivery Platform Business Service Teams Prefer

Want to see a custom demo or get help finding the right plan? We'd love to chat. Lean into AI/ML with Harness - the simple, scalable CI/CD platform.

- Fully Integrated Modules
- Test Intelligence
- Enterprise-Grade Security
- Fine-Grained RBACÂ Security
- Complete Flexibility

- AI/ML DrivenÂ Workflows
- Smart Feature Rollouts
- Multi-Cloud Cost Management
- Comprehensive Auditability

Contact Sales

By signing up, you agree to our Privacy Policy and our Terms of Use.

Nick Willson, TechOps Lead, GoSpotCheck

Take Harness for a spin

Source URL: <https://www.harness.io/learn/use-cases/gitops>

GitOps

GitOps centers itself around a version control system such as git to control the software delivery lifecycle. Leverage a git repo as a source of truth for changes and easily auto-sync changes to Kubernetes clusters. You can have confidence that changes managed through git are active within a Kubernetes environment.

What are Pipelines as Code?

Pipelines as Code are programmatic ways of defining and executing software delivery. The build, test, and deployment of applications should be treated as code.

How Harness uses Pipelines as Code.

The Harness Platform is backed by configuration-as-code. All aspects of the Harness Platform can be saved as configuration-as-code so your pipelines are accessible as code. The Harness Platform supports a bi-directional Git Sync, so changes made in a Source Code Management [SCM] solution or vice versa in the Harness Platform are represented as code.

Essential Reads

Ruckus Networks leveraged Harness GitOps and Pipeline as Code, allowing developers to manage pipeline, service, environment, Helm, and Kubernetes configuration all in one place as code. All pipelines can be version-controlled and managed like code in any Git repository.

GitOps has picked up momentum recently and has been garnering participation in the CNCF. Learn how to leverage GitOps to supercharge your CI/CD pipelines using a Pipelines as Code approach.

What is Infrastructure as Code?

Infrastructure as Code or IaC is the administration and management of infrastructure through code vs physical configurations. Examples of IaC are Chef, Puppet, Terraform, and CloudFormation.

How Infrastructure as Code relates to Harness.

Harness has the ability to interact and manage with several infrastructure-as-code providers. As part of a modern deployment, infrastructure is provisioned at deployment time. Harness can orchestrate popular infrastructure-as-code providers such as Terraform and CloudFormation.

Source URL: <https://www.harness.io/company/partners>

Partner Program

Grow your business and help your customers thrive with Harness.

We collaborate with our partner ecosystem to enable our mutual customers to deliver uncompromising, world-class software. Through the Harness Partner Program, partners enjoy program benefits including incentives, deal registration, access to training and enablement, marketing support and much more.

Harness Partners With

With Harness, you will grow more pipeline and win more deals. Our partners are able to provide better pricing and faster delivery, while enjoying higher gross margins on the same work as offerings from other competitors.

Manage your customers' cloud spend with Harness Cloud Cost Management, decrease downtime in their customers' SRE practice with Harness SRM & Chaos Engineering, provide unified reporting and governance and much more.

The Harness platform has over 300 native DevOps integrations. As our platform grows, so does the opportunity for more robust partnerships with other technology partners. If there is a customer benefit and go-to-market opportunity, we'd love to work with you.

Why Partner With Harness?

Partnering with Harness provides you with a versatile library of solution offerings and capabilities

Increase your book of business by continuing to help your customers conquer their biggest challenges

Confidently deliver innovative user experiences while providing world-class customer satisfaction.

Press & news

Partners

Contact us

Source URL: https://www.harness.io/company/contact-sales?utm_source=harness_io

The Software Delivery Platform Business Service Teams Prefer

Want to see a custom demo or get help finding the right plan? We'd love to chat. Lean into AI/ML with Harness - the simple, scalable CI/CD platform.

- Fully Integrated Modules
 - Test Intelligence
 - Enterprise-Grade Security
 - Fine-Grained RBACÂ Security
 - Complete Flexibility
-
- AI/ML DrivenÂ Workflows
 - Smart Feature Rollouts
 - Multi-Cloud Cost Management
 - Comprehensive Auditability

Contact Sales

By signing up, you agree to our Privacy Policy and our Terms of Use.

Nick Willson, TechOps Lead, GoSpotCheck

Take Harness for a spin

Source URL: <https://www.harness.io/company/contact-sales>

The Software Delivery Platform Business Service Teams Prefer

Want to see a custom demo or get help finding the right plan? We'd love to chat. Lean into AI/ML withÂ Harness - the simple, scalable CI/CD platform.

- Fully Integrated Modules
 - Test Intelligence
 - Enterprise-Grade Security
 - Fine-Grained RBACÂ Security
 - Complete Flexibility
-
- AI/ML DrivenÂ Workflows
 - Smart Feature Rollouts
 - Multi-Cloud Cost Management
 - Comprehensive Auditability

Contact Sales

By signing up, you agree to our Privacy Policy and our Terms of Use.

Nick Willson, TechOps Lead, GoSpotCheck

Take Harness for a spin

Source URL: https://www.harness.io/company/contact-sales?utm_source=harness_io&utm_medium=cta&utm_campaign=platform&utm_content=main_nav

The Software Delivery Platform Business Service Teams Prefer

Want to see a custom demo or get help finding the right plan? We'd love to chat. Lean into AI/ML withÂ Harness - the simple, scalable CI/CD platform.

- Fully Integrated Modules
 - Test Intelligence
 - Enterprise-Grade Security
 - Fine-Grained RBACÂ Security
 - Complete Flexibility
-
- AI/ML DrivenÂ Workflows
 - Smart Feature Rollouts
 - Multi-Cloud Cost Management
 - Comprehensive Auditability

Contact Sales

By signing up, you agree to our Privacy Policy and our Terms of Use.

Nick Willson, TechOps Lead, GoSpotCheck

Take Harness for a spin

Source URL: <https://www.harness.io/case-studies/united-airlines-accelerates-deployments-harness>

United Airlines Accelerates Deployments by 75% With Harness

As part of its digital transformation efforts, United moved from monolithic applications to microservices. Legacy tools couldn't cope with this increase in scalability.

- 75% faster deployment time â from 22 minutes to 5 minutes
- Increased compliance using templates and automation

United Airlines operates the most comprehensive global route network among North American carriers. United is bringing back its customers' favorite destinations and adding new ones on its way to becoming the world's best airline. In 2022, United kicked off the launch of its largest transatlantic expansion in its history, and it is making its fleet bigger and better than ever by adding up to 200 widebody planes â the largest order by a U.S. carrier in commercial airline history. Its 80,000-plus employees are on a mission to do good in the air and on the ground, working to make the world a happier, greener, more inclusive, and more fascinating place.

Challenges: Migrate to the Cloud, Deploy Faster

As part of the United Next strategy, United aims to move 80% of its workload to the cloud on Amazon Web Services (AWS). When looking at its internal development process, the company quickly saw the need for more innovation that would enhance its current tools, skills, people, and culture.

âOur legacy tools required a lot of manual configuration, and we had long wait times because they took too much time to commit the code, create an image, and deploy it to an environment,â said Raji Koppala, Senior Manager of DevOps at United Airlines. âWe had some automation, but it couldn't keep up with process, security, and compliance changes. Plus, the developers who created and deployed the builds weren't familiar with the organizational security and compliance standards or the resulting cost to the company of manual change management and slow deployments, so we started to seek out new solutions that could streamline the deployments.â

As part of its digital transformation efforts and migration to the cloud, United moved from monolithic applications to microservices, turning a single monolith deployment process into hundreds of independently deployable microservices. This required a massive increase in scalability, and legacy tools couldn't cope. To adapt, United's DevOps team started listing deployment requirements, detailing gaps, and creating an inventory of existing tools used across its software delivery process.

âWe wanted to concentrate more on innovation and moving things to the cloud,â said Ratna Devarapalli, Director of IT - Architecture, Platform Engineering & DevOps at United Airlines. âWe also wanted to âshift leftâ and put governance in the hands of developers during the build process.â

Solution: A Platform to Automate and Self-Serve Deployments

With United working to shift security and testing earlier in the development process, it also needed guardrails to make sure that processes were followed, and governance controls were in place. After evaluating multiple continuous integration and continuous delivery (CI/CD) tools, United chose Harness.

âNone of the other tools we evaluated could give us the âshift leftâ approach that we needed,â said Devarapalli. âBy choosing Harness for CI and CD, we were able to give the governance policies to the developers and create the guardrails we needed. Harness gives us a platform rather than just a DevOps tool.â

United implemented Harness CI/CD to enable developers to execute more self-service deployments. Simply by creating a help ticket, Harness kicks off automation to manage the delivery process with minimal manual intervention required. This accelerates deployments, as well as enables United to scale its delivery process by running deployment pipelines in parallel.

âWe were pushing our teams to migrate to the cloud, but that has a chain of dependencies, and developers had to move much faster,â Koppala said. âHarness gave us a lot of best practices right out of the box with reusable templates so developers don't need to perform any guesswork to determine how the best pipeline should be built.â This simplified developer experience and faster time-to-value with Harness help maximize developer productivity and efficiency at United.

United developers now use Harness CI to simply select a template and start building a new pipeline or to generate pipelines dynamically for each new service. Harness automates reporting to remove manual effort from the developers' workloads and free up time to focus on more value-add tasks, such as working on new feature developments related to the United Next initiative.

âWe started using Harness CI and CD to automatically generate pipelines, use pre-built pipelines, and optimize our deployment process,â Koppala added.

Results: 75% Faster Deployment Times

With Harness, United has automated its delivery and deployment processes to enable developers with a self-service approach for deployment, increase compliance with governance policies and processes, and dramatically accelerate its software deployment cycles. Harness is now a critical component in helping DevOps keep the company moving toward its United Next goals.

âThe most exciting feature of Harness is on-demand automation,â said Koppala. âOur delivery processes are now more dynamic, more developer friendly, and Harness has eliminated most of the manual coordination we used to do. Itâs really enabled us to scale all parts of our delivery process.â

Deployment speed has been enhanced as well. Prior to Harness, a CI pipeline build for an airport operation application took nearly 22 minutes. Now, itâs done in under five minutes for the exact same code. Moreover, teams can build microservices in parallel without unnecessary testing of the services not affected by the new code. Harness also enables developers to orchestrate deployments and sequentially deliver builds as they like.

âWe have seen 75% efficiency gains with Harness,â said Devarapalli. âRunning pipelines in parallel gives us scalability, too. We can generate ten pipelines in 50 seconds. That used to take days or even weeks if we had a junior DevOps engineer working on it. Harness has been instrumental in helping us work toward our goal of migrating 80% of our workloads to the cloud, increasing operation efficiency that matches United Next objectives, and reducing on-premises costs. Weâre also giving time back to developers because they no longer have to babysit deployments. They can just go to the Harness dashboard to view the status of every deployment.â

United also uses Harness to make sure that deployments are compliant with security and governance policies. It lets developers work faster and deliver more functionality without the risk or friction of manual compliance processes.

âThereâs usually a lot of conflict between automation and compliance, especially as policies are updated,â added Koppala. âHarness gives us templated governance to enforce organizational standards without developers even realizing a policy has changed. The impact on developers is almost zero.â

The combination of speed, compliance, and scale has given Unitedâs DevOps team the time to focus on optimizing processes and working toward their digital transformation efforts. âWith Harness, itâs the democratization, the governance, the speed â itâs all been a huge gain for United,â concluded Devarapalli. âWeâre way more efficient than we were a couple years back. The Harness platform is allowing us to do our part as the DevOps team, so United can reach our United Next goals.â

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/citi-improves-software-delivery-performance-reduces-toil-with-harness-cd>

Citi Improves Software Delivery Performance, Reduces Toil

With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

About

Citi is a preeminent banking partner for institutions with cross-border needs, a global leader in wealth management and a valued personal bank in its home market of the United States. Citi does business in nearly 160 countries and jurisdictions, providing corporations, governments, investors, institutions and individuals with a broad range of financial products and services.

Challenge: Implement Continuous Delivery at Enterprise Scale While Maintaining a Focus on Risk and Controls

Citi's next-generation software delivery platform provides an integrated experience across all stages of software delivery with a user base of over 20,000 people across the bank's software engineering, site reliability engineering (SRE), product, and risk and control functions. The platform brings together all the tools and services involved in software delivery with the aim of improving performance, consistency, and maintenance across the enterprise. And all this while operating in the unique regulatory and risk management environment that comes with being one of the biggest banks in the world.

As Stefanos Piperoglou, Technical Program Manager for the platform, explained, the deployment solution in place at Citi was not meeting their requirements:

- It did not provide continuous delivery / continuous deployment (CD) capabilities
- It required manual configuration and a centralized operations team
- It was not cloud-native and did not play well with modern microservice architectures
- It was difficult to provide a clear audit trail of the hundreds of thousands of production releases that are executed annually

After evaluating several other solutions, Citi landed on Harness Continuous Delivery, which met all of its current needs and enabled its ambitions to host the majority of its software in the public cloud.

Solution: Enter Harness CD

With the adoption of Harness Continuous Delivery, the platform team was able to automate the continuous integration and continuous delivery (CI/CD) process end to end, helping deliver on its goals and more.

Move Fast, Deploy Often, Roll Back Quickly

Improving Citi's core DORA outcomes – Deployment Frequency, Lead Time for Changes, Change Failure Rate and Time to Restore Service – is at the core of the platform's mission. Harness CD allows Citi to construct CD pipelines that go through all necessary deployment, testing, audit logging, and change management stages after every code change is reviewed and accepted. It allows ops teams to quickly roll back any change that causes an issue in production. And Harness CD's integration with Citi's other observability tools allows CD pipelines to alert ops teams or even automatically roll back changes if they cause any unexpected behaviour in production systems.

“Harness CD lets us release each change within minutes of a pull request being merged. Running all the necessary tests and scans during the pipeline gives us the confidence to do this. Using Harness's implementations of deployment strategies such as Canary and Blue/Green allows us to verify deployments while minimizing the blast radius of any problems, and the UX allows us to quickly roll back any change with minimal fuss as soon as we detect a problem,” Piperoglou said.

Flexibility with Control

Development teams at Citi operate in an environment of much higher scrutiny and control than is typical in many other industries. The high standards set by financial services regulators in the various jurisdictions in which Citi operates, and the unique security posture required by such a high-profile target, mean that delivering software to production involves an extensive collection of controls. Application teams are frequently subject to internal and external audits, security threat modelling exercises, and penetration tests. It is critical to keep all of this process in place, but equally important not to affect developer productivity or slow down deliveries.

“Harness CD's template system and OPA framework lets us give application teams the flexibility to design CD pipelines in a way that fits the requirements of the businesses they serve while being sure they are following all required and recommended practices. It lets users enforce policy and promote standardisation without restricting what engineers can achieve,” said Michael Freeman, lead software engineer for the platform.

Cloud-Native, but not Cloud-Only

Citi has been adopting cloud patterns such as containers and Kubernetes on-premises for several years, and is also ramping up the migration of many of its applications to various public cloud providers. Meanwhile, many applications will continue to be hosted either on Citi's premises, or on High-Frequency Trading (HFT) platforms that need dedicated, physical infrastructure for the foreseeable future. It was critical that whatever CD solution Citi adopted would cater to both on- and off-premises use cases and allow for a smooth,

gradual migration to the cloud while maintaining consistency for users on their journey.

Result: Improved Software Delivery Performance, Reduced Toil

With Citiâs previous deployment solution, teams were releasing code with a cadence that usually ranged from once per week to once every several months. *The developer was the pipeline. They'd sit in front of a screen and click to start deployment for each environment promotion, while manually entering change management details and attesting that all necessary tests and controls had been completed. That process was pure toil and took hours, sometimes days,*â said Piperoglou. *Harness CD allowed us to automate all of this and go from build to running in production in under 7 minutes, with most of our users performing production deployments several times a day.*â

Moving to the Cloud with Confidence

Operating in a highly regulated environment and with a backdrop of increasingly frequent and sophisticated attacks on the software supply chain, Citiâs top-speed innovation is enabled by Harnessâs industry experience. *Working with Harnessâs solutions team has helped us validate our ideas and understand what our peers in financial services as well as other industries are doing in this space. Building on top of a widely adopted platform that has been proven successful gives you an extra layer of confidence and provides valuable input into the design process,*â said Freeman.

No Looking Back

With the Harness Continuous Delivery solution, Citi has given its application developers and operations teams control, ease of use, and unmatched speed for its deployment pipeline while eliminating toil, reducing risk, and increasing security. Above all, it has allowed Citi to deliver new and innovative solutions that help its clients achieve their goals and prosper today and in the future.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

Advanced Achieves Cloud Cost Governance Excellence, Saves 33% on Cloud Costs with Harness CCM

Advanced was struggling with cost overruns, and lack of adoption of the previous cost management tool. After implementing Harness Cloud Cost Management, Advanced has seen 33% annualized cost savings to-date.

The Modern Software Delivery Platform^{Â®}

Need more info? Contact Sales

Source URL: <https://www.harness.io/company/press-and-news>

Harness Press Room

The Latest News and Media Coverage about Harness

In the News

Jan 11, 2024

CI/CD startup Harness acquires venture-backed rival Armory

Jan 11, 2024

Harness Acquires Armory Continuous Delivery Assets

Jan 11, 2024

Harness acquires Armory

Dec 6, 2023

Harness enables every team to deliver code reliably, efficiently and quickly to their users.

Dec 5, 2023

The return of the whiteboardâand four other trends shaping the software developer experience in 2024

Dec 1, 2023

Harness: Unlocking Modern Software Delivery

Oct 1, 2023

Most Agile / Responsive SaaS Solution Of The Year

Sep 15, 2023

Not measuring your software engineering maturity? Hereâs why you should start

Sep 21, 2023

Harness debuts new code management tools for its software delivery platform

Sep 21, 2023

Harness Launches Open Source Gitness Platform

Aug 14, 2023

How Harness is using AI to safely streamline operations

Sep 22, 2023

Ensuring 'Developer Happiness': How To Hang On To Software Engineers

Press Releases

Jan 11, 2024

Harness Acquires Armory Assets

Sep 21, 2023

Harness Announces Engineering Leadership Community to Codify Engineering Excellence Standards

Sep 21, 2023

Harness Releases Gitness - Open Source Git Platform

Sep 21, 2023

Harness Introduces Four New Modules to Enhance Efficiency, Collaboration, and Security Across the Software Delivery Lifecycle

Jun 21, 2023

Harness Empowers Developers with Generative AI Assistant for the Software Development Life Cycle: Announces the Launch of AIDA

Jun 16, 2023

Harness Recognized as a Visionary in the 2023 Gartner® Magic Quadrant for DevOps Platforms Report

Jun 2, 2023

Harness Recognized With Highest Scores in CI/CD and Platform Capabilities Evaluation Among Top Integrated Software Delivery Platforms

Apr 24, 2023

Harness Accelerates Secure Application Delivery With Latest Release of Security Testing Orchestration, Improving Collaboration Between Developers and Security Practitioners to Save Time, Cost, and Effort

Apr 18, 2023

Harness Introduces New Jira Software Integrations, Enabling a More Efficient Workflow for Developers

Feb 14, 2023

Harness Becomes First Comprehensive DevSecOps & CI/CD Pipeline Solution With Integrated Security Testing Orchestration, Now Available to Government Agencies on U.S. Air Force Platform One

Jan 24, 2023

Harness® Acquires Propelo, Bringing Actionable Engineering Insights to Award-Winning Software Delivery Platform

Nov 30, 2022

Harness® Launches Cluster Orchestrator for Amazon EKS to Help Customers Save up to 90% on Cloud Costs

Awards and Recognition

CI/CD startup Harness acquires venture-backed rival Armory

Harness Acquires Armory Continuous Delivery Assets

Harness acquires Armory

Harness Acquires Armory Assets

Harness enables every team to deliver code reliably, efficiently and quickly to their users.

The Evolution of Harness with Martin Reynolds | AWS re:Invent 2023

The return of the whiteboard and four other trends shaping the software developer experience in 2024

Harness: Unlocking Modern Software Delivery

Most Agile / Responsive SaaS Solution Of The Year

Not measuring your software engineering maturity? Here's why you should start

Harness debuts new code management tools for its software delivery platform

Harness Launches Open Source Gitness Platform

How Harness is using AI to safely streamline operations

Ensuring 'Developer Happiness': How To Hang On To Software Engineers

Harness launches Gitness, an open-source GitHub competitor

Harness Announces Engineering Leadership Community to Codify Engineering Excellence Standards

Harness Releases Gitness - Open Source Git Platform

Harness Introduces Four New Modules to Enhance Efficiency, Collaboration, and Security Across the Software Delivery Lifecycle

Forbes Cloud 100

Large Bay Area Private company

Best Place to Work in the Bay Area

The 100 Coolest Cloud Computing Companies Of 2023 - Harness #10

Best End-to-End DevOps Tool/Service (DevOps Dozen Winner)

Promising Companies founded and managed by Indians in the US

CNBC's Top Startups for the Enterprise

The Logan Bartlett Show | Jyoti Bonsal: Life After Selling a \$3.7B Company

Harness Eliminates Common Developer Pain Points With a Simplified End-to-End Software Delivery Platform

Companies find their FinOps Wanting in the Struggle Against Rampant Cloudflation

The Playbook Universe - Jason Eubanks #009

Bad developer UX? Look out for breakages, burnout, and "biological single points of failure"

Are Developers Actually Using A.I. Tools in Their Workflows?

Harness Brings Generative AI Capabilities to DevOps Workflows

Harness launches AI assistant to enhance productivity across SDLC

Retail companies gain DORA metrics ROI from specialist tools

United Airlines clears CI/CD pipelines for takeoff

Harness releases generative AI assistant to help increase developer efficiency

Continuous software delivery startup Harness deploys generative AI assistant to ease developer headaches

Harness Empowers Developers with Generative AI Assistant for the Software Development Life Cycle: Announces the Launch of AIDA®

Harness Recognized as a Visionary in the 2023 Gartner® Magic Quadrant® for DevOps Platforms Report

Harness Recognized With Highest Scores in CI/CD and Platform Capabilities Evaluation Among Top Integrated Software Delivery Platforms

Chaos Engineering with Uma Mukkara

3 Ways Tech Companies Can Bring Their Cloud Costs Back to Earth

AI Has Become Integral to the Software Delivery Lifecycle

Harness Introduces Continuous Error Tracking to Provide Developer-First Observability for Modern Applications

Harness CEO Jyoti Bansal: The enterprise needs guardrails to ship code quickly

Harness Introduces Continuous Error Tracking

Observability for modern applications

Observability for modern applications

Harness announces new feature to proactively identify errors

SLOconf 2023 QA: Harness Showcases Reliability Modules (SRM and CE) as Part of Their Software Delivery Platform

3 Security Investments Takeaways From RSA 2023

Out-Siloing Security and Development to Mitigate Cyber Risk

AI Will Change the Role of Developers Forever. Here's Why That's Good News

Season 3, Ep. 27 â Improving software delivery with AI and Open Source, with Scott Sanchez, CMO, Harness

Uma Mukkara, Harness | KubeCon + CloudNativeCon Europe 2023

Software Devs Embrace AI and Acknowledge Benefits

United Airlines Deploys Software 75% Quicker Using the Harness DevOps Platform

AITech Interview with Nick Durkin, Field CTO at Harness

Rethinking 'Shifting Left': Empowering Developers and Unleashing Innovation

Google's Bard Can Write Code. What's That Mean for You?

Harness Introduces Continuous Error Tracking to Provide Developer-First Observability for Modern Applications

United Airlines Accelerates Deployments by 75% with Harness

Harness Accelerates Secure Application Delivery With Latest Release of Security Testing Orchestration, Improving Collaboration Between Developers and Security Practitioners to Save Time, Cost, and Effort

Harness Introduces New Jira Software Integrations, Enabling a More Efficient Workflow for Developers

A.I. Could Automate 800 Million Jobs: Report

What ChatGPT Means for Developers

Your Biggest Cybersecurity Threat is Something Youâve Never Heard Of

Silicon Valley Lost its Bank. Expect âZombie VCsâ and Dark Times for Startups

The Startup Within a Startup: How to Stay Ahead of the Game as you Scale

Tech Connects' Podcast: How Automation Could Impact Tech Jobs

Developing Success: Seven Pillars of DX to Invest In

How to Measure Time by Impact with Jyoti Bansal of Harness and Traceable

AHEAD and Harness: Enterprise Cloud Acceleration Partnership

Harness Lands on Air Force Platform One DevSecOps Platform

Harness Becomes First Comprehensive DevSecOps & CI/CD Pipeline Solution With Integrated Security Testing Orchestration

Harness CI/CD, Security Testing Modules Now Available on Platform One

Harness Becomes First Comprehensive DevSecOps & CI/CD Pipeline Solution With Integrated Security Testing Orchestration, Now Available to Government Agencies on U.S. Air Force Platform One

Harness CD/CI/STO solutions chosen for DoDâs Platform One

The 20 Coolest Cloud Monitoring And Management Companies Of The 2023 Cloud 100

Harness Becomes First Comprehensive DevSecOps & CI/CD Pipeline Solution With Integrated Security Testing Orchestration, Now Available to Government Agencies on U.S. Air Force Platform One

Real-time Analytics News for Week Ending January 28

ChatGPT Offers Fast Help to Coders with Questions

Tackle the Tedium: Automation for Better Business

Top WorkTech News From the Week of January 27th: Updates from Agiloft, Oracle, Tine, and More

Software Engineering M&A: Harness Acquires Propelo

5 Channel Partner & MSP Market News Updates

Three Venture Investors Weigh In on the Future Of Biotech as They Launch a New \$350M Fund

Harness Acquires Propelo

Harness Acquires Propelo to Surface Software Engineering Bottlenecks

Harness Acquires Propelo to Surface Software Engineering Bottlenecks

VC Daily: DOJ Sues Google; Why M&A Deals Fall Apart

Reduce Kubernetes spend with these 10 Kubecost alternatives

Harness Acquires Propelo

IT Budgets Under Pressure Spur Tool Consolidation

Harness® Acquires Propelo, Bringing Actionable Engineering Insights to Award-Winning Software Delivery Platform

Diversity and Inclusion: 7 Best Practices for Changing Your Culture

Developer Effectiveness and the Rise of Platform Engineering | 2023 Predictions By Harness

What is Hot for Fintech Investors in 2023?

Harness 2023 Predictions: The Year of Automation, Platform Engineering, and More

Workshop Wednesday: LIVE with Harness.io CRO

CI/CD startup Harness acquires venture-backed rival Armory

Harness Acquires Armory Continuous Delivery Assets

Harness acquires Armory

Harness Acquires Armory Assets

Harness enables every team to deliver code reliably, efficiently and quickly to their users.

The Evolution of Harness with Martin Reynolds | AWS re:Invent 2023

The return of the whiteboard and four other trends shaping the software developer experience in 2024

Harness: Unlocking Modern Software Delivery

Most Agile / Responsive SaaS Solution Of The Year

Not measuring your software engineering maturity? Here's why you should start

Harness debuts new code management tools for its software delivery platform

Harness Launches Open Source Gitness Platform

How Harness is using AI to safely streamline operations

Ensuring 'Developer Happiness': How To Hang On To Software Engineers

Harness launches Gitness, an open-source GitHub competitor

Harness Announces Engineering Leadership Community to Codify Engineering Excellence Standards

Harness Releases Gitness - Open Source Git Platform

Harness Introduces Four New Modules to Enhance Efficiency, Collaboration, and Security Across the Software Delivery Lifecycle

Forbes Cloud 100

Large Bay Area Private company

Best Place to Work in the Bay Area

The 100 Coolest Cloud Computing Companies Of 2023 - Harness #10

Best End-to-End DevOps Tool/Service (DevOps Dozen Winner)

Promising Companies founded and managed by Indians in the US

CNBC's Top Startups for the Enterprise

The Logan Bartlett Show | Jyoti Bonsal: Life After Selling a \$3.7B Company

Harness Eliminates Common Developer Pain Points With a Simplified End-to-End Software Delivery Platform

Companies find their FinOps Wanting in the Struggle Against Rampant Cloudflation

The Playbook Universe - Jason Eubanks #009

Bad developer UX? Look out for breakages, burnout, and "biological single points of failure"

Are Developers Actually Using A.I. Tools in Their Workflows?

Harness Brings Generative AI Capabilities to DevOps Workflows

Harness launches AI assistant to enhance productivity across SDLC

Retail companies gain DORA metrics ROI from specialist tools

United Airlines clears CI/CD pipelines for takeoff

Harness releases generative AI assistant to help increase developer efficiency

Continuous software delivery startup Harness deploys generative AI assistant to ease developer headaches

Harness Empowers Developers with Generative AI Assistant for the Software Development Life Cycle: Announces the Launch of AIDA®

Harness Recognized as a Visionary in the 2023 Gartner® Magic Quadrant for DevOps Platforms Report

Harness Recognized With Highest Scores in CI/CD and Platform Capabilities Evaluation Among Top Integrated Software Delivery Platforms

Chaos Engineering with Uma Mukkara

3 Ways Tech Companies Can Bring Their Cloud Costs Back to Earth

AI Has Become Integral to the Software Delivery Lifecycle

Harness Introduces Continuous Error Tracking to Provide Developer-First Observability for Modern Applications

Harness CEO Jyoti Bansal: The enterprise needs guardrails to ship code quickly

Harness Introduces Continuous Error Tracking

Observability for modern applications

Observability for modern applications

Harness announces new feature to proactively identify errors

SLOconf 2023 QA: Harness Showcases Reliability Modules (SRM and CE) as Part of Their Software Delivery Platform

3 Security Investments Takeaways From RSA 2023

Out-Siloing Security and Development to Mitigate Cyber Risk

AI Will Change the Role of Developers Forever. Here's Why That's Good News

Season 3, Ep. 27 â Improving software delivery with AI and Open Source, with Scott Sanchez, CMO, Harness

Uma Mukkara, Harness | KubeCon + CloudNativeCon Europe 2023

Software Devs Embrace AI and Acknowledge Benefits

United Airlines Deploys Software 75% Quicker Using the Harness DevOps Platform

AITech Interview with Nick Durkin, Field CTO at Harness

Rethinking 'Shifting Left': Empowering Developers and Unleashing Innovation

Google's Bard Can Write Code. What's That Mean for You?

Harness Introduces Continuous Error Tracking to Provide Developer-First Observability for Modern Applications

United Airlines Accelerates Deployments by 75% with Harness

Harness Accelerates Secure Application Delivery With Latest Release of Security Testing Orchestration, Improving Collaboration Between Developers and Security Practitioners to Save Time, Cost, and Effort

Harness Introduces New Jira Software Integrations, Enabling a More Efficient Workflow for Developers

A.I. Could Automate 800 Million Jobs: Report

What ChatGPT Means for Developers

Your Biggest Cybersecurity Threat is Something You've Never Heard Of

Silicon Valley Lost its Bank. Expect a Zombie VCs and Dark Times for Startups

The Startup Within a Startup: How to Stay Ahead of the Game as you Scale

Tech Connects' Podcast: How Automation Could Impact Tech Jobs

Developing Success: Seven Pillars of DX to Invest In

How to Measure Time by Impact with Jyoti Bansal of Harness and Traceable

AHEAD and Harness: Enterprise Cloud Acceleration Partnership

Harness Lands on Air Force Platform One DevSecOps Platform

Harness Becomes First Comprehensive DevSecOps & CI/CD Pipeline Solution With Integrated Security Testing Orchestration

Harness CI/CD, Security Testing Modules Now Available on Platform One

Harness Becomes First Comprehensive DevSecOps & CI/CD Pipeline Solution With Integrated Security Testing Orchestration, Now Available to Government Agencies on U.S. Air Force Platform One

Harness CD/CI/STO solutions chosen for DoD's Platform One

The 20 Coolest Cloud Monitoring And Management Companies Of The 2023 Cloud 100

Harness Becomes First Comprehensive DevSecOps & CI/CD Pipeline Solution With Integrated Security Testing Orchestration, Now Available to Government Agencies on U.S. Air Force Platform One

Real-time Analytics News for Week Ending January 28

ChatGPT Offers Fast Help to Coders with Questions

Tackle the Tedium: Automation for Better Business

Top WorkTech News From the Week of January 27th: Updates from Agiloft, Oracle, Tine, and More

Software Engineering M&A: Harness Acquires Propelo

5 Channel Partner & MSP Market News Updates

Three Venture Investors Weigh In on the Future Of Biotech as They Launch a New \$350M Fund

Harness Acquires Propelo

Harness Acquires Propelo to Surface Software Engineering Bottlenecks

Harness Acquires Propelo to Surface Software Engineering Bottlenecks

VC Daily: DOJ Sues Google; Why M&A Deals Fall Apart

Reduce Kubernetes spend with these 10 Kubecost alternatives

Harness Acquires Propelo

IT Budgets Under Pressure Spur Tool Consolidation

Harness® Acquires Propelo, Bringing Actionable Engineering Insights to Award-Winning Software Delivery Platform

Diversity and Inclusion: 7 Best Practices for Changing Your Culture

Developer Effectiveness and the Rise of Platform Engineering | 2023 Predictions By Harness

What is Hot for Fintech Investors in 2023?

Harness 2023 Predictions: The Year of Automation, Platform Engineering, and More

Workshop Wednesday: LIVE with Harness.io CRO

Partners

Contact us

The Modern Software Delivery Platform™ in Action

- Smart Automation, build pipelines in minutes
- Bake enterprise-grade security into pipelines
- Build insights for apps, teams, and pipelines
- Integrates with all clouds and tech stacks
- Supports Blue/Green & Canary strategies

Contact Harness

By signing up, you agree to our Privacy Policy and our Terms of Use.

Subscription Terms

As of Sep 21, 2023

These Harness Subscription Terms (collectively, the "Agreement") between Harness Inc., a Delaware corporation, with its principal place of business at 55 Stockton St., 8th Floor, San Francisco, CA 94108, U.S.A. ("Harness", "we", "us" or "our") and you ("Customer", "you" or "your") applies to your use of the Harness Platform (as defined below). By clicking on the designated button, entering an Order Form (as defined below), or by downloading, installing, accessing or using the Harness Platform, you agree to the terms of this Agreement. If you are entering into this Agreement on behalf of your organization or entity, you represent that you have the authority to bind such organization or entity to the Agreement, and the terms "Customer", "you" and "your" will refer to such organization or entity. If you do not agree to the terms of this Agreement, or if you are not authorized to accept this Agreement on behalf of your organization or entity, do not download, install, access or use the Harness Platform. "Harness Platform" means the downloadable and/or online software products that are specified in the applicable Order Form, or otherwise accessed by you, and subsequent updates made generally available by Harness under this Agreement, inclusive of the Delegate. The Harness Platform excludes Third Party Products, Extensions, and Beta Services.

1. Harness Platform

1.1. License Grant. Subject to payment of the applicable fees, Harness's receipt of a purchase order number from Customer (if needed), and Customer's ongoing compliance with the terms of this Agreement, Harness grants to Customer a limited, non-exclusive, non-transferable, non-sublicensable right and license to access and use the Software for internal business purposes in accordance with the Documentation (as defined below) during the applicable License Term, only for the number of License Units (as defined below) as specifically authorized by the applicable Order Form. "License Term" means the duration of your subscription or license to the applicable Harness Platform beginning and ending on the start and end dates, respectively, specified in the applicable Order Form, which include the Initial Term (as defined in Section 5), and all Renewal Terms (as defined in Section 5), as applicable. "License Unit" means a specific license type or metric, and a numeric quantity thereof, identified in an Order Form, to establish the extent and amount of Customer's license or right to use to the Harness Platform. All Order Forms are hereby incorporated into, supplement, and form a part of this Agreement. "Order Form" means an ordering document or online order that sets forth the applicable Harness Platform or Services (as defined below), fees and payment terms and start and end dates (as applicable) and is entered into between Customer and Harness, or Customer and an authorized reseller.

1.2. Restrictions on Use. Except as otherwise expressly provided in this Agreement, Customer shall not (and shall not permit any third party to): (a) sublicense, sell, resell, transfer, assign, distribute, share, lease, rent, make any external commercial use of, outsource, use on a timeshare or service bureau basis, or use in an application service provider or managed service provider environment, or otherwise generate income from the Harness Platform or Extensions; (b) copy the Harness Platform or Extensions onto any public or distributed network, except for an internal and secure cloud computing environment; (c) cause or permit the decompiling, disassembly, or reverse engineering of any portion of the Harness Platform or Extensions, or attempt to discover any source code or underlying algorithms or other operational mechanisms of the Harness Platform or Extensions (except where such restriction is expressly prohibited by law without the possibility of waiver, and then only upon prior written notice to Harness); (d) modify, adapt, translate or create derivative works based on all or any part of the Harness Platform or Extensions; (e) violate the Acceptable Use Policy ("AUP") located at <https://harness.io/legal/aup>; or (f) use free or trial accounts for certain products after having purchased License Units for the same products; (g) attempt to probe, scan or test the vulnerability of the Harness Platform (excluding the Delegate); (h) breach the security or authentication measures of the Harness Platform without proper authorization or willfully render any part of the Harness Platform or Extensions unusable; or (i) otherwise use the Harness Platform in violation of applicable law (including any export law) or outside the scope expressly permitted hereunder and in the applicable Order Form. Additionally, Customer shall not export or re-export, directly or indirectly, any Harness Platform or technical data or any copy, portions or direct product thereof (i) in violation of any applicable laws and regulations, (ii) to any country for which the United States or any other government, or any agency thereof, at the time of export requires an export license or other governmental approval, including Cuba, Iran, North Korea, Syria, the Crimea region of Ukraine, and the so-called Donetsk People's Republic, and Luhansk People's Republic regions of Ukraine, or any other Group D:1 or E:2 country (or to a national or resident thereof) specified in the then current Supplement No. 1 to part 740 of the U.S. Export Administration

Regulations (or any successor supplement or regulations, without first obtaining such license or approval) or (ii) to anyone on the U.S. Treasury Department's list of Specially Designated Nationals or the U.S. Commerce Department's Table of Denial Orders. Customer shall, at its own expense, obtain all necessary customs, import, or other governmental authorizations and approvals.

âDelegateâ means the software agent provided by Harness to Customer which facilitates Customer's use of the Harness Platform. For the purposes of this Agreement, the Delegate is not considered an Extension.

1.3. Free Access. If the Harness Platform is provided to Customer on a limited trial, free, or beta basis (âFree Accessâ), Customer agrees that such use and access of the Harness Platform is governed by this Agreement. Harness shall have the right to downgrade, limit or otherwise modify the Harness Platform provided for a Free Access account at any time without notice to Customer, and Harness disclaims all liability arising from Free Access, and shall have no liability, nor warranty, indemnity, maintenance, support, or security obligations to Customer with respect to any such Free Access. Customer may only use the number and type of License Units for the specified duration indicated by Harness prior to Customer downloading or accessing the Harness Platform with respect to any such Free Access. Harness may immediately revoke and terminate any Free Access at any time and without any notice to Customer. Customer agrees to provide feedback related to the Harness Platform as reasonably requested by Harness with respect to any Free Access. Customer agrees that Free Access is not a guarantee of any future Harness Platform or Harness product features. Customer will not utilize a Free Access account for the same products it has purchased.

1.4. Unauthorized Use. Customer shall notify Harness promptly of any unauthorized use or access of the Harness Platform (including unauthorized users or unauthorized disclosure of any password or account), or any other known or suspected breach of security or misuse of the Harness Platform. Customer is responsible for use of the Harness Platform (and all other acts or omissions) by its employees, contractors, Affiliates or other users that it allows to use or access the Harness Platform.

1.5. Support. During the License Term, Harness shall provide support to Customer in accordance with Harness's then-current support policy, and as identified in an Order Form. In the event that the level of support is not identified in the Order Form, Customer shall receive a âstandardâ level of support that is included with the Harness Platform at no additional cost. For any support tier above standard support, the applicable support fees will be a percentage of all of Customer's Harness Platform-based fees, and will be prorated for mid-year expansions based on the remaining months in the then current Initial Term or Renewal Term. Further, Customer agrees to facilitate any connections and access necessary for Harness to (i) deliver, deploy and provide the Harness Platform as provided hereunder and (ii) to perform its obligations hereunder (including any support obligations). Notwithstanding anything to the contrary in this Agreement, Harness has no warranty, indemnity or other obligation or liability with respect to modifications made to the Harness Platform or Documentation (as defined below) by Customer or on Customer's behalf other than the generally available updates provided by Harness.

1.6. Purchasing Through Authorized Resellers. If you purchase a subscription to the Harness Platform or any Services through a Harness authorized reseller (âPartnerâ), this Agreement and any agreed upon usage limitations will govern the use of such Harness Platform and Services unless otherwise agreed by Harness and Customer. You also agree that Harness is an express third party beneficiary of your agreement with any authorized reseller. Your payment obligations for the Harness Platform and Services will be with the authorized reseller as further described in Section 2.6 below, not Harness, and you will have no direct fee payment obligations to Harness, provided that Harness may terminate this Agreement if you breach any of your payment obligations to such authorized reseller for the Harness Platform and Services. Any terms agreed to between you and the authorized reseller that are in addition to or inconsistent with this Agreement are solely between you and the authorized reseller. No agreement between you and an authorized reseller is binding on Harness, nor will it have any force or effect with respect to the rights in, or the operation, use or provision of, the Harness Platform or Services.

1.7. Contractors and Third Party Providers. You may permit your authorized consultants, contractors, and agents (âThird Party Providersâ) to access and use the Harness Platform, but only on your behalf in connection with providing the Harness Platform to you, and subject to the terms and conditions of this Agreement. Any access or use by a Third Party Provider will be subject to the same limitations and restrictions that apply to you under this Agreement, and you will be responsible for any Third Party Provider's actions or omissions relating to its use of the Harness Platform. The aggregate use by you and all of your Third Party Providers must not exceed the allotted License Units (without paying the overage fees set forth in Section 2.2), and nothing in this Section is intended to or will be deemed to increase such License Units.

1.8. Services. Harness will use commercially reasonable efforts to provide the Services as described in an applicable Order Form (or statement of work referencing this Agreement entered into between the parties (âSOWâ)), if any. Harness will retain all right, title and interest in and to the deliverables and other results of the Services under this Agreement, and, subject to payment of the applicable fees and compliance with the terms of this Agreement, Harness hereby grants to Customer a limited, non-exclusive, non-transferable, non-sublicensable right and license to use such deliverables and results solely for Customer's internal business purposes and only in connection with Customer's permitted use of the Harness Platform. The Services (and any deliverables or other results) are not subject to any acceptance procedure. âServicesâ mean any training, enablement, consulting, installation and/or other professional services described in the applicable Order Form or SOW.

1.9. Customer Affiliates. Customer Affiliates may purchase and use the Harness Platform and Services subject to the terms of this Agreement by executing Order Forms or SOWs hereunder that incorporate by reference the terms of this Agreement. In each such case, all references in this Agreement to Customer shall be deemed to refer to such Customer Affiliate for purposes of such Order Form(s) or SOW(s), and Customer Affiliate agrees to be bound by this Agreement. âAffiliateâ means, with respect to Harness or Customer, any entity that directly or indirectly controls, is controlled by, or is under common control with Harness or Customer, respectively. âControl,â for purposes of this definition, means direct or indirect ownership or control of more than 50% of the voting interests of the subject entity.

1.10 Security. Harness will establish and maintain appropriate administrative, technical, and physical safeguards and controls to: (i) help ensure the ongoing confidentiality, integrity, availability, and resiliency of the Harness Platform and Customer Data, and (ii) have in place a process for regularly testing, assessing and evaluating the effectiveness of technical and organizational measures to help ensure

the security of the Harness Platformâs processing.

1.11 Extensions. Customer may use the Extensions solely in connection with the applicable Harness Platform subject to the Documentation, open source licenses, the applicable terms within this Agreement (including with respect to the Term), and the payment of any Fees associated with the Extensions. âExtensionâ means any separately downloadable or accessible suite, agent, configuration file, add-on, technical add-on, plug-in, example module, command, function, playbook, content or application that enables or extends the features or functionality of the applicable Harness Platform.

1.12 Third Party Products. This Agreement does not govern Customerâs use of Third Party Products used in connection with the Harness Platform. Third Party Products are governed solely by the terms and conditions between Customer and the Third Party Product developer. Harness does not make any commitments or claims regarding security, confidentiality, or performance of any Third Party Products, and specifically disclaims any liability regarding Third Party Products. Customer acknowledges and accepts that Third Party Products: (i) are activated and used at the sole risk of Customer; (ii) are not warranted, supported, or endorsed by Harness; and (iii) may degrade the performance of the Harness Platform beyond Harnessâs reasonable control. To the extent any Third Party Product accesses, processes, or gathers personal data, the applicable third party is Customerâs direct data processor, and is not acting as a data sub-processor of Harness. âThird Party Product(s)â means any product, software, application, platform, or service (i) selected by Customer (ii) not developed by Harness, and (iii) which integrates, interacts, or interoperates with, or adds functionality to, the Harness Platform.

2. Fees

â2.1. Fees. You agree to pay all fees specified in the Order Form and/or SOW, or as otherwise agreed upon by the parties. Fees are non-cancelable, non-refundable, and due and payable within thirty (30) days from the date of the invoice, or as otherwise specified in the Order Form. Unless otherwise agreed to by the parties in writing, all fees hereunder are payable in United States dollars. Additionally, Customer must provide all purchase order numbers (if applicable) to Harness by no later than the applicable Start Date as specified in the Order Form. All payments shall be made through automated clearing house (ACH) transfers, or wire transfers, to Harnessâs designated account, unless otherwise agreed by Harness. Fees do not include any customizations of the Harness Platform (nor support for any such customizations, unless otherwise agreed in writing).

2.2. Excess Usage. If Customerâs use of the Harness Platform exceeds the number of License Units set forth in the Order Form, Customer will be billed for and Customer will pay those overages at a prorated amount for the remainder of the applicable License Term under the Order Form, based on the pricing specified in the applicable Order Form. If Harness believes in good faith that Customerâs use of the Harness Platform exceeds the number of License Units set forth on the Order Form, Harness may (i) audit Customerâs use of the Harness Platform (not more frequently than twice per calendar year), upon at least twenty-four (24) hoursâ notice, and (ii) require that Customer provide Harness with all relevant records within five (5) business days of such request in order to determine if Customerâs use of the Harness Platform exceeds the number of authorized License Units.

2.3. Payment Terms. Customer acknowledges that purchases made under this Agreement are neither contingent on the delivery of any future functionality or features of the Harness Platform nor dependent on any oral or written public comments made by Harness regarding future functionality or features of the Harness Platform. All fees shall be fixed during the Initial Term (as defined in Section 5) unless Customer purchases additional License Units. If the number of purchased License Units does not increase upon renewal, then all Harness Platform and support fees will increase by the percentage specified in the Order Form at the start of each applicable Renewal Term (as defined in Section 5). If Customer is overdue on any payment, then Harness may (i) require that Customer pay a late fee equal to the lesser of 1.5% of the then-outstanding unpaid balance per month or the maximum amount allowable by law, and/or (ii) suspend Customerâs use of and access to the Harness Platform and/or Services associated with Customerâs account until such non-payment is corrected. Customer represents and warrants that the billing and contact information provided to Harness is complete and accurate, and Harness shall have no responsibility for any invoices that are not received due to inaccurate or missing information provided by Customer. All amounts invoiced under this Agreement and any Order Form shall be paid by Customer in full without any set-off, counterclaim, deduction, or withholding (excluding any tax withholding deductions as required by law).

2.4 Credit Cards. If Harness authorizes you to pay by credit or debit card in writing, you: (i) will provide Harness or its designated third-party payment processor with valid credit or debit card information; and (ii) hereby authorize Harness or its designated third-party payment processor to charge such credit or debit card for all items listed in the applicable Order Form or SOW or as otherwise agreed by the parties. Such charges must be paid in advance or in accordance with any different billing frequency stated in the applicable Order Form or SOW (if applicable). You are responsible for providing complete and accurate billing and contact information and notifying Harness in a timely manner of any changes to such information.

2.5 Taxes. The fees paid by Customer are exclusive of all taxes, levies, or duties (âTaxesâ) imposed by taxing authorities, if any, and Customer shall be responsible for payment of all such Taxes, excluding taxes based on Harnessâs income. Customer represents and warrants that the billing and contact information provided to Harness is complete and accurate, and Harness shall have no responsibility for any invoices that are not received due to inaccurate or missing information provided by Customer.

2.6 Payment Through Partner. Notwithstanding anything herein to the contrary, if Customer has licensed the Harness Platform from a Harness Partner, then Customer shall make its payments for the Harness Platform directly to such Partner, and the payment terms agreed by Customer and such Partner shall supersede and govern to the extent anything in this Section 2 conflicts with such payment terms.

3. Confidentiality

â3.1. Confidential Information and Restrictions. âConfidential Informationâ means all information disclosed by a party (âDisclosing Partyâ) to the other party (âReceiving Partyâ), whether orally or in writing, that is designated as âconfidentialâ or âproprietary,â or that, given the nature of the information or circumstances surrounding its disclosure, should reasonably be understood to be confidential. âConfidential Informationâ does not include any information that: (i) is or becomes generally known to the public without breach of any obligation owed to the Disclosing Party, (ii) was known to the Receiving Party prior to its disclosure by the Disclosing Party without

breach of any obligation owed to the Disclosing Party, (iii) is received from a third party without breach of any obligation owed to the Disclosing Party, or (iv) was independently developed by the Receiving Party. The Receiving Party will: (i) not use the Disclosing Party's Confidential Information for any purpose outside of this Agreement; (ii) not disclose such Confidential Information to any person or entity, other than its Affiliates, employees, consultants, subcontractors, subprocessors, agents and professional advisers (âRepresentativesâ) who have a âneed to knowâ for the Receiving Party to exercise its rights or perform its obligations hereunder, provided that such employees, consultants, and agents are bound by agreements or, in the case of professional advisers, ethical duties respecting such Confidential Information in accordance with the terms of this Section 3; and (iii) use reasonable measures to protect the confidentiality of such Confidential Information. The Receiving Party shall be liable for any breach of this section by its Representatives. If the Receiving Party is required by applicable law or court order to make any disclosure of such Confidential Information, it will first give written notice of such requirement to the Disclosing Party, and, to the extent within its control, permit the Disclosing Party to intervene in any relevant proceedings to protect its interests in its Confidential Information, and provide full cooperation to the Disclosing Party in seeking to obtain such protection. Further, this Section 3 will not apply to information that the Receiving Party can document: (i) was rightfully in its possession or known to it prior to receipt without any restriction on its disclosure; (ii) is or has become public knowledge or publicly available through no fault of the Receiving Party; (iii) is rightfully obtained by the Receiving Party from a third party without breach of any confidentiality obligation; or (iv) is independently developed by employees of the Receiving Party who had no access to such information.

3.2. Equitable Relief. The Receiving Party acknowledges that unauthorized disclosure of the Disclosing Party's Confidential Information could cause substantial harm to the Disclosing Party for which damages alone might not be a sufficient remedy and, therefore, that upon any such disclosure by the Receiving Party the Disclosing Party will be entitled to seek appropriate equitable relief in addition to whatever other remedies it might have at law or equity.

3.3. Feedback. Customer acknowledges and agrees that (a) any questions, comments, suggestions, ideas, feedback or other information about Harness, the Harness Platform, Extensions, the Services, the Documentation or other materials provided by Harness (collectively, âFeedbackâ) provided by Customer are non-confidential, (b) Harness will have full discretion to determine whether or not to proceed with the development of any requested enhancements, new features or functionality, and (c) Harness will have the full, unencumbered right, without any obligation to compensate or reimburse Customer, to use, incorporate and otherwise fully exercise and exploit any such Feedback in connection with its products and services.

4. Proprietary Rights

4.1 Ownership. Harness owns and shall retain all proprietary rights, including all copyright, patent, trade secret, trademark and all other intellectual property rights, in and to the Harness Platform (and all derivatives, improvements or enhancements thereof), System Data, Delegate, Extensions, Documentation, the results generated by the Harness Platform, and any Services, or any materials generated by Harness.Â

4.2 FOSS; Third Party Components. Customer acknowledges that the rights granted under this Agreement do not provide Customer with title to or ownership of the Harness Platform, in whole or in part. Certain âfreeâ or âopen sourceâ based software (the âFOSS Softwareâ) and third party software included with the Harness Platform (the âThird Party Softwareâ) is shipped with the Harness Platform but is not considered part of the Harness Platform hereunder. Use, reproduction, and distribution of FOSS Software is governed by the terms of the applicable open source software license and not this Agreement. Harness will provide Customer with a list of the FOSS Software and Third Party Software embedded in the Harness Platform upon request. With respect to Third Party Software included with the Harness Platform, such Third Party Software suppliers are third party beneficiaries of this Agreement. The Harness Platform and Third Party Software may only be used and accessed by Customer as prescribed by the instructions, code samples, on-line help files and technical documentation made publicly available by Harness for the Harness Platform, as may be updated from time to time by Harness (the âDocumentationâ). Harness will not be responsible for any act or omission of any third party, including the third party's access to or use of any Customer data or the performance of the Harness Platform in combination with such Third Party Software.

5. Term and Termination

This Agreement will be in full force and effect beginning on the earlier of the start date of the first Order Form entered into hereunder and the date you first access or otherwise use the Harness Platform, and will remain in effect until this Agreement is terminated pursuant to this Section. Termination of a specific Order Form will not affect the effectiveness of this Agreement or any other Order Form. Unless indicated otherwise in an Order Form, each Order Form shall be valid from the earliest start date therein through the initial end date therein (the âInitial Termâ), and shall automatically renew for additional successive twelve (12) month terms (each, a âRenewal Termâ), unless either party provides notice of non-renewal no less than thirty (30) days prior to the end of then-current Initial Term or Renewal Term, as applicable. If either party commits a material breach of this Agreement, and such breach has not been cured within thirty (30) days after receipt of written notice thereof, the non-breaching party may terminate this Agreement, except that Harness may immediately terminate this Agreement and/or terminate or suspend Customer's use of and access to the Harness Platform associated with Customer's account upon Customer's breach of Section 1.2 (Restrictions on Use) or Section 2.1 (Fees). Additionally, Harness may temporarily suspend access to the Harness Platform if Customer's use poses a security risk or adversely impacts Harness's business. Either party may also terminate this Agreement upon written notice if (a) the other party suspends payment of its debts or experiences any other insolvency or bankruptcy-type event or (b) there are no Order Forms or SOWs then in effect. Upon expiration or termination of an Order Form, for any reason, all rights granted to Customer with respect to such Order Form shall terminate and Customer shall destroy any copies of the Harness Platform and Documentation provided under such Order Form within Customer's possession and control. Upon any termination of this Agreement, each Receiving Party will return or destroy, at the Disclosing Party's option, the Disclosing Party's Confidential Information in the Receiving Party's possession or control. All payment obligations that have accrued as of such expiration or termination, any other rights or obligations that by their nature should survive, along with Sections 1.2, 1.3, 1.4, 1.6, 1.7, 2, 3, 4, 5, 6.2 and 7 through 10, will survive any expiration or termination hereof.

6. Warranties

6.1. Harness Platform Warranty. Harness warrants that during the first thirty (30) days after the beginning of a License Term under the applicable Order Form, the Harness Platform will, in all material respects, conform to the functionality described in the then-current Documentation for the applicable version of the Harness Platform. Harness's sole and exclusive obligation, and Customer's sole and exclusive remedy, for a breach of this warranty shall be that Harness will use commercially reasonable efforts to repair or replace the Harness Platform to conform in all material respects to the Documentation, and if Harness is unable to materially restore such functionality within thirty (30) days from the date of written notice of breach of this warranty by Customer, Customer shall be entitled to terminate the applicable Order Form upon written notice to Harness, and Harness shall promptly provide a pro-rata refund of the subscription fees under such Order Form that have been paid in advance for the remainder of the License Term under such Order Form (beginning on the date of termination). To be eligible for the foregoing remedy, Customer must notify Harness in writing of any warranty breaches within such warranty period, and Customer must have installed (if applicable), used and configured the Harness Platform in accordance with this Agreement and the Documentation.

6.2. Disclaimer. EXCEPT AS EXPRESSLY PROVIDED IN THIS SECTION 6, THE HARNESS PLATFORM, DOCUMENTATION, SERVICES, MAINTENANCE AND SUPPORT ARE PROVIDED "AS IS," AND HARNESS AND ITS SUPPLIERS EXPRESSLY DISCLAIM ANY AND ALL OTHER REPRESENTATIONS AND WARRANTIES, EITHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT THERETO, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, NON-INFRINGEMENT, OR THE CONTINUOUS, UNINTERRUPTED, ERROR-FREE, VIRUS-FREE, OR SECURE ACCESS TO OR OPERATION OF THE HARNESS PLATFORM. HARNESS EXPRESSLY DISCLAIMS ANY WARRANTY AS TO THE ACCURACY OR COMPLETENESS OF ANY INFORMATION OR DATA ACCESSED OR USED IN CONNECTION WITH THE HARNESS PLATFORM, DOCUMENTATION, SERVICES, MAINTENANCE OR SUPPORT. Additionally, Harness is not responsible for any delays, delivery failures, or any other loss or damage resulting from the transfer of data over communications networks and facilities, including the Internet, and Customer acknowledges that the Harness Platform, Delagate, Extensions, Services and Documentation may be subject to limitations, delays and other problems inherent in the use of such communications facilities. The Harness Platform is not fault-tolerant and is not designed or intended for use in hazardous environments, including without limitation, in the operation of aircraft or other modes of human mass transportation, nuclear or chemical facilities, life support systems, implantable medical equipment, motor vehicles or weaponry systems, or any other application in which failure of the Harness Platform could lead to death or serious bodily injury of a person, or to severe physical or environmental damage (each, a "High Risk Use"). Harness expressly disclaims any express or implied warranty or representation of fitness for High Risk Use. Harness shall not be liable to Customer for any loss, damage or harm suffered by Customer that is directly or indirectly caused by Customer's unauthorized use of the Harness Platform to process Prohibited Data. Prohibited Data means: (a) special categories of data enumerated in European Union Regulation 2016/679, Article 9(1) or any successor legislation; (b) patient, medical, or other protected health information regulated by the Health Insurance Portability and Accountability Act (as amended and supplemented) ("HIPAA"); (c) credit, debit, or other payment card data or financial account information, including bank account numbers or other personally identifiable financial information; (d) social security numbers, driver's license numbers, or other government identification numbers; (e) other information subject to regulation or protection under specific laws such as the Children's Online Privacy Protection Act or Gramm-Leach-Bliley Act ("GLBA") (or related rules or regulations); or (f) any data similar to the above protected under foreign or domestic laws.

6.3. Mutual Warranty. Each party hereby represents and warrants to the other that: (a) such party has the right, power, and authority to enter into this Agreement and to fully perform all of its obligations hereunder, (b) entering into this Agreement does not and will not violate any agreement or obligation existing between such party and any third party, and (c) this Agreement, when executed and delivered, will constitute a valid and binding obligation of such party and will be enforceable against such party in accordance with its terms.

6.4. Beta Software. From time to time, Customer may have the option to participate in a program with Harness where Customer is given access to alpha or beta software, services, products, features and documentation (collectively, "Beta Software") offered by Harness. The Beta Software is not generally available and may contain bugs, errors, defects or harmful components. Accordingly, Harness is providing the Beta Software to Customer "as is." Notwithstanding anything to the contrary in this Agreement, Harness makes no warranties of any kind with respect to the Beta Software, whether express, implied, statutory or otherwise, including any implied warranties of merchantability, fitness for a particular purpose, or non-infringement, and has no indemnity or other obligation or liability with respect to Beta Software. Harness does not warrant that the Beta Software will meet any specified service level, or will operate without interruptions or downtime.

7. Indemnification

7.1. By Harness. Harness agrees to defend, at its expense, Customer against (or, at Harness's sole option, settle) any third party claim to the extent such claim alleges that the Harness Platform infringes or misappropriates any patent, copyright, trademark or trade secret of a third party, and Harness shall pay all costs and damages finally awarded against Customer by a court of competent jurisdiction as a result of any such claim.

7.2. Remedies. In the event that the use of the Harness Platform is, or in Harness's sole opinion is likely to become, subject to such a claim, Harness, at its option and expense, may (a) replace the applicable Harness Platform with functionally equivalent non-infringing technology, (b) obtain a license for Customer's continued use of the applicable Harness Platform, or (c) terminate the applicable Order Form and provide a pro-rata refund of the subscription fees under such Order Form that have been paid in advance for the remainder of the License Term under such Order Form (beginning on the date of termination).

7.3. Limitations. The foregoing indemnification obligation of Harness will not apply: (1) if the Harness Platform is or has been modified by Customer or its agent; (2) if the Harness Platform is combined with other non-Harness products, applications, or processes, but solely to the extent the alleged infringement is caused by such combination; (3) to any unauthorized use of the Harness Platform or breach of this Agreement; or (4) if Customer fails to install or use any functionally equivalent non-infringing aspect of the Harness Platform that would have avoided the alleged infringement. The foregoing shall be Customer's sole remedy with respect to any claim of infringement of third party intellectual property rights.

7.4 By Customer. Customer agrees to defend, at its expense, Harness and its Affiliates, its suppliers and its resellers against any third party claim to the extent such claim alleges, arises from, or is made in connection with Customer's breach of Section 1 (Harness Platform) or Section 10 (Data Use), any High Risk Use, or Customer's negligence or willful misconduct. Customer shall pay all costs and damages finally awarded against Harness by a court of competent jurisdiction as a result of any such claim.

7.5 Indemnification Requirements. In connection with any claim for indemnification under this Section 7, the indemnified party must promptly provide the indemnifying party with notice of any claim that the indemnified party believes is within the scope of the obligation to indemnify, provided, however, that the failure to provide such notice shall not relieve the indemnifying party of its obligations under this Section 7, except to the extent that such failure materially prejudices the indemnifying party's defense of such claim. The indemnified party may, at its own expense, assist in the defense if it so chooses, but the indemnifying party shall control the defense and all negotiations related to the settlement of any such claim. Any such settlement intended to bind either party shall not be final without the other party's written consent, which consent shall not be unreasonably withheld, conditioned or delayed; provided, however, that Customer's consent shall not be required when Harness is the indemnifying party if the settlement involves only the payment of money by Harness.

8. Limitation of Liability

The limits below will not apply to the extent prohibited by applicable law. EXCEPT FOR LIABILITY ARISING FROM VIOLATIONS OF SECTION 1.2 (RESTRICTIONS ON USE), CUSTOMER'S PAYMENT OBLIGATIONS, OR A PARTY'S INFRINGEMENT OR MISAPPROPRIATION OF THE OTHER PARTY'S INTELLECTUAL PROPERTY RIGHTS, UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER IN TORT, CONTRACT, OR OTHERWISE, WILL EITHER PARTY BE LIABLE TO THE OTHER PARTY UNDER THIS AGREEMENT FOR ANY (A) INDIRECT, SPECIAL, INCIDENTAL, CONSEQUENTIAL OR PUNITIVE DAMAGES OF ANY CHARACTER, INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, LOST PROFITS, LOST SALES OR BUSINESS, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, LOST DATA, OR FOR ANY AND ALL OTHER INDIRECT DAMAGES OR LOSSES, EVEN IF SUCH PARTY HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY OF SUCH DAMAGES, OR (B) DIRECT DAMAGES, COSTS, OR LIABILITIES IN EXCESS OF THE AMOUNTS PAID BY CUSTOMER DURING THE TWELVE (12) MONTHS IMMEDIATELY PRECEDING THE INCIDENT OR CLAIM UNDER THE APPLICABLE ORDER FORM OR SOW. THESE LIMITATIONS SHALL APPLY NOTWITHSTANDING ANY FAILURE OF ESSENTIAL PURPOSE OF ANY REMEDY.

9. Data Use

9.1 System Data. Harness shall have the right collect and analyze System Data (including, without limitation, information concerning Customer and data derived therefrom), and Harness will be free (during and after the term hereof) to (i) use such information and data to improve and enhance the Harness Platform and for other development, diagnostic and corrective purposes in connection with the Harness Platform and other Harness offerings, and (ii) disclose such data solely in aggregate or other de-identified form in connection with its business. "System Data" means data collected by Harness regarding the Harness Platform that may be used to generate logs, statistics or reports regarding the performance, availability, usage, integrity or security of the Harness Platform.

9.2 Customer Data and License to Customer Data. "Customer Data" means electronic data submitted by Customer through the Harness Platform. As between the parties, Customer exclusively owns and reserves all rights, title, and interest in and to the Customer Data and Customer's software applications (and all derivatives, modifications, improvements, or enhancements related to any of the foregoing), which includes all copyright, patent, trade secret, trademark and all other intellectual property rights related to any of the foregoing. Customer hereby grants to Harness and its Affiliates a non-exclusive right to access, use, and process Customer Data, as necessary to provide the Harness Platform and the Services in accordance with this Agreement, and to improve the functioning and usability of the Harness Platform. Customer is solely responsible for the quality and integrity of Customer Data. Customer represents and warrants that it has obtained all the necessary consents to provide Harness with the foregoing license to Customer Data.

9.3 Personal Identifiable Information. If Customer provides Harness with any personally identifiable information ("Personal Data"), Customer represents and warrants that such Personal Information is not Prohibited Data, has been collected by Customer in accordance with the provisions of all applicable data protection laws and regulations, and that Customer has all right and consents necessary to provide such Personal Data to Harness.

10. Miscellaneous

10.1 Governing Law; Venue. This Agreement shall be governed by and construed under the laws of the State of California, U.S.A, as if performed wholly within the state and without giving effect to the principles of conflict of law. The parties consent to the exclusive jurisdiction and venue of the courts located in and serving San Francisco, California. The Uniform Computer Information Transactions Act (UCITA) nor the United Nations Convention for the International Sale of Goods will apply to this Agreement.

10.2 No Waiver. Failure by either party to exercise any of its rights under, or to enforce any provision of, this Agreement will not be deemed a waiver or forfeiture of such rights or ability to enforce such provision. If any provision of this Agreement is held by a court of competent jurisdiction to be illegal, invalid or unenforceable, such provision will be amended to achieve as nearly as possible the same economic effect of the original provision and the remainder of this Agreement will remain in full force and effect.Â

10.3 Entire Agreement. This Agreement (including each Order Form and SOW) represents the entire agreement between the parties and supersedes any previous or contemporaneous oral or written agreements or communications regarding the subject matter of this Agreement.Â

10.4 Order of Precedence. This Agreement shall control over additional or different terms of any purchase order, confirmation, invoice, statement of work or similar document (other than the Order Form or SOW, which will take precedence), even if accepted in writing by both parties, and waivers and amendments to this Agreement shall be effective only if made by non-pre-printed agreements clearly

understood by both parties to be an amendment or waiver to this Agreement.Â

10.5 Interpretation. All capitalized terms used but not defined in an Order Form or SOW shall have the meanings provided to them in the Agreement. For purposes of this Agreement, âincludingâ means âincluding without limitation.âÂ

10.6 Severability. If any provision of this Agreement is held by a court of competent jurisdiction to be illegal, invalid or unenforceable, such provision will be amended to achieve as nearly as possible the same economic effect of the original provision and the remainder of this Agreement will remain in full force and effect.

10.7 Cumulative Remedy. The rights and remedies of the parties hereunder will be deemed cumulative and not exclusive of any other right or remedy conferred by this Agreement or by law or equity.Â

10.8 Independent Contractors. No joint venture, partnership, employment, or agency relationship exists between the parties as a result of this Agreement or use of the Harness Platform.Â

10.9 Assignment and Delegation. Customer may not assign this Agreement without the prior written consent of Harness, and any purported assignment in violation of this Section 10.9 shall be void. Harness may assign, transfer or subcontract this Agreement in whole or in part without Customerâs consent. Upon any assignment of this Agreement by Customer that is approved by Harness or other corporate transaction involving Customer that would materially increase its Licensee Unit usage, if the Order Form contains a subscription for an âunlimitedâ amount of Licensee Units, such subscription will, with respect to Customer or the successor entity, as applicable, be capped at the monthly average of authorized Licensee Units used by Customer under such Order Form during the three full calendar months prior to such assignment (or if the Harness Platform has been used for fewer than three full calendar months, then the monthly average based on a pro rata calculation of such use). Harness reserves the right to perform its obligations from locations and/or through use of Affiliates, contractors and subcontractors, worldwide, provided that Harness will be responsible for such parties.

10.10 Force Majeure. With the exception of Customerâs payment obligations, no delay, failure, or default, other than a failure to pay fees when due, will constitute a breach of this Agreement to the extent caused by epidemics, acts of war, terrorism, hurricanes, earthquakes, cyberattacks, other acts of God or of nature, strikes or other labor disputes, riots or other acts of civil disorder, embargoes, government orders responding to any of the foregoing, or other causes beyond the performing partyâs reasonable control.

10.11 Publicity. Customer agrees that Harness may refer to Customer by its trade name and logo, and may briefly describe Customerâs business, in Harnessâs marketing materials and website. Additionally, Customer and Harness shall collaborate in good faith for the purpose of executing various co-marketing activities (e.g., customer testimonial videos, case study write ups, conference speaking slots, serving as a referenceable customer, etc.). The parties agree that all co-marketing activities will be contingent on a successful deployment of the Harness Platform.

10.12 Notice. Harness may give notice to Customer by electronic mail to Customerâs email address as provided by Customer on the Order Form or on record in Customerâs account information, or by written communication sent by first class mail or pre-paid post to Customerâs address as provided by Customer on the Order Form or on record in Customerâs account information. Customer may give notice to Harness at any time by any letter delivered by nationally recognized overnight delivery service or first class postage prepaid mail to Harness at the following address or such other address as may be notified to Customer from time to time: Harness, 55 Stockton St., 8th Floor, San Francisco, CA 94108, Attn.: Legal Department. Notice under this Agreement shall be deemed given when received, if personally delivered; when receipt is electronically confirmed, if transmitted by email; the day after it is sent, if sent for next-day delivery by a recognized overnight delivery service; and upon receipt, if sent by certified or registered mail, return receipt requested.Â

10.13 Updates. Harness may update this Agreement from time to time by providing you with prior written notice of material updates at least thirty (30) days in advance of the effective date. Such notice will be given in accordance with this section. Except as otherwise specified by Harness, (a) updates will be effective upon the effective date indicated at the top of this Agreement or in such notice, (b) your continued access or use of the Harness Platform or Services on or after the effective date of such updates constitutes your acceptance of such updates and (c) if you do not agree to such updates, you should stop using the Harness Platform and Services. However, if you have paid for a subscription to the Harness Platform, and we update this Agreement during your License Term, the updates with respect to that subscription will be effective upon your next Renewal Term, if applicable, and in this case, if you object to the updates, as your sole and exclusive remedy, you may choose not to renew, in accordance with the terms hereof. The updated version of the Agreement will supersede all prior versions.

Source URL: <https://www.harness.io/legal/privacy>

Privacy Statement

As of

Last Updated: July 27, 2023

â

Previous Privacy Statements / Policies can be found [here](#).

Scope

Harness Inc. (âHarness,â âWe,â âOur,â âUsâ) is committed to protecting the privacy of its customers, business partners, event attendees, job applicants and website visitors. This Harness Privacy Statement (âPrivacy Statementâ) reflects our global privacy practices and

standards as of the Last Updated date and 12 months prior. This Privacy Statement details our privacy practices for the collection, use, processing, storage, hosting, transfer, and disclosure of information that we may collect about you through interacting directly with Harness or our websites, including, but not limited to the website or subdomains of Harness.ioÂ (e.g., our public facing sites and support site), other websites or applications owned and controlled by Harness (collectively, the âWebsiteâ), along with our subsidiaries, products, and services that link to this Privacy Statement (collectively, the âServiceâ).Â

This Privacy Statement is applicable to Harness as the Data Controller of our customersâ information that relates to an identified or identifiable individual (âPersonal Dataâ). This Privacy Statement DOES NOT apply to Harness acting as the Data Processor. Additionally, our Service is not directed at or intended for the use of children. We do not knowingly collect the Personal Data of minors. For our privacy information regarding AIDA, our AI Developer Assistant, please see the AIDA Privacy Policy.

Harness as the Data ControllerÂ

Harness serves as the Data Controller of your Personal Data, as described in this Privacy Statement, unless otherwise stated. As the Data Controller Harness is responsible for and controls the processing of your personal information collected through our Service.

Harness as the Data ProcessorÂ

Harness serves as the Data Processor on our customers behalf. We will process information in accordance with the agreements we enter with our customers, who serve as the Data Controller. Please note, Harness is not responsible for the privacy or security practices of our customers, which may differ from those set forth in this Privacy Statement.Â For information regarding how Personal Data is managed or protected by Harness customers or to exercise your privacy rights, for information provided to us by your employer, please contact your employer.Â

Information We CollectÂ Â

- **Contact Information and Identifiers** such as your name, email address, mailing address, or telephone number.
- **Professional and Business Data** such as employer, job title, or certifications, business phone, business email address, or industry.
- **Online Identifiers** such as IP address, location details, username, social media identifiers and profiles, device OS, or internet browser.
- **Marketing, Sales, Training and Demo** related information such as products and services of interest, calendar details, video and audio recordings.
- **Account Registration, Customer Account, and Financial Information** such as account ID, authentication credentials, products in use, payment information, or billing details.Â
- **Support and Communication** such as email communications or service tickets.
- **Employment Application Data** such as resume, work experience, education, salary, or background check information. Please note, if an offer is extended sensitive information may be requested such as Social Security Number, passport, or other government identifier, racial or ethnic origin.
- **Single Sign-On Data** such as authentication tokens. WE DO NOT receive your login credentials.Â
- **Analytics and Log Data** such as the most used features, time spent on a page, and page visits pages.
- **Web Session Data** such as cookies, beacons, application and website usage activity.

How We Collect Information

Provided By You Â We collect information you provide to us when you:Â

- Sign up for or request information regarding our products and services;Â
- Communicate with us for support, information requests, or demos;Â
- Provide feedback or post on community forums;Â
- Register for, attend, or participate in a Harness event, training, or promotions;Â
- Visit our offices; or
- Inquire about or apply for employment.

Provided By 3rd Parties Â We collect information from 3rd parties:

- When you register for, attend, or participate in events where we are a sponsor or website forms hosted by third parties that may provide content about us;
- When you apply for a job or we receive an employment referral;
- When you participate in an open-source project or our public bug bounty program;Â
- From companies such as information aggregators and entities from whom we have licensed business contact information;
- From our partners or affiliates for sales leads; or
- When partnering, investing, or acquiring your employing or retaining company.

Automatically Collected Â We automatically collect information (via Cookies & Beacons):

- When you interact with our websites; or
- When you utilize our products and services.

Cookies & Beacons

Harness automatically collects information via cookies and beacons.Â Cookies are small pieces of information that are stored on your hard drive or in device memory. We may use both session Cookies (which expire once you close your web browser) and persistent Cookies (which stay on your computer until you delete them) to provide you with a more personal and interactive experience on our

Website. The categories of cookies used are described below:

Strictly Necessary Cookies - These cookies are necessary for the website to function as intended and cannot be turned off.

Functional Cookies - We use functional cookies to help enhance our websitesâ performance, functionality and personalisation. Disabling use of these cookies may prevent services from functioning properly.

Performance CookiesÂ - These cookies allow us to count visits and traffic sources so we can measure and improve the performance of our site. They help us to know which pages are the most and least popular and see how visitors move around the site.

Targeting Cookies - We use targeting and advertising cookies to help us understand our marketing efforts and to reach potential customers across the web. If you do not allow these cookies, you will experience less targeted advertising.

Beacons Â - We use beacons in our websites and in email communications to you. Beacons provide us with information about your activity and help us to improve our business operations and strategy such as by understanding our email communicationsâ functionality and improving our Website and content. For example, if you click on a marketing email we send to you about a new product or service, the beacon will provide signals to us that you and your organization may be interested in learning more.

How We Use Collected InformationÂ

How Harness uses the Personal Data it collects depends, in part, on how you choose to communicate with us, how you use our Websites and interact with us, and any preferences you have communicated to us. We use the information we collect for following legitimate business interests, legal obligations, and commercial purposes (e.g. Service Delivery and Fulfillment, Consent, Public Interest):

- Fulfill the original purpose for which the Personal Data was collected;Â
- Provide Harness products and services requested;
- Register, verify, and administer accounts;
- Process payments for services provided;
- Determine how our products and services are used and how they perform;
- Enhance and innovate our products and services;
- Improve the security of our products and services;
- Provide customer support and troubleshoot issues;
- Conduct data analysis and determine trends; or
- Communicate transactional notices, updates, security alerts, and administrative messages regarding our products and services;
- Promote and communicate marketing related information such as new products, features and enhancements;
- Support marketing promotions and contests;
- Identify and protect against misuse, policy violations, suspicious, or fraudulent activity;
- Support recruitment, employment and staffing decisions; and
- Support and comply with legal claims, regulatory obligations and audits.

When We Share Personal Information

We take care to only share, transmit, or grant access to Personal Data when there is a business need.Â We only share personal information if there is a legitimate need to know, enabling us to deliver our products and services and ensure appropriate privacy and security controls are in place to protect personal data. The parties and scenarios in which we may share personal data with include the following:

- Partners and subsidiaries - these are companies we have created, acquired, partnered or merged with;
- Service Providers, vendors, and sub-processors - these are companies we have contracted with toÂ provide services on our behalf including but not limited to hosting our Services, financial services, insurance providers, advertising firms, event sponsors, and background check services;
- Regulatory bodies, legal firms and advisors, or law enforcement agencies; or
- With your consent.

Please note Harness does not sell Personal Information for monetary value. However, we may disclose Personal Information to third parties, such as our subprocessors, to deliver our products and services, which is considered a sale of Personal Information as defined by CCPA.Â

Intentional Disclosures to Third PartiesÂ

Websites

As part of the functionality we make available on our Websites and to better reach our customers and prospective customers, there may be categories of third parties that are authorized by us to operate on our Websites and access your Personal Data, such as your contact data, IP address or cookies. Depending on your location (for example, California and the European Union), Harness only shares Personal Data with such third parties if you agree to such sharing via your privacy setting selections. In other parts of the world, this information may be automatically collected when you visit our websites. At any time, you may choose to withdraw your decision to share personal data with these third parties through our websites by visiting theÂ Privacy Rights and Choices section below.Â

Products and Services

As a part of our Service delivery and fulfillment obligations we may share Personal Data, such as account and financial information, with the applicable third parties. Harness only shares the Personal Data with such third parties as required to deliver services.Â

International Data Transfers

Your Personal Data may be collected, transferred to, and stored by us in the United States, and by our employees, subsidiaries and third parties that are based in other countries. Therefore, Personal Data may be processed outside your jurisdiction, and in countries that are not subject to an adequacy decision by the European Commission or your local legislature and/or regulator, and that may not provide for the same level of data protection as your jurisdiction, such as the European Economic Area. To ensure that the recipient of your Personal Data offers an adequate level of data protection, standard contractual clauses (SCCs), as detailed in GDPR Article 46, are utilized when data is transferred.Â Â Â

How We Secure Your Data

Data Protection

Harness has an inherent responsibility to protect the data our customers share with us. Our Services are built with privacy in mind and are designed to be used in a manner consistent with U.S. and international data privacy regulations. We continue to look for innovative ways to improve our overall security posture, identify and mitigate any potential risks. Information Security at Harness is, therefore, a critical business function which we have incorporated into all aspects of our business practices and operations. We maintain a comprehensive, written information security program that contains industry-standard administrative and technical safeguards designed to prevent unauthorized access to or disclosure of Personal Data. Additional security related information can be found at our Trust Center.

Data Retention

We will retain your Personal Data for a period of time that is consistent with the original purpose of the data collection, or as necessary to comply with our legal obligations, resolve disputes, and enforce our agreements. We determine the appropriate retention period for Personal Data by considering the amount, nature and sensitivity of your Personal Data processed, the potential risk of harm from unauthorized use or disclosure of your Personal Data, whether we can achieve the purposes of the processing through other means, and on the basis of applicable legal requirements (such as applicable statutes of limitation).

Privacy Rights and ChoicesÂ

The information below explains your privacy rights, the choices you have regarding how your Personal Data is managed, and how to exercise your rights. Depending on your jurisdiction the privacy rights you are entitled to may differ. However, Harness respects your privacy, as such we will make our best efforts to honor your privacy rights and choices regardless of locale. All parties have the rights listed below.Â

Right To Know: You can request that we disclose the Personal Data we have collected. Please submit a request in our Privacy Request Center.

Right To Delete (To Be Forgotten):Â You can request that we delete the Personal Data we have collected.Â Please submit a request in our Privacy Request Center.

Please note that we reserve the right to retain limited information as needed to fulfill our business and regulatory obligations (e.g, to deliver products and services, accounting transactions, legal matters).

Right To Correct (To Rectification): If you believe that the Personal Data we have is inaccurate you can request we correct it. Please submit a request in our Privacy Request Center.

Right To Portability: You may request to have your Personal Data provided to you in a machine readable format. Please submit a request in our Privacy Request Center.

Right to Opt Out of Automated Decision-Making Technologies: You have the right to not be subject to a decision based solely on automated processing, including profiling.Â

Please note that Harness does not use related technologies in regards to Personal Data.Â

Right To Opt Out of the selling, sharing or processing: You may request we not share or continue to process your Personal Data, please submit a request at Sell Share opt-out.Â

Right To Opt Out of email marketing:Â If you wish to withdraw from direct email marketing communications from Harness, you may click the âunsubscribeâ link included in our emails.Â

Right To Opt Out of interest-based analytics and advertising: If you wish to opt-out of interest-based advertising, please review the Cookies & Beacons section, and the Cookies link at the bottom of Harness.io.Â

Right To Opt Out of platform based analytics: If you are a user of the Harness online service via a subscription purchased for you by a Harness customer, and you wish to opt-out of platform-based analytics on an individual level, please submit a request at pt-out.

Right to Non-Discrimination and Non-Retaliation: You have the right not to be discriminated against for exercising any of your data rights.

California Privacy Rights (CCPA / CPRA)

Under the California Consumer Privacy Act of 2018 (CCPA), effective January 1, 2020, and the California Privacy Rights Act (CPRA),

effective January 1, 2023, which amended CCPA, California residents also have the following rights (in addition those detailed in Privacy Rights and Choices):

Right to Know of Automated Decision Making: If automated decision-making technologies are in use you have the right to know how the technology works and the possible outcome. Please note that Harness does not use related technologies in regards to Personal Data.

Right to Opt In for Minors: Minors, parents or guardians have the right to manage the collection and use of Personal Data. Explicit consent via an opt-in versus an implied consent with an opt-out option is required. Please note Harness does not knowingly collect the Personal Data of minors.

Right to Limit the Use and Disclosure of Sensitive Personal Information (SPI): You have the right to limit the use of your Sensitive Personal Data for specific purposes.

Right to Authorized Agents: In certain circumstances California residents are permitted to use an authorized agent on their behalf. The Data Subject must assign an Authorized Agent via a written signed letter and must be able to verify their identity.

EU Privacy Rights (GDPR)

Under the General Data Protection Regulation (GDPR), if you are in the European Union you also have the following rights (in addition those detailed in Privacy Rights and Choices):

Right to Restrict Processing: You have the right to restrict the processing of Personal Data if:

- The accuracy of the data is under question;
- Processing is unlawful;
- It is needed in order to establish or exercise legal claims or defenses; or
- You have exercised the right to object.

Right to Object: You have the right to object to the processing of Personal Data if the data is not being used for a legitimate purpose.

Right to Lodge a Complaint: You have the right to file a complaint with your local Data Protection Authority. The UK Information Commissioner's Office can be found here. To contact the Swiss Federal Data Protection and Information Commissioner can be reached here.

Contact Information

For questions regarding our Privacy Statement please email privacy@harness.io or contact us at:

Harness, Inc.

Attn: Legal Department

55 Stockton Street, 8th Floor

San Francisco, CA 94108

The addresses of our offices can be found [here](#).

Quick reference - How to exercise your rights

Rights	Links
Right to Know	
Right to Delete	
Right to Correct	Privacy Request Center
Right to Portability	
Right To Opt Out of the selling, sharing or processing	Sell Share opt-out
Right To Opt Out of email marketing:	Unsubscribe (email footer)
Right To Opt Out of platform based analytic	Analytics opt-out

Source URL: <https://www.harness.io/products/continuous-delivery/ai-assisted-deployment-verification>

AI-assisted Deployment Verification

Harness eliminates the need to monitor deployment health by using AI to automatically verify metrics and logs. You can automatically rollback a deployment if a regression is found.

Verify your Deployments

with your Metrics

Taking the toil out of manually correlating deployment metrics, Harness has the ability to verify your deployments from a wide variety of metric providers such as AppDynamics, Datadog, Prometheus, and Cloudwatch. Harness can analyze the incoming metrics and compare those metrics against a stable baseline and determine if you are trending towards regression.

Verify your Deployments with Logs

Harness has the ability to verify your deployments from a wide variety of log providers such as Splunk, Elasticsearch, Grafana Loki, and Google Cloud Ops. Harness can analyze log events for their presence and increase in frequency against a stable baseline using unsupervised machine learning.

Take automatic or manual action

As part of your Harness Pipeline, Deployment Verification can follow any failure strategy that a Harness Pipeline supports. You can automatically rollback a deployment if a regression is found or bring in human intervention with smart notifications.

Leverage continuous verification today

Monitor Deployment Health

Monitor the health of a deployment by validating metrics and logs from one or many sources

Rollback Automation

Automated rollback of problem releases tied to deployment strategy

Smart Notification

Optionally bring in human intervention with smart notifications

Ride the wave of Modern Software Delivery

Have a question? We are here to help!

Source URL: <https://www.harness.io/legal/website-terms-of-use>

Website Terms Of Use

As of Sep 20, 2023

Welcome to **www.harness.io**(together with any related websites, collectively, the "Site"). The Site is owned and operated by Harness Inc. (herein referred to as the "Harness", "we", "us" or "our"). Please read these Website Terms of Use ("Terms") carefully before using the Site.

1. Acceptance of Terms

1.1 By accessing, browsing, or otherwise using the Site or any of the content on the Site you acknowledge that you have read, understood, and agree to be legally bound by these Terms. If you do not accept these Terms, you shall not access, browse or use the Site or any of its content.

1.2 You understand and agree that we may change these Terms at any time without prior notice. You may read a current, effective copy of these Terms at any time by selecting the "Website Terms of Use" link on the Site. The revised terms and conditions will become effective at the time of posting. You can determine when these Terms were last revised by referring to the date at the top of these Terms. Any use of the Site after such date shall constitute your acceptance of such revised terms and conditions. If any change to these Terms is not acceptable to you, your sole remedy is to cease accessing, browsing and otherwise using the Site.

1.3 Your access to and use of the Site is also subject to the Harness Privacy Policy located at <https://harness.io/privacy/>, the terms and conditions of which are hereby incorporated herein by reference.

1.4 You represent to Harness that you are lawfully able to enter into contracts (e.g., you are not a minor). If you are entering into these Terms for an entity, such as the company or organization you work for, you represent that you have authority to bind that entity and you agree that "you" as used in these Terms includes both you personally and the entity you represent. You and Harness are collectively referred to as the "Parties" and each is a "Party".

2. Use of the Site

2.1 These Terms do not govern the use of any Harness product or services sold or provided by Harness. Your right to access, download, and/or use any of the Harness products or services made available through the Site or other means is governed by the applicable online software license agreement, or such other written contract as may be separately agreed and signed between you and Harness.

2.2 You agree not to: (a) take any action that imposes an unreasonable load on the Site's infrastructure, (b) use any device, software or routine to interfere or attempt to interfere with the proper working of the Site or any activity being conducted on the Site, (c) attempt to decipher, decompile, disassemble or reverse engineer any of the software comprising or making up the Site, (d) delete or alter any material posted on the Site by Harness or any other person or entity, or (e) frame or link to any of the materials or information available on the Site.

2.3 You represent and warrant that any information that you provide in connection with your use of the Site is and shall remain true, accurate, and complete, and that you will maintain and update such information regularly. You agree that if any information that you provide is or becomes false, inaccurate, obsolete or incomplete, Harness may terminate your use of the Site.

Unless otherwise specifically agreed to by you and Harness in writing, by uploading, e-mailing, posting, submitting, publishing or otherwise transmitting information, data, event types, tags, comments, suggestions, content or other materials to the Site or Harness (each a "Contribution"), you hereby acknowledge that such Contribution is non-confidential and automatically grant (or warrant that the owner of such rights has expressly granted) to Harness a perpetual, irrevocable, world-wide, non-exclusive, sublicensable, fully paid-up and royalty-free license to use, make, have made, copy, distribute, perform, display (whether publicly or otherwise), modify, adapt, publish, and transmit such Contributions in any form, medium, or technology now known or later developed, and to grant to others rights to do any of the foregoing. In addition, you represent and warrant that all so-called moral rights in the content have been waived. For each Contribution, you represent and warrant that you have all rights necessary for you to grant the licenses granted in this Section, and that such Contribution, and your provision thereof to and through the Site, complies with all applicable laws, rules and regulations. Harness cannot and will not be liable for any loss or damage arising from your failure to comply with this Section.

Harness will not pre-screen or review Contributions, but Harness reserves the right to refuse or delete any Contributions in its discretion. You acknowledge and agree that Harness reserves the right (but has no obligation) to do one or more of the following in its discretion, without notice or attribution to you: (i) monitor Contributions as well as your access to the Site; (ii) alter, remove, or refuse to post or allow to be posted any Contribution; and/or (iii) disclose any Contributions, and the circumstances surrounding their transmission, to any third party in order to operate the Site, in order to protect Harness, its suppliers or licensees and their respective employees, officers, directors, shareholders, affiliates, agents, representatives, and the Site's users and visitors; to comply with legal obligations or governmental requests; to enforce these Terms; or for any other reason or purpose.

Harness disclaims any responsibility for the Contributions displayed on its Site and assumes no responsibility for the timeliness, deletion, mis-delivery or failure to store any Contributions or other user information or personalization settings.

3. Rules of Conduct

3.1 While using the Site you agree to comply with all applicable laws, rules and regulations. In addition, Harness expects users of the Site to respect the rights and dignity of others. Your use of the Site is conditioned on your compliance with the rules of conduct set forth in this Section; any failure to comply may also result in termination of your access to the Site pursuant to Section 13 (Termination). You agree that you will not:

- Post, transmit, or otherwise make available, through or in connection with the Site: Anything that is or may be (a) threatening, harassing, degrading or hateful; (b) defamatory; (c) fraudulent or tortious; (d) obscene, indecent or otherwise objectionable; or (e) protected by copyright, trademark or other proprietary right without the express prior written consent of the owner of such right. Any material that would give rise to criminal or civil liability or that encourages conduct that constitutes a criminal offense. Any virus, worm, Trojan horse or other computer code, file, or program that is harmful or invasive or may or is intended to damage or hijack the operation of any hardware or software. Any unsolicited or unauthorized advertising, promotional materials, "junk mail," "spam," a chain letter, a pyramid scheme or investment opportunity, or any other form of solicitation.
- Use the Site for any fraudulent or unlawful purpose. Harvest or collect personally identifiable information about other users of the Site. Interfere with or disrupt the operation of the Site or the servers or networks used to make the Site available; or violate any requirements, procedures, policies or regulations of such networks. Restrict or inhibit any other person from using the Site (including by hacking or defacing any portion of the Site). Use the Site to advertise or offer to sell or buy any goods or services without Harness's express prior written consent. Reproduce, duplicate, copy, sell, resell or otherwise exploit for any commercial purposes, any portion of, use of, or access to the Site (including any Content, software and other materials available through the Site). Modify, adapt, create derivative works of, translate, reverse engineer, decompile or disassemble any portion of the Site (including any Content, software and other materials available through the Site), except as and solely to the extent expressly authorized under applicable law overriding any of these restrictions. Remove any copyright, trademark or other proprietary rights notice from the Site or Content, software and other materials originating from the Site. Frame or mirror any part of the Site without Harness's express prior written consent. Create a database by systematically downloading and storing all or any Content. Use any robot, spider, site search/retrieval application or other manual or automatic device to retrieve, index, "scrape," "data mine" or in any way reproduce or circumvent the navigational structure or presentation of the Site, without Harness's express prior, written consent.

4. Harness Content

The Site contains HTML, software, applications, messages, text, files, images, photos, video, sounds, profiles, works of authorship and other content (collectively, "Content") of Harness or its licensors ("Harness Content"). The Site (including the Harness Content) is protected by copyright, trademark, trade secret and other laws; and as between you and Harness, Harness owns and retains all rights in the Site and the Harness Content. Harness grants to you a limited, revocable, non-sublicensable license to access, display and perform the Harness Content (excluding any computer code) solely for your personal, non-commercial use and solely as necessary to access and

use the Site. Except as expressly permitted by Harness in these Terms or on the Site, you may not copy, download, stream, capture, reproduce, duplicate, archive, upload, modify, translate, create derivative works based upon, publish, broadcast, transmit, retransmit, distribute, perform, display, sell or otherwise use or transfer any Harness Content. You may not, either directly or through the use of any device, software, online resource or other means, remove, alter, bypass, avoid, interfere with or circumvent any copyright, trademark or other proprietary notice on the Harness Content or any digital rights management mechanism, device, or other content protection or access control measure associated with the Harness Content. The use or posting of any of the Content on any other Site or in a networked computer environment for any purpose is expressly prohibited. If you violate any part of these Terms, your right to access and/or use the Content and Site shall automatically terminate and you shall immediately destroy any copies you have made of the Content.

The trademarks, service marks, and logos of Harness (the "Harness Trademarks") used and displayed on this Site are registered and unregistered trademarks or service marks of Harness. Other Harness, product, and service names located on the Site may be trademarks or service marks owned by third-parties (the "Third-Party Trademarks"), and, collectively with the Harness Trademarks, the "Trademarks"). Nothing on this Site or in these Terms should be construed as granting, by implication, estoppel, or otherwise, any license or right to use any Trademark displayed on this Site without the prior written consent of Harness specific for each such use. The Trademarks may not be used to disparage Harness or the applicable third-party, Harness's or third-party's products or services, or in any manner (using commercially reasonable judgment) that may damage any goodwill in the Trademarks. Use of any Trademarks as part of a link to or from any Site is prohibited without Harness's prior written consent. All goodwill generated from the use of any Harness Trademark shall inure to Harness's benefit.

5. Third-Party Sites

You may find links to other websites on the Site. Those links will let you leave the Site. Harness exercises no control whatsoever over such third-party websites and any contents or web-based resources found on those third-party sites and is not responsible or liable for the availability thereof or the content, advertising, products or other materials thereon or any updates or changes thereto. Harness is providing these links to you only as a convenience, and the inclusion of any link does not imply endorsement by Harness of any linked sites. Harness shall not be responsible or liable, directly or indirectly, for any damage or loss incurred or suffered by any user in connection therewith. Your access and use of those websites, including your use of any content, information, data, advertising, products, or other materials on or available through such websites, is solely at your own risk and subject to the terms and conditions of use and privacy policy(ies) applicable to such sites and resources. The Harness Privacy Policy is applicable only when you are on the Site. Once you choose to be directed to another website, you should read that website's privacy statement before disclosing any personal information.

6. Compliance with Laws

You represent that, in agreeing to, and performing under, these Terms, you are not violating, and will not violate, any governmental laws, rules, regulations or orders that are applicable to your use of the Site ("Applicable Laws"). Without limiting the foregoing, you represent that, in connection with your performance under these Terms, you shall: (a) comply with Applicable Laws relating to anti-bribery and anti-corruption, which may include the US Foreign Corrupt Practices Act of 1977 and the UK Bribery Act 2010; (b) comply with Applicable Laws administered by the U.S. Commerce Bureau of Industry and Security, U.S. Treasury Office of Foreign Assets Control or other governmental entity imposing export controls and trade sanctions ("Export Laws"), including designating countries, entities and persons ("Sanctions Targets"); and (c) not directly or indirectly export, re-export or otherwise deliver any Harness software, content or services to a Sanctions Target, or broker, finance or otherwise facilitate any transaction in violation of any Export Laws. You represent that you are not a Sanctions Target or prohibited from receiving Harness software, content or services pursuant to these Terms under Applicable Laws, including Export Laws.

7. Indemnity

You agree to defend, indemnify, and hold harmless Harness, its affiliates, and their respective employees, contractors, agents, officers and directors from and against any and all claims, damages, obligations, losses, liabilities, costs, debt or expenses (including without limitation attorneys' fees) arising out of or related to any claim, suit, action or proceeding by a third party arising out of or relating to your use of the Site, breach of these Terms (including any Harness policy referenced in these Terms), violation of law, or any Content that you post, upload or cause to interface with the Site, or otherwise transfer, process, use or store in connection with the Site.

8. Disclaimers

YOUR USE OF THE SITE IS AT YOUR OWN RISK. THE SITE AND ANY CONTENT, INFORMATION, PRODUCTS OR SERVICES MADE AVAILABLE ON OR THROUGH THE SITE ARE PROVIDED ON AN "AS IS" AND "AS AVAILABLE" BASIS WITHOUT WARRANTY OF ANY KIND. HARNESS AND/OR ITS SUPPLIERS AND LICENSORS HEREBY DISCLAIM ALL WARRANTIES AND CONDITIONS WITH REGARD TO THIS SITE OR ANY INFORMATION, CONTENT, PRODUCTS OR SERVICES CONTAINED THEREIN, WHETHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT. SPECIFICALLY, HARNESS MAKES NO WARRANTY THAT (I) THE SITE WILL MEET YOUR REQUIREMENTS, (II) ANY USER ACCESS TO THE SITE WILL BE UNINTERRUPTED, TIMELY, SECURE OR ERROR-FREE, (III) THE QUALITY OF ANY CONTENT, PRODUCTS, SERVICES, INFORMATION OR OTHER MATERIAL OBTAINED THROUGH THE SITE WILL MEET YOUR EXPECTATIONS, AND (IV) ANY ERRORS IN THE SOFTWARE WILL BE CORRECTED. THE SITE, THE PRODUCTS AND SERVICES AVAILABLE THROUGH THE SITE AND THE INFORMATION, CONTENT, SOFTWARE, DOCUMENTS, AND RELATED GRAPHICS PUBLISHED ON THIS SITE COULD INCLUDE TECHNICAL INACCURACIES, ERRORS, OR OMISSIONS. THE DISCLAIMERS OF WARRANTY AND LIMITATIONS OF LIABILITY APPLY, WITHOUT LIMITATION, TO ANY DAMAGES OR INJURY CAUSED BY THE FAILURE OF PERFORMANCE, ERROR, OMISSION, INTERRUPTION, DELETION, DEFECT, DELAY IN OPERATION OR TRANSMISSION, COMPUTER VIRUS, COMMUNICATION LINE FAILURE, THEFT OR DESTRUCTION OR UNAUTHORIZED ACCESS TO, ALTERATION OF OR USE OF ANY ASSET,

WHETHER ARISING OUT OF BREACH OF CONTRACT, TORTIOUS BEHAVIOUR, NEGLIGENCE OR ANY OTHER COURSE OF ACTION BY HARNESS.

9. Limitation of Liability

TO THE FULLEST EXTENT PERMITTED BY APPLICABLE LAW: (A) IN NO EVENT SHALL HARNESS, ITS AFFILIATES, OR THEIR RESPECTIVE EMPLOYEES, CONTRACTORS, AGENTS, OFFICERS OR DIRECTORS BE LIABLE FOR ANY INDIRECT, PUNITIVE, INCIDENTAL, SPECIAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES, INCLUDING WITHOUT LIMITATION DAMAGES FOR BUSINESS INTERRUPTION, LOSS OF PROFITS, GOODWILL, USE, DATA OR OTHER INTANGIBLE LOSSES ARISING OUT OF OR RELATING TO THE SITE; AND (B) IN NO EVENT SHALL HARNESSâS CUMULATIVE AND AGGREGATE LIABILITY UNDER THESE TERMS EXCEED ONE HUNDRED U.S. DOLLARS. THE EXCLUSIONS AND LIMITATIONS IN THIS SECTION APPLY WHETHER THE ALLEGED LIABILITY IS BASED ON CONTRACT, TORT, NEGLIGENCE, STRICT LIABILITY OR ANY OTHER BASIS, EVEN IF YOU HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. SOME STATES DO NOT ALLOW EXCLUSION OF IMPLIED WARRANTIES OR LIMITATION OF LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU. IN SUCH STATES, THE LIABILITY OF THE HARNESS PARTIES SHALL BE LIMITED TO THE GREATEST EXTENT PERMITTED BY LAW.

10. Digital Millennium Copyright Act

The Digital Millennium Copyright Act of 1998 (the âDMCAâ) provides recourse for copyright owners who believe that material appearing on the Internet infringes their rights under U.S. copyright law. If you believe in good faith that materials on the Site infringe your copyright, you (or your agent) may send us a notice requesting that the material be removed, or access to it blocked. Notices and counter-notices must meet the then current statutory requirements imposed by the DMCA (see www.loc.gov/copyright for details). Notices and counter notices with respect to the Site should be sent to Harness at:

By Mail: Copyright Agent 55 Stockton Street, San Francisco, CA 94108

By Email: legalnotices@harness.io

11. U.S. Government Rights

The Site is provided to the U.S. Government as âcommercial items,â âcommercial computer software,â âcommercial computer software documentation,â and âtechnical dataâ with the same rights and restrictions generally applicable to the Site. If you are using the Site on behalf of the U.S. Government and these terms fail to meet the U.S. Governmentâs needs or are inconsistent in any respect with federal law, you must immediately discontinue use of the Site. The terms listed above are defined in the Federal Acquisition Regulation and the Defense Federal Acquisition Regulation Supplement.

12. Notice for California Residents

Under California Civil Code Section 1789.3, California users are entitled to the following consumer rights notice: Harness does not currently charge any fees for access and use of the Site. If you have a question or complaint regarding the Site, please contact Harness by writing to: Harness, Inc., Attn: Legal, 55 Stockton St., 8th Floor, San Francisco, CA 94108, CA 94105, or legalnotices@harness.io, or by calling Harness at (855) 879-7727. California residents may reach the Complaint Assistance Unit of the Division of Consumer Services of the California Department of Consumer Affairs by mail at 1625 North Market Blvd., Sacramento, CA 95834, or by telephone at (916) 445â1254 or (800) 952â5210.

13. Termination of the Agreement

Harness reserves the right, in its sole discretion, to restrict, suspend, or terminate these Terms and your access to all or any part of the Site or the Content at any time and for any reason without prior notice or liability. Sections 2 (Use of the Site), 7 (Indemnification), 8 (Disclaimers), 9 (Limitation of Liability), 13 (Termination of the Agreement), and 14 (Miscellaneous) shall survive the termination of these Terms.

14. Miscellaneous

These Terms are governed by the internal substantive laws of the State of California, without respect to its conflict of laws provisions. You expressly agree to submit to the exclusive personal jurisdiction of the state and federal courts sitting in the City of San Francisco in the State of California. If any provision of these Terms is found to be invalid by any court having competent jurisdiction, the invalidity of such provision shall not affect the validity of the remaining provisions of these Terms, which shall remain in full force and effect. Failure of Harness to act on or enforce any provision of these Terms shall not be construed as a waiver of that provision or any other provision in these Terms. As mentioned in Section 1, Harness may modify these Terms at any time by posting a revised version on the Site. By accessing, browsing, or otherwise using the Site, you agree to the latest version of these Terms. No waiver shall be effective against Harness unless made in writing, and no such waiver shall be construed as a waiver in any other or subsequent instance. Except as expressly agreed by Harness and you, these Terms constitute the entire agreement between you and Harness with respect to the subject matter hereof, and supersedes all previous or contemporaneous agreements, whether written or oral, between the parties with respect to the subject matter. The section headings are provided merely for convenience and shall not be given any legal import. These Terms will inure to the benefit of our successors and assigns. Any information submitted or provided by you to the Site might be publicly accessible. Important and private information should be protected by you. Harness is not liable for protection of privacy of electronic mail or other information transferred through the Internet or any other network that you may use.

15. Contact

Please contact Harness at legalnotices@harness.io with any questions regarding these Terms.

Source URL: <https://www.harness.io/products/continuous-delivery/harness-gitops>

Harness GitOps

What is GitOps?

GitOps is simply using Git to perform Operations Team (Ops) tasks, such as building infrastructure, release automation, and orchestration.

How Harness does GitOps better?

Enterprise GitOps: More than Just ArgoCD.

Pull Request Pipelines

Pull request pipelines add a layer of automation on top of standard GitOps deployments, making it easy to propagate changes across multiple services and environments without manually triggering each promotion.

Intelligent Rollback

Harness includes continuous verification that leverages your observability and logging tools to monitor deployment health. Harness can create a rollback pull request to limit the impact of the problem when an anomaly is detected.

Fully Managed GitOps

Harness GitOps reduces the overhead of managing GitOps implementations. Instead of installing and managing GitOps infrastructure, Harness GitOps manages everything for you.

Bring your own Flux or Argo CD Instances

Harness makes it easy to import your existing GitOps applications and configurations into Harness GitOps - no need to recreate Argo CD or Flux based deployments from scratch, thus saving time.

Centralized Dashboards

Harness provides a centralized control plane across multiple GitOps clusters to view and manage GitOps deployments in a single pane of glass. Your teams can access granular service and environment health information, and even trigger rollbacks. You can also create custom Looker-based dashboards to meet your needs.

Drift Detection

GitOps-as-a-Service syncs changes made in Git with clusters running in environments. This ensures that your deployments match the desired state based on recent code changes and the application configuration is the same as what is stored in Git.

Application Sync

Harness GitOps syncs changes made in Git with clusters running in environments. This ensures that your clusters are always in the desired state.

Comprehensive APIs

Harness provides well-documented REST APIs for automation across our entire platform, including onboarding new teams and projects at scale. Our APIs enable you to automate and manage your end-to-end workflow, and integrate Harness into your existing tooling.

Ride the wave of Modern Software Delivery

Have a question? We are here to help!

Source URL: <https://www.harness.io/legal/trademark>

Trademark Guidelines

As of Dec 01, 2021

This trademark policy was prepared to help you understand how to use HARNESS INC.'s trademarks, service marks and logos with HARNESS INC.'s HARNESS® software.

While some of our software is available under a free and open-source software license, that copyright license does not include a license to use our trademarks, and this Policy is intended to explain how to use our marks consistent with background law and community expectation.

This Policy covers:

This policy encompasses all trademarks and service marks, whether they are registered or not.

1. General Guidelines

Whenever you use one of our marks, you must always do so in a way that does not mislead anyone about what they are getting and from whom. For example, you cannot say you are distributing HARNESS INC.'s HARNESS® software when you're distributing modified version of it, because recipients may not understand the differences between your modified versions and our own.

You also cannot use our logo on your website in a way that suggests that your website is an official website or that we endorse your website.

You can, though, say you like the HARNESS® software, that you participate in the HARNESS® community, that you are providing an unmodified version of the HARNESS® software.

You may not use or register our marks, or variations of them as part of your own trademark, service mark, domain name, company name, trade name, product name or service name.

Trademark law does not allow your use of names or trademarks that are too similar to ours. You therefore may not use an obvious variation of any of our marks or any phonetic equivalent, foreign language equivalent, takeoff, or abbreviation for a similar or compatible product or service. By way of example only, we would consider the following too similar to one of our Marks:

MyHARNESS

HARNESS Tools

HARNESS Software

2. Acceptable Uses

Unmodified code

When you redistribute an unmodified copy of our software -- the exact form in which we make it available -- you must retain the marks we have placed on the software to identify your redistribution.

Modified code

If you distribute a modified version of our software, you must remove all of our logos from it. To assist you with this, we have removed our logos from our source trees, and include them only in our binaries. You may use our word marks, but not our logos, to truthfully describe the origin of the software that you are providing. For example, if the code you are distributing is a modification of our software, you may say, "This software is derived from the source code for HARNESS INC.'s HARNESS® software."

Statements about compatibility

You may use the word marks, but not the logos, to truthfully describe the relationship between your software and ours. Any other use may imply that we have certified or approved your software. If you wish to use our logos, please contact us to discuss license terms.

Naming Compatible Products

If you wish to describe your product with reference to the HARNESS® software, here are the conditions under which you may do so. You may call your software XYZ (where XYZ is your product name) for HARNESS® only if:

- All versions of the HARNESS® software you deliver with your product are the exact binaries provided by us.
- You use the following legend in marketing materials or product descriptions: "HARNESS® is a trademark of HARNESS INC. www.harness.io"

User groups

You can use the Word Marks as part of your user group name provided that:

- The main focus of the group is our software

- The group does not make a profit
- Any charge to attend meetings are to cover the cost of the venue, food and drink only

You are not authorized to conduct a conference using our marks.

No Domain Names

You must not register any domain that includes our word marks or any variant or combination of them.

3. How to Display Our Marks

When you have the right to use our mark, here is how to display it.

Trademark marking and legends

The first or most prominent mention of a mark on a webpage, document, or documentation should be accompanied by a symbol indicating whether the mark is a registered trademark (â®) or an unregistered trademark (â¢). If you don't know which applies, contact us.

Place the following notice at the foot of the page where you have used the mark: âHARNESSâ® is trademark of Harness, Inc.â

Use of trademarks in text

Always use trademarks in their exact form with the correct spelling, neither abbreviated, hyphenated, or combined with any other word or words.

Unacceptable: HARNESS-DB

Don't pluralize a trademark.

Unacceptable: I have seventeen HARNESSES running in my lab.

Always use a trademark as an adjective modifying a noun.

Unacceptable: This is a HARNESSâ®.

Acceptable: This is a HARNESSâ® software application.

Use of Logos

You may not change any logo except to scale it. This means you may not add decorative elements, change the colors, change the proportions, distort it, add elements, or combine it with other logos.

However, when the context requires the use of black-and-white graphics and the logo is color, you may reproduce the logo in a manner that produces a black-and-white image.

This Document

These guidelines are based on the Model Trademark Guidelines, available at <http://www.modeltrademarkguidelines.org>, used under a Creative Commons Attribution

3.0 Unported license: https://creativecommons.org/licenses/by/3.0/deed.en_US.

Source URL: <https://www.harness.io/learn/resource-center?resource-type=Webinar>

Resources

Browse and register for e-books, guides, webinars, videos and upcoming events!

The Modern Software Delivery Platformâ®

Need more info? Contact Sales

Source URL: <https://www.harness.io/products/continuous-delivery/powerful-pipelines>

Build Powerful Pipelines

Pipeline studio provides a user-friendly and intuitive graphical interface for designing and configuring pipelines. It allows users to drag and drop components, stages, and pipeline canvas.

Easy to author pipelines

Along with the graphical interface, Pipeline studio offers a YAML editor for developers who prefer authoring pipelines as code.

Failure Strategies

Stay in control, even when things go wrong. Harness treats the failure of deployment steps as an expected event, making it easy to declare what should happen.

Should it contact an approver if a quality policy is violated, retry a flaky step, or automatically rollback if telemetry from production looks bad? You decide, and Harness will automate the right action.

Looping and Matrixes

Looping and matrices: Run an action, or a group of actions called a *stage*, repeatedly. Setup matrix actions to process combinations like testing each of three browsers on two different operating systems.

Manual Gates

Add manual approval gates to require human intervention, which can be crucial for compliance and risk management.

Harness has built-in approval steps, and can also integrate with ticketing systems like Jira or ServiceNow.

Conditional Logic

Conditional logic: Set up conditional behavior based on environment variables, deployment success/failure, or other criteria.

Insightful Pipeline Execution Panel

The execution panel offers access to logs, outputs, and diagnostic information for each pipeline execution, providing invaluable metrics for troubleshooting and investigation of failures. Users can also view the progress of individual pipeline stages and actions in real-time. Historical execution data is retained allowing post-deployment analysis and auditing based on past pipeline executions, their outcomes, and any associated details.

Ride the wave of Modern Software Delivery

Have a question? We are here to help!

Source URL: <https://www.harness.io/products/continuous-error-tracking/cet-features>

Delivering Full Visibility into Software Errors

Connect your source code

Instant linkage of application errors to their corresponding source code, including instrumented code.

Streamlined access to code repository, branch, and commits for efficient code change tracing.

Enhanced developer collaboration and effective error routing to right developers.

Reproduce every issue

Simplified access to the complete call stack.

Navigation option to traverse the call stack, view linked code and variable data.

Code variables and objects 10 levels deep into the heap.

Recorded Variables at the time of the event provide rich context for reproducing the issue.

Access logs and machine data instantly

Enhanced by up to 250 log entries produced during the error event occurrence.

No Disk IO captured directly from the JVM

Access to CPU, Heap, and IO statistics at the moment of the event.

Thread data visibility for active, blocking, and terminated states.

Host / container environment state recorded.

Safeguard downstream environments with CI/CD integration

Early identification and resolution of errors in the CI stage to elevate developer productivity.

Block unreliable deployments from getting promoted to downstream environments.

Analyze errors trends across deployment versions

Augment your observability

Access exception data directly from your logs with log links.

Complete source code, stack, variables and environment details with a simple click from your logs

Get started with Continuous Error Tracking

Deliver more reliable software

Source URL: <https://www.harness.io/support/tiers-and-definitions>

Support Tiers and Definitions

Harness Support tiers

Harness has three easy-to-understand support tiers:

- Community - Any Harness user is a member of the Harness community, and as such is entitled to community support.
- Standard - All paying customers are entitled to standard support.
- Premier - Customers may, at their discretion, purchase a premier support package for a 20% premium of their subscription cost.

What is included in each support tier?

â

Harness Support standard coverage hours are 9 am to 5 pm Pacific standard time, during Harness business days.

Support escalation

For any support ticket escalation, if it remains outstanding even after escalating via ticket itself, please reach out to CX Leadership by mailing us at cx-escalation@harness.io.

Support issues priorities

Support issues response SLA is based on the issue priority. Harness uses these definitions for its priorities (Unless agreed upon differently in the Master Service agreement)

Support for Early Access(Beta) and Public Preview features

Beta features:

These features might include bugs or performance issues. There is a probability that these functionalities will not be included in the GA release and hence will not be covered under our Support SLA or Escalation entitlement.

Public Preview features:

Harness has limited support for Public Preview features and these are released behind a feature flag. We donât recommend using these features in Production environments and support for these is mostly delivered directly by the Harness Engineering team.

Security issues

Security issues may include reports of CVEs (Common Vulnerabilities and Exposures), data leaks, privacy concerns, and regulatory compliance. Before reporting these to Harness Support, you should review our security FAQs portal at: <https://trust.harness.io/>

Source URL: <https://www.harness.io/products/feature-flags/resources>

Resources

Tyler Technologies Takes CI/CD to the Next Level and Achieves Unparalleled Velocity With Feature Flags

Description text.

Featured Posts

Blog

eBook

Solution Brief

Case Study

On-Demand Webinar

Infographic

case study

Metrikus Shortens Commit-to-Production Time by 66%

case study

Feature Flags: Making Software Delivery Faster

news

Entur Improves Deployment Frequency from Twice a Week to 14 Times a Day, Rollback Time from Two Hours to Five Minutes

What Makes Harness a Leader and Outperformer in the GigaOm Radar Report for Feature Flags

blog

blog

6 Use Cases for Kill Switches in Production Using Feature Flags

blog

Why Using Feature Flags Is Compliant and Secure

case study

Metrikus Shortens Commit-to-Production Time by 66%

case study

Entur Improves Deployment Frequency from Twice a Week to 14 Times a Day, Rollback Time from Two Hours to Five Minutes

blog

What Makes Harness a Leader and Outperformer in the GigaOm Radar Report for Feature Flags

blog

6 Use Cases for Kill Switches in Production Using Feature Flags

blog

Why Using Feature Flags Is Compliant and Secure

blog

Common Feature Flags Security Questions

blog

Feature Flags Overview For PMs: Getting Qualitative Feedback and Managing Releases

blog

Thinking About Feature Flags As a Part of Your Culture

Try Smart Feature Flagging Today

Experience developer automation with GitOps

Source URL: <https://www.harness.io/products/feature-flags/features>

Automation With Smart Feature Flags

Manage Flags using Gitops Experience

Manage your flags from a YAML file in your Git repository

Synchronize your changes from Git or the Harness Platform

Build Powerful Pipelines

Approvals

Streamline feature flag approvals for faster deployment.

Scheduled

Schedule feature flag releases with precision and ease.

Notifications

Stay informed with real-time feature flag updates and alerts.

Progressive Delivery

Gradually roll out feature changes with smart delivery strategies.

Governance

Maintain control and governance over your feature flag operations.

Approvals

Streamline feature flag approvals for faster deployment.

Scheduled

Schedule feature flag releases with precision and ease.

Notifications

Stay informed with real-time feature flag updates and alerts.

Progressive Delivery

Gradually roll out feature changes with smart delivery strategies.

Governance

Maintain control and governance over your feature flag operations.

Release Management

Individual and Group Targets

Unlock the power of targeted rollouts and seamless release progression with feature flags. Take control of your software releases, test new features with select audiences, and gather valuable user feedback all while minimizing risks.

Performance and Security

Open Source High Scale Relay Proxy

Global CDN for Maximum Performance

Audit Logging

Secure Flag Change Controls

Programmatic Control with APIs

Manage Every Action with the API

Integrations

Seamless Workflow Integrations

Analytics and Reporting

Manage Features and Utilization

Lifecycle Management

Keep Your Codebase Organized

Try Smart Feature Flagging Today

Experience developer automation with GitOps

Source URL: <https://www.harness.io/products/chaos-engineering/features>

Confidently Deliver Software by

Proactively Building Resilience

Find out how your applications perform against real-life failures scenarios and learn how to build a more resilient system to reduce costly downtime

Discover how your applications stand up to real-world failure scenarios.

We have all you need

Supercharge your continuous resilience journey with powerful features on Security, Collaboration and Analytics

Resilience & Probes

Probes provide the ability to understand the experiment's impact on the system or other dependent systems and measure the steady state of your system during chaos experiments. Probes help give you feedback from the experiment, so you don't have to manually monitor or observe impacts on tested systems.

Get onboard to the resilience journey today

Discover how your applications stand up to real-world failure scenarios. Gain insights to construct a resilient system that minimizes downtime and saves on costs.

Source URL: <https://www.harness.io/products/continuous-delivery/deploy-anywhere>

Deploy Anywhere

Unparalleled deployment versatility with Harness Continuous Delivery! Whether you're deploying to cloud native, on-premises, serverless, hybrid infrastructures, or even edge environments, Harness Continuous Delivery has you covered.

Out of the Box Deployment Strategies

Harness offers versatility in progressive deployment strategies mitigating risk.

Leverage the Canary Strategy to put a small amount of traffic on the new version, rolling out widely later. Or use Blue Green to quickly swap versions with near instantaneous rollback.

Deployment strategies are target-aware - Harness understands that managing traffic for a Canary Strategy is different for an Azure Web App than in Kubernetes.

Custom Deployment Types

Harness gives you the ability to define custom deployment types for any type of use case or application. If your application needs further customization, Harness can still support the orchestration and deployment of your application. Harness helps with monitoring and visibility of custom applications. Users can also bring in their own plugins to extend Harness deployment functionality to cater to your needs. It's as simple as bringing your own container and running it as a step in the pipeline!

Streamline Configuration with Reuse

Discover the transformative power of our template library, designed to infuse uniformity and precision into every pipeline. Our curated collection showcases templates meticulously crafted to cater to a myriad of deployment scenarios and environments. With our intuitive interface, developers can seamlessly author, select, refine, and roll out pipelines based on these templates. The outcome is consistent deployment pipelines that champion best practices and dramatically reduce errors.

Our templates are backed by robust version control, empowering you to experiment and iterate with utmost confidence, all while ensuring pipelines which consume a template remain unaffected.

Provision Infrastructure on demand

Deploy to ephemeral environments, or use your CD pipelines to provision long-lived environments. With tight integrations to infrastructure-as-code (IaC) technologies, including AWS Cloud Formation, Harness Infrastructure as Code Management, Terraform, Terragrunt, Terraform Cloud, Azure Blueprints and more; Harness CD connects application and infrastructure change like never before.

Ride the wave of Modern Software Delivery

Have a question? We are here to help!

Source URL: <https://www.harness.io/events/devops-dining-club-4>

DevOps Dining Club

Time: 6:30 PM until 10:00 PM

Welcome to the DevOps Dining Club, a series of networking events for technology leaders who seek to share and learn with each other in world-class restaurants.

Our next event will take place at Hawksmoor Borough, London on the 24th of January. Join us to discuss challenges, trends and opportunities across the Software Development Lifecycle (SDLC) and explore best practices in areas such as Developer Experience, Cloud-Native Technologies, Security and Compliance, and Automation and AI.

Event details:

Date: Wednesday, 24th January

Time: 6:30 PM - 10:00 PM

Venue: Hawksmoor Borough, London

Space at this event is limited. To secure your place please request a ticket as soon as possible.

Please note: this event will operate under the Chatham House Rule and is intended for technology leaders managing large-scale software engineering teams and projects. Harness reserves the right to refuse entry and at the request of regular attendees, we will be striving to maintain the peer-to-peer nature of attendee profiles.

Source URL: <https://www.harness.io/products/continuous-delivery/devops-pipeline-governance>

DevOps Pipeline Governance

To go fast safely, you need sharp steering and good brakes. Harness empowers developers while satisfying governance, security, and compliance teams.Â

Policy as Code

Harness Policy as Code is a centralized policy management and rules service that leverages the Open Policy Agent (OPA) to meet compliance requirements across software delivery and enforce governance policies. Policies are written as declarative code, so they are easy to understand and modify, enabling teams to have autonomy over their processes with oversight and guardrails in place to prevent them from straying from standards.

Role-Based Access Control

Harness provides fine-grained RBAC (role-based access control) to enforce separation of duties and control what user groups are granted access to specific resources based on assigned roles. This allows businesses to protect their data and key business processes through company-set rules and roles.

Built-in roles are available by default to quickly create the desired permissions at the account, organization, and project level within Harness, as well as the ability to create custom roles for additional flexibility based on business needs that fall outside of the scope provided by default roles.

Audit Trail

Harness Audit Trails provide the visibility needed to meet organizational governance needs and prepare for external audits. With Harness Audit Trails, you can view and track changes to your Harness resources within your Harness account with data stored from up to two years prior. Without this data, developers are forced to manually compile information for audits.

Log Sanitization

Harness scrubs deployment logs and any script outputs to mask text secret values. For text and file secrets, the secrets are stored in the Secrets Manager you select. When a text secret is displayed in a deployment log, Harness substitutes the text secret value with asterisks (*) so that the secret value is never displayed.

Ride the wave of Modern Software Delivery

Have a question? We are here to help!

Source URL: <https://www.harness.io/support/support-faqs>

Support FAQs

Effective support request

Want to know what is the best way to report an issue, or ask a question? Follow these guidelines.

Issues, problems, or bugs

When reporting an issue, it is best to include the following components:

Feature requests

If reporting a request for new functionality, make sure to include:

Questions

If asking a question, please tell us:

What happens when a ticket is created?

- Once the Ticket has been created by a user, it will show up in the Agent queue on our end which is what the Support Engineers use for Communication and further updates in order to resolve the issue.
- Harness support engineers are actively working on the incoming issues 24x7 365 days a year as covered in our SLA based on support tier.

Ticket Routing Flow

- Every ticket created with us has an associated SLA according to assigned priority, which ensures you get a response from the support engineers in a stipulated amount of time.
- When the Engineer replies to your ticket and puts the status into Pending, it means we are awaiting a reply from you with an action item to move forward with the issue resolution. In case of unresponsiveness on raised issues, they are auto-solved based on criteria defined under the below section âAutomatic close of pending issuesâ.
- For any Product bugs/defects, we work very closely with our Engineering team to get them fixed as soon as possible. Anything defined as a Production or Adoption blocker does get priority and is eligible for immediate hotfix depending upon the business impact associated with that issue.
- Specifically for High and Urgent priority issues, the support team works more closely with Product Leads to ensure prompt responses leading toward resolution.
- For any Feature Request reaching the support queue, they are prioritized by our Product Management team along with your account Customer Success Manager(CSM). Kindly reach out to your CSM in order to prioritize such requests if they are adoption/onboarding blockers.

CSAT survey

At Harness, we believe in continuous improvement and hence every interaction with our users is an opportunity for us to further improve our services and offerings. Therefore as soon as a ticket is Solved an automatic survey request email is sent to the requester to capture the feedback on the support experience offered.

Ticket merge

To prevent duplication of effort, if two people in the same organization report the same issue, we will merge the two into one. A notice will be sent to all parties about the merge. Both parties will be copied on the merged ticket.

Automatic close of pending issues

A support ticket is considered pending if it is waiting for a response from the ticket originator (the requester). Harness will send an automatic reminder two and four business days after the ticket is set to Pending. After five business days, if no response has been received, a final message will be sent, and the ticket will be marked solved, regardless of status.Â

The requester may re-open a solved ticket within two weeks. After that time, a ticket is marked closed and cannot be reopened.

Code of conduct

Harness is committed to treating all customers, prospects, partners, and users with respect, empathy, and kindness, as outlined in the Harness Support Code Of Conduct.

The same is expected from anyone requesting support. Misconduct toward Harness employees and support staff may result in support tickets being closed, a ban on a specific individual from future support requests, or in the case of repeat offenders, termination of the business relationship with Harness.

Sharing files with Harness Support

Sometimes sharing files with support is needed.

Acceptable file formats

To prevent a support request from being flagged as malicious, do not share executable files (*.exe, *.cmd, *.bat, etc). If you need to share a code sample, save the file with the *.txt extension before sharing.

Large files

When communicating with Harness Support, we limit attachment sizes to 7 MB per individual file and 10 MB for the entire message.

To share files exceeding the above limit, you may use our drop zone (provided by SendSafely) at harness.sendsafely.com/dropzone/support.

Encrypted files

Do not send encrypted files or messages as Harness may not have access to decryption tools, and will not be able to respond to your message.

Sanitization

Make sure to not send credentials, passwords, keys, personal identifying information, secrets, tokens, or other secure content in clear text or image. Obfuscate these, or delete them all together before sending them to support.

Do not share passwords with support. We will not ask for it.

Self-solving of support tickets

If you have found our solution helpful, or figured it out yourself, please let us know that we can close the support ticket.Â

You can do this yourself by logging into the support portal and marking the ticket as solved.Â

Please note that only the ticket's main contact (the requester) may close the ticket. You can not solve tickets you are copied on.

Copying others on support tickets

You may add others to your support ticket as CC. This means that they will be copied on all messages to/from Harness Support and yourself.

If you are using email, just copy the new person on the email message, and they will be copied on all future communication.

If you are using the support portal, you can mark others as copied. You can also ask your support engineer to add others as copied.

Source URL: <https://www.harness.io/products/infrastructure-as-code-management/features>

Empowering Infrastructure Teams with Seamless Management.

Pipeline for Infrastructure Changes

CICD for Infrastructure

Harness IaCM offers a cutting-edge and sophisticated CI/CD solution tailored to infrastructure changes.

Integrated Å Pipeline

Seamlessly incorporate into our comprehensive pipeline platform, enabling streamlined automation and orchestration of infrastructure provisioning and updates.

State management

Harness provides an out-of-the-box hosted backend, eliminating the need to host state and manage locking mechanism and access control.

Drift detection

Harness IaCM provides automated drift detection and reconciliation, preventing discrepancies between desired and actual state, ensuring git is the single source of truth for infrastructure changes.

Cost estimation

Harness IaCM helps users proactively identify cost impact associated with resource changes, which prevents unplanned and exorbitant cloud bills.

PR automation

By implementing PR automation, developers significantly reduce the likelihood of errors when making resource changes. This capability brings the visibility of planned changes into the PR process.

COMING SOON

Reports and insights

Harness IaCM provide actionable insights into the managed resources, including input on activities, status, and governance.

COMING SOON

Governance

Harness IaCM provides a comprehensive set of functionality that helps to govern the provisioning process, including granular access control, audit trail, and OPA policies.

Roadmap & Vision

Roadmap & Vision

Continuous Delivery Integration

Harness CD customers will benefit from seamless integration between IaCM and CD, providing resource visibility in Services and Environments.

Cloud Costs Management Integration

Integration with Harness CCM will give users more cost insight into the infrastructure and better cost control and recommendations earlier in the provisioning process.

Module Registry

Harness IaCM will provide a private Module Registry for Terraform modules, making it easier to publish approved modules within the organization.

Tool Agnostic

Harness IaCM will expand its support to other IaC tools such as Terragrunt, Crossplane, Pulumi, CDK, AWS CloudFormation, and others, providing one platform for all IaC technologies.

Manage your infrastructure as code.

End-to-end.

Covering all your Infrastructure as Code Management needs.

Book a Demo

Manage your infrastructure as code. End-to-end.

Deploy faster with better reliability

Book a Demo

Source URL: <https://www.harness.io/legal/>

Subscription Terms

As of Sep 21, 2023

These Harness Subscription Terms (collectively, the "Agreement") between Harness Inc., a Delaware corporation, with its principal place of business at 55 Stockton St., 8th Floor, San Francisco, CA 94108, U.S.A. ("Harness", "we", "us" or "our") and you ("Customer", "you" or "your") applies to your use of the Harness Platform (as defined below). By clicking on the designated button, entering an Order Form (as defined below), or by downloading, installing, accessing or using the Harness Platform, you agree to the terms of this Agreement. If you are entering into this Agreement on behalf of your organization or entity, you represent that you have the authority to bind such organization or entity to the Agreement, and the terms "Customer", "you" and "your" will refer to such organization or entity. If you do not agree to the terms of this Agreement, or if you are not authorized to accept this Agreement on behalf of your organization or entity, do not download, install, access or use the Harness Platform. "Harness Platform" means the downloadable and/or online software products that are specified in the applicable Order Form, or otherwise accessed by you, and subsequent updates made generally available by Harness under this Agreement, inclusive of the Delegate. The Harness Platform excludes Third Party Products, Extensions, and Beta Services.

1. Harness Platform

1.1. License Grant. Subject to payment of the applicable fees, Harness's receipt of a purchase order number from Customer (if needed), and Customer's ongoing compliance with the terms of this Agreement, Harness grants to Customer a limited, non-exclusive, non-transferable, non-sublicensable right and license to access and use the Software for internal business purposes in accordance with the Documentation (as defined below) during the applicable License Term, only for the number of License Units (as defined below) as specifically authorized by the applicable Order Form. "License Term" means the duration of your subscription or license to the applicable Harness Platform beginning and ending on the start and end dates, respectively, specified in the applicable Order Form, which include the Initial Term (as defined in Section 5), and all Renewal Terms (as defined in Section 5), as applicable. "License Unit" means a specific license type or metric, and a numeric quantity thereof, identified in an Order Form, to establish the extent and amount of Customer's license or right to use to the Harness Platform. All Order Forms are hereby incorporated into, supplement, and form a part of this Agreement. "Order Form" means an ordering document or online order that sets forth the applicable Harness Platform or Services (as defined below), fees and payment terms and start and end dates (as applicable) and is entered into between Customer and Harness, or Customer and an authorized reseller.

1.2. Restrictions on Use. Except as otherwise expressly provided in this Agreement, Customer shall not (and shall not permit any third party to): (a) sublicense, sell, resell, transfer, assign, distribute, share, lease, rent, make any external commercial use of, outsource, use on a timeshare or service bureau basis, or use in an application service provider or managed service provider environment, or otherwise generate income from the Harness Platform or Extensions; (b) copy the Harness Platform or Extensions onto any public or distributed network, except for an internal and secure cloud computing environment; (c) cause or permit the decompiling, disassembly, or reverse engineering of any portion of the Harness Platform or Extensions, or attempt to discover any source code or underlying algorithms or other operational mechanisms of the Harness Platform or Extensions (except where such restriction is expressly prohibited by law without the possibility of waiver, and then only upon prior written notice to Harness); (d) modify, adapt, translate or create derivative works based on all or any part of the Harness Platform or Extensions; (e) violate the Acceptable Use Policy ("AUP") located at <https://harness.io/legal/aup>; or (f) use free or trial accounts for certain products after having purchased License Units for the same products; (g) attempt to probe, scan or test the vulnerability of the Harness Platform (excluding the Delegate); (h) breach the security or authentication measures of the Harness Platform without proper authorization or willfully render any part of the Harness Platform or Extensions unusable; or (i) otherwise use the Harness Platform in violation of applicable law (including any export law) or outside the scope expressly permitted hereunder and in the applicable Order Form. Additionally, Customer shall not export or re-export, directly or indirectly, any Harness Platform or technical data or any copy, portions or direct product thereof (i) in violation of any applicable laws and regulations, (ii) to any country for which the United States or any other government, or any agency thereof, at the time of export requires an export license or other governmental approval, including Cuba, Iran, North Korea, Syria, the Crimea region of Ukraine, and

the so-called Donetsk Peopleâs Republic, and Luhansk Peopleâs Republic regions of Ukraine, or any other Group D:1 or E:2 country (or to a national or resident thereof) specified in the then current Supplement No. 1 to part 740 of the U.S. Export Administration Regulations (or any successor supplement or regulations, without first obtaining such license or approval) or (ii) to anyone on the U.S. Treasury Departmentâs list of Specially Designated Nationals or the U.S. Commerce Departmentâs Table of Denial Orders. Customer shall, at its own expense, obtain all necessary customs, import, or other governmental authorizations and approvals.

âDelegateâ means the software agent provided by Harness to Customer which facilitates Customerâs use of the Harness Platform. For the purposes of this Agreement, the Delegate is not considered an Extension.

1.3. Free Access. If the Harness Platform is provided to Customer on a limited trial, free, or beta basis (âFree Accessâ), Customer agrees that such use and access of the Harness Platform is governed by this Agreement. Harness shall have the right to downgrade, limit or otherwise modify the Harness Platform provided for a Free Access account at any time without notice to Customer, and Harness disclaims all liability arising from Free Access, and shall have no liability, nor warranty, indemnity, maintenance, support, or security obligations to Customer with respect to any such Free Access. Customer may only use the number and type of License Units for the specified duration indicated by Harness prior to Customer downloading or accessing the Harness Platform with respect to any such Free Access. Harness may immediately revoke and terminate any Free Access at any time and without any notice to Customer. Customer agrees to provide feedback related to the Harness Platform as reasonably requested by Harness with respect to any Free Access. Customer agrees that Free Access is not a guarantee of any future Harness Platform or Harness product features. Customer will not utilize a Free Access account for the same products it has purchased.

1.4. Unauthorized Use. Customer shall notify Harness promptly of any unauthorized use or access of the Harness Platform (including unauthorized users or unauthorized disclosure of any password or account), or any other known or suspected breach of security or misuse of the Harness Platform. Customer is responsible for use of the Harness Platform (and all other acts or omissions) by its employees, contractors, Affiliates or other users that it allows to use or access the Harness Platform.

1.5. Support. During the License Term, Harness shall provide support to Customer in accordance with Harnessâs then-current support policy, and as identified in an Order Form. In the event that the level of support is not identified in the Order Form, Customer shall receive a âstandardâ level of support that is included with the Harness Platform at no additional cost. For any support tier above standard support, the applicable support fees will be a percentage of all of Customerâs Harness Platform-based fees, and will be prorated for mid-year expansions based on the remaining months in the then current Initial Term or Renewal Term. Further, Customer agrees to facilitate any connections and access necessary for Harness to (i) deliver, deploy and provide the Harness Platform as provided hereunder and (ii) to perform its obligations hereunder (including any support obligations). Notwithstanding anything to the contrary in this Agreement, Harness has no warranty, indemnity or other obligation or liability with respect to modifications made to the Harness Platform or Documentation (as defined below) by Customer or on Customerâs behalf other than the generally available updates provided by Harness.

1.6. Purchasing Through Authorized Resellers. If you purchase a subscription to the Harness Platform or any Services through a Harness authorized reseller (âPartnerâ), this Agreement and any agreed upon usage limitations will govern the use of such Harness Platform and Services unless otherwise agreed by Harness and Customer. You also agree that Harness is an express third party beneficiary of your agreement with any authorized reseller. Your payment obligations for the Harness Platform and Services will be with the authorized reseller as further described in Section 2.6 below, not Harness, and you will have no direct fee payment obligations to Harness, provided that Harness may terminate this Agreement if you breach any of your payment obligations to such authorized reseller for the Harness Platform and Services. Any terms agreed to between you and the authorized reseller that are in addition to or inconsistent with this Agreement are solely between you and the authorized reseller. No agreement between you and an authorized reseller is binding on Harness, nor will it have any force or effect with respect to the rights in, or the operation, use or provision of, the Harness Platform or Services.

1.7. Contractors and Third Party Providers. You may permit your authorized consultants, contractors, and agents (âThird Party Providersâ) to access and use the Harness Platform, but only on your behalf in connection with providing the Harness Platform to you, and subject to the terms and conditions of this Agreement. Any access or use by a Third Party Provider will be subject to the same limitations and restrictions that apply to you under this Agreement, and you will be responsible for any Third Party Providerâs actions or omissions relating to its use of the Harness Platform. The aggregate use by you and all of your Third Party Providers must not exceed the allotted License Units (without paying the overage fees set forth in Section 2.2), and nothing in this Section is intended to or will be deemed to increase such License Units.

1.8. Services. Harness will use commercially reasonable efforts to provide the Services as described in an applicable Order Form (or statement of work referencing this Agreement entered into between the parties (âSOWâ)), if any. Harness will retain all right, title and interest in and to the deliverables and other results of the Services under this Agreement, and, subject to payment of the applicable fees and compliance with the terms of this Agreement, Harness hereby grants to Customer a limited, non-exclusive, non-transferable, non-sublicensable right and license to use such deliverables and results solely for Customerâs internal business purposes and only in connection with Customerâs permitted use of the Harness Platform. The Services (and any deliverables or other results) are not subject to any acceptance procedure. âServicesâ mean any training, enablement, consulting, installation and/or other professional services described in the applicable Order Form or SOW.

1.9. Customer Affiliates. Customer Affiliates may purchase and use the Harness Platform and Services subject to the terms of this Agreement by executing Order Forms or SOWs hereunder that incorporate by reference the terms of this Agreement. In each such case, all references in this Agreement to Customer shall be deemed to refer to such Customer Affiliate for purposes of such Order Form(s) or SOW(s), and Customer Affiliate agrees to be bound by this Agreement. âAffiliateâ means, with respect to Harness or Customer, any entity that directly or indirectly controls, is controlled by, or is under common control with Harness or Customer, respectively. âControlâ for purposes of this definition, means direct or indirect ownership or control of more than 50% of the voting interests of the subject entity.

1.10 Security. Harness will establish and maintain appropriate administrative, technical, and physical safeguards and controls to: (i) help

ensure the ongoing confidentiality, integrity, availability, and resiliency of the Harness Platform and Customer Data, and (ii) have in place a process for regularly testing, assessing and evaluating the effectiveness of technical and organizational measures to help ensure the security of the Harness Platform's processing.

1.11 Extensions. Customer may use the Extensions solely in connection with the applicable Harness Platform subject to the Documentation, open source licenses, the applicable terms within this Agreement (including with respect to the Term), and the payment of any Fees associated with the Extensions. "Extension" means any separately downloadable or accessible suite, agent, configuration file, add-on, technical add-on, plug-in, example module, command, function, playbook, content or application that enables or extends the features or functionality of the applicable Harness Platform.

1.12 Third Party Products. This Agreement does not govern Customer's use of Third Party Products used in connection with the Harness Platform. Third Party Products are governed solely by the terms and conditions between Customer and the Third Party Product developer. Harness does not make any commitments or claims regarding security, confidentiality, or performance of any Third Party Products, and specifically disclaims any liability regarding Third Party Products. Customer acknowledges and accepts that Third Party Products: (i) are activated and used at the sole risk of Customer; (ii) are not warranted, supported, or endorsed by Harness; and (iii) may degrade the performance of the Harness Platform beyond Harness's reasonable control. To the extent any Third Party Product accesses, processes, or gathers personal data, the applicable third party is Customer's direct data processor, and is not acting as a data sub-processor of Harness. "Third Party Product(s)" means any product, software, application, platform, or service (i) selected by Customer (ii) not developed by Harness, and (iii) which integrates, interacts, or interoperates with, or adds functionality to, the Harness Platform.

2. Fees

2.1. Fees. You agree to pay all fees specified in the Order Form and/or SOW, or as otherwise agreed upon by the parties. Fees are non-cancelable, non-refundable, and due and payable within thirty (30) days from the date of the invoice, or as otherwise specified in the Order Form. Unless otherwise agreed to by the parties in writing, all fees hereunder are payable in United States dollars. Additionally, Customer must provide all purchase order numbers (if applicable) to Harness by no later than the applicable Start Date as specified in the Order Form. All payments shall be made through automated clearing house (ACH) transfers, or wire transfers, to Harness's designated account, unless otherwise agreed by Harness. Fees do not include any customizations of the Harness Platform (nor support for any such customizations, unless otherwise agreed in writing).

2.2. Excess Usage. If Customer's use of the Harness Platform exceeds the number of License Units set forth in the Order Form, Customer will be billed for and Customer will pay those overages at a prorated amount for the remainder of the applicable License Term under the Order Form, based on the pricing specified in the applicable Order Form. If Harness believes in good faith that Customer's use of the Harness Platform exceeds the number of License Units set forth on the Order Form, Harness may (i) audit Customer's use of the Harness Platform (not more frequently than twice per calendar year), upon at least twenty-four (24) hours' notice, and (ii) require that Customer provide Harness with all relevant records within five (5) business days of such request in order to determine if Customer's use of the Harness Platform exceeds the number of authorized License Units.

2.3. Payment Terms. Customer acknowledges that purchases made under this Agreement are neither contingent on the delivery of any future functionality or features of the Harness Platform nor dependent on any oral or written public comments made by Harness regarding future functionality or features of the Harness Platform. All fees shall be fixed during the Initial Term (as defined in Section 5) unless Customer purchases additional License Units. If the number of purchased License Units does not increase upon renewal, then all Harness Platform and support fees will increase by the percentage specified in the Order Form at the start of each applicable Renewal Term (as defined in Section 5). If Customer is overdue on any payment, then Harness may (i) require that Customer pay a late fee equal to the lesser of 1.5% of the then-outstanding unpaid balance per month or the maximum amount allowable by law, and/or (ii) suspend Customer's use of and access to the Harness Platform and/or Services associated with Customer's account until such non-payment is corrected. Customer represents and warrants that the billing and contact information provided to Harness is complete and accurate, and Harness shall have no responsibility for any invoices that are not received due to inaccurate or missing information provided by Customer. All amounts invoiced under this Agreement and any Order Form shall be paid by Customer in full without any set-off, counterclaim, deduction, or withholding (excluding any tax withholding deductions as required by law).

2.4 Credit Cards. If Harness authorizes you to pay by credit or debit card in writing, you: (i) will provide Harness or its designated third-party payment processor with valid credit or debit card information; and (ii) hereby authorize Harness or its designated third-party payment processor to charge such credit or debit card for all items listed in the applicable Order Form or SOW or as otherwise agreed by the parties. Such charges must be paid in advance or in accordance with any different billing frequency stated in the applicable Order Form or SOW (if applicable). You are responsible for providing complete and accurate billing and contact information and notifying Harness in a timely manner of any changes to such information.

2.5 Taxes. The fees paid by Customer are exclusive of all taxes, levies, or duties ("Taxes") imposed by taxing authorities, if any, and Customer shall be responsible for payment of all such Taxes, excluding taxes based on Harness's income. Customer represents and warrants that the billing and contact information provided to Harness is complete and accurate, and Harness shall have no responsibility for any invoices that are not received due to inaccurate or missing information provided by Customer.

2.6 Payment Through Partner. Notwithstanding anything herein to the contrary, if Customer has licensed the Harness Platform from a Harness Partner, then Customer shall make its payments for the Harness Platform directly to such Partner, and the payment terms agreed by Customer and such Partner shall supersede and govern to the extent anything in this Section 2 conflicts with such payment terms.

3. Confidentiality

3.1. Confidential Information and Restrictions. "Confidential Information" means all information disclosed by a party ("Disclosing Party") to the other party ("Receiving Party"), whether orally or in writing, that is designated as "confidential" or "proprietary," or that, given the nature of the information or circumstances surrounding its disclosure, should reasonably be understood to be confidential.

âConfidential Informationâ does not include any information that: (i) is or becomes generally known to the public without breach of any obligation owed to the Disclosing Party, (ii) was known to the Receiving Party prior to its disclosure by the Disclosing Party without breach of any obligation owed to the Disclosing Party, (iii) is received from a third party without breach of any obligation owed to the Disclosing Party, or (iv) was independently developed by the Receiving Party. The Receiving Party will: (i) not use the Disclosing Partyâs Confidential Information for any purpose outside of this Agreement; (ii) not disclose such Confidential Information to any person or entity, other than its Affiliates, employees, consultants, subcontractors, subprocessors, agents and professional advisers (âRepresentativesâ) who have a âneed to knowâ for the Receiving Party to exercise its rights or perform its obligations hereunder, provided that such employees, consultants, and agents are bound by agreements or, in the case of professional advisers, ethical duties respecting such Confidential Information in accordance with the terms of this Section 3; and (iii) use reasonable measures to protect the confidentiality of such Confidential Information. The Receiving Party shall be liable for any breach of this section by its Representatives. If the Receiving Party is required by applicable law or court order to make any disclosure of such Confidential Information, it will first give written notice of such requirement to the Disclosing Party, and, to the extent within its control, permit the Disclosing Party to intervene in any relevant proceedings to protect its interests in its Confidential Information, and provide full cooperation to the Disclosing Party in seeking to obtain such protection. Further, this Section 3 will not apply to information that the Receiving Party can document: (i) was rightfully in its possession or known to it prior to receipt without any restriction on its disclosure; (ii) is or has become public knowledge or publicly available through no fault of the Receiving Party; (iii) is rightfully obtained by the Receiving Party from a third party without breach of any confidentiality obligation; or (iv) is independently developed by employees of the Receiving Party who had no access to such information.

3.2. Equitable Relief. The Receiving Party acknowledges that unauthorized disclosure of the Disclosing Partyâs Confidential Information could cause substantial harm to the Disclosing Party for which damages alone might not be a sufficient remedy and, therefore, that upon any such disclosure by the Receiving Party the Disclosing Party will be entitled to seek appropriate equitable relief in addition to whatever other remedies it might have at law or equity.

3.3. Feedback. Customer acknowledges and agrees that (a) any questions, comments, suggestions, ideas, feedback or other information about Harness, the Harness Platform, Extensions, the Services, the Documentation or other materials provided by Harness (collectively, âFeedbackâ) provided by Customer are non-confidential, (b) Harness will have full discretion to determine whether or not to proceed with the development of any requested enhancements, new features or functionality, and (c) Harness will have the full, unencumbered right, without any obligation to compensate or reimburse Customer, to use, incorporate and otherwise fully exercise and exploit any such Feedback in connection with its products and services.

4. Proprietary Rights

4.1 Ownership. Harness owns and shall retain all proprietary rights, including all copyright, patent, trade secret, trademark and all other intellectual property rights, in and to the Harness Platform (and all derivatives, improvements or enhancements thereof), System Data, Delegate, Extensions, Documentation, the results generated by the Harness Platform, and any Services, or any materials generated by Harness.Â

4.2 FOSS; Third Party Components. Customer acknowledges that the rights granted under this Agreement do not provide Customer with title to or ownership of the Harness Platform, in whole or in part. Certain âfreeâ or âopen sourceâ based software (the âFOSS Softwareâ) and third party software included with the Harness Platform (the âThird Party Softwareâ) is shipped with the Harness Platform but is not considered part of the Harness Platform hereunder. Use, reproduction, and distribution of FOSS Software is governed by the terms of the applicable open source software license and not this Agreement. Harness will provide Customer with a list of the FOSS Software and Third Party Software embedded in the Harness Platform upon request. With respect to Third Party Software included with the Harness Platform, such Third Party Software suppliers are third party beneficiaries of this Agreement. The Harness Platform and Third Party Software may only be used and accessed by Customer as prescribed by the instructions, code samples, on-line help files and technical documentation made publicly available by Harness for the Harness Platform, as may be updated from time to time by Harness (the âDocumentationâ). Harness will not be responsible for any act or omission of any third party, including the third partyâs access to or use of any Customer data or the performance of the Harness Platform in combination with such Third Party Software.

5. Term and Termination

This Agreement will be in full force and effect beginning on the earlier of the start date of the first Order Form entered into hereunder and the date you first access or otherwise use the Harness Platform, and will remain in effect until this Agreement is terminated pursuant to this Section. Termination of a specific Order Form will not affect the effectiveness of this Agreement or any other Order Form. Unless indicated otherwise in an Order Form, each Order Form shall be valid from the earliest start date therein through the initial end date therein (the âInitial Termâ), and shall automatically renew for additional successive twelve (12) month terms (each, a âRenewal Termâ), unless either party provides notice of non-renewal no less than thirty (30) days prior to the end of then-current Initial Term or Renewal Term, as applicable. If either party commits a material breach of this Agreement, and such breach has not been cured within thirty (30) days after receipt of written notice thereof, the non-breaching party may terminate this Agreement, except that Harness may immediately terminate this Agreement and/or terminate or suspend Customerâs use of and access to the Harness Platform associated with Customerâs account upon Customerâs breach of Section 1.2 (Restrictions on Use) or Section 2.1 (Fees). Additionally, Harness may temporarily suspend access to the Harness Platform if Customerâs use poses a security risk or adversely impacts Harnessâs business. Either party may also terminate this Agreement upon written notice if (a) the other party suspends payment of its debts or experiences any other insolvency or bankruptcy-type event or (b) there are no Order Forms or SOWs then in effect. Upon expiration or termination of an Order Form, for any reason, all rights granted to Customer with respect to such Order Form shall terminate and Customer shall destroy any copies of the Harness Platform and Documentation provided under such Order Form within Customerâs possession and control. Upon any termination of this Agreement, each Receiving Party will return or destroy, at the Disclosing Partyâs option, the Disclosing Partyâs Confidential Information in the Receiving Partyâs possession or control. All payment obligations that have accrued as of such expiration or termination, any other rights or obligations that by their nature should survive, along with Sections 1.2, 1.3, 1.4, 1.6, 1.7, 2, 3, 4, 5, 6.2 and 7 through 10, will survive any expiration or termination hereof.

6. Warranties

6.1. Harness Platform Warranty. Harness warrants that during the first thirty (30) days after the beginning of a License Term under the applicable Order Form, the Harness Platform will, in all material respects, conform to the functionality described in the then-current Documentation for the applicable version of the Harness Platform. Harness's sole and exclusive obligation, and Customer's sole and exclusive remedy, for a breach of this warranty shall be that Harness will use commercially reasonable efforts to repair or replace the Harness Platform to conform in all material respects to the Documentation, and if Harness is unable to materially restore such functionality within thirty (30) days from the date of written notice of breach of this warranty by Customer, Customer shall be entitled to terminate the applicable Order Form upon written notice to Harness, and Harness shall promptly provide a pro-rata refund of the subscription fees under such Order Form that have been paid in advance for the remainder of the License Term under such Order Form (beginning on the date of termination). To be eligible for the foregoing remedy, Customer must notify Harness in writing of any warranty breaches within such warranty period, and Customer must have installed (if applicable), used and configured the Harness Platform in accordance with this Agreement and the Documentation.

6.2. Disclaimer. EXCEPT AS EXPRESSLY PROVIDED IN THIS SECTION 6, THE HARNESS PLATFORM, DOCUMENTATION, SERVICES, MAINTENANCE AND SUPPORT ARE PROVIDED "AS IS," AND HARNESS AND ITS SUPPLIERS EXPRESSLY DISCLAIM ANY AND ALL OTHER REPRESENTATIONS AND WARRANTIES, EITHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT THERETO, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, NON-INFRINGEMENT, OR THE CONTINUOUS, UNINTERRUPTED, ERROR-FREE, VIRUS-FREE, OR SECURE ACCESS TO OR OPERATION OF THE HARNESS PLATFORM. HARNESS EXPRESSLY DISCLAIMS ANY WARRANTY AS TO THE ACCURACY OR COMPLETENESS OF ANY INFORMATION OR DATA ACCESSED OR USED IN CONNECTION WITH THE HARNESS PLATFORM, DOCUMENTATION, SERVICES, MAINTENANCE OR SUPPORT. Additionally, Harness is not responsible for any delays, delivery failures, or any other loss or damage resulting from the transfer of data over communications networks and facilities, including the Internet, and Customer acknowledges that the Harness Platform, Delegate, Extensions, Services and Documentation may be subject to limitations, delays and other problems inherent in the use of such communications facilities. The Harness Platform is not fault-tolerant and is not designed or intended for use in hazardous environments, including without limitation, in the operation of aircraft or other modes of human mass transportation, nuclear or chemical facilities, life support systems, implantable medical equipment, motor vehicles or weaponry systems, or any other application in which failure of the Harness Platform could lead to death or serious bodily injury of a person, or to severe physical or environmental damage (each, a "High Risk Use"). Harness expressly disclaims any express or implied warranty or representation of fitness for High Risk Use. Harness shall not be liable to Customer for any loss, damage or harm suffered by Customer that is directly or indirectly caused by Customer's unauthorized use of the Harness Platform to process Prohibited Data. Prohibited Data means: (a) special categories of data enumerated in European Union Regulation 2016/679, Article 9(1) or any successor legislation; (b) patient, medical, or other protected health information regulated by the Health Insurance Portability and Accountability Act (as amended and supplemented) ("HIPAA"); (c) credit, debit, or other payment card data or financial account information, including bank account numbers or other personally identifiable financial information; (d) social security numbers, driver's license numbers, or other government identification numbers; (e) other information subject to regulation or protection under specific laws such as the Children's Online Privacy Protection Act or Gramm-Leach-Bliley Act ("GLBA") (or related rules or regulations); or (f) any data similar to the above protected under foreign or domestic laws.

6.3. Mutual Warranty. Each party hereby represents and warrants to the other that: (a) such party has the right, power, and authority to enter into this Agreement and to fully perform all of its obligations hereunder, (b) entering into this Agreement does not and will not violate any agreement or obligation existing between such party and any third party, and (c) this Agreement, when executed and delivered, will constitute a valid and binding obligation of such party and will be enforceable against such party in accordance with its terms.

6.4. Beta Software. From time to time, Customer may have the option to participate in a program with Harness where Customer is given access to alpha or beta software, services, products, features and documentation (collectively, "Beta Software") offered by Harness. The Beta Software is not generally available and may contain bugs, errors, defects or harmful components. Accordingly, Harness is providing the Beta Software to Customer "as is." Notwithstanding anything to the contrary in this Agreement, Harness makes no warranties of any kind with respect to the Beta Software, whether express, implied, statutory or otherwise, including any implied warranties of merchantability, fitness for a particular purpose, or non-infringement, and has no indemnity or other obligation or liability with respect to Beta Software. Harness does not warrant that the Beta Software will meet any specified service level, or will operate without interruptions or downtime.

7. Indemnification

7.1. By Harness. Harness agrees to defend, at its expense, Customer against (or, at Harness's sole option, settle) any third party claim to the extent such claim alleges that the Harness Platform infringes or misappropriates any patent, copyright, trademark or trade secret of a third party, and Harness shall pay all costs and damages finally awarded against Customer by a court of competent jurisdiction as a result of any such claim.

7.2. Remedies. In the event that the use of the Harness Platform is, or in Harness's sole opinion is likely to become, subject to such a claim, Harness, at its option and expense, may (a) replace the applicable Harness Platform with functionally equivalent non-infringing technology, (b) obtain a license for Customer's continued use of the applicable Harness Platform, or (c) terminate the applicable Order Form and provide a pro-rata refund of the subscription fees under such Order Form that have been paid in advance for the remainder of the License Term under such Order Form (beginning on the date of termination).

7.3. Limitations. The foregoing indemnification obligation of Harness will not apply: (1) if the Harness Platform is or has been modified by Customer or its agent; (2) if the Harness Platform is combined with other non-Harness products, applications, or processes, but solely to the extent the alleged infringement is caused by such combination; (3) to any unauthorized use of the Harness Platform or breach of this Agreement; or (4) if Customer fails to install or use any functionally equivalent non-infringing aspect of the Harness Platform that

would have avoided the alleged infringement. The foregoing shall be Customer's sole remedy with respect to any claim of infringement of third party intellectual property rights.

7.4 By Customer. Customer agrees to defend, at its expense, Harness and its Affiliates, its suppliers and its resellers against any third party claim to the extent such claim alleges, arises from, or is made in connection with Customer's breach of Section 1 (Harness Platform) or Section 10 (Data Use), any High Risk Use, or Customer's negligence or willful misconduct. Customer shall pay all costs and damages finally awarded against Harness by a court of competent jurisdiction as a result of any such claim.

7.5 Indemnification Requirements. In connection with any claim for indemnification under this Section 7, the indemnified party must promptly provide the indemnifying party with notice of any claim that the indemnified party believes is within the scope of the obligation to indemnify, provided, however, that the failure to provide such notice shall not relieve the indemnifying party of its obligations under this Section 7, except to the extent that such failure materially prejudices the indemnifying party's defense of such claim. The indemnified party may, at its own expense, assist in the defense if it so chooses, but the indemnifying party shall control the defense and all negotiations related to the settlement of any such claim. Any such settlement intended to bind either party shall not be final without the other party's written consent, which consent shall not be unreasonably withheld, conditioned or delayed; provided, however, that Customer's consent shall not be required when Harness is the indemnifying party if the settlement involves only the payment of money by Harness.

8. Limitation of Liability

The limits below will not apply to the extent prohibited by applicable law. EXCEPT FOR LIABILITY ARISING FROM VIOLATIONS OF SECTION 1.2 (RESTRICTIONS ON USE), CUSTOMER'S PAYMENT OBLIGATIONS, OR A PARTY'S INFRINGEMENT OR MISAPPROPRIATION OF THE OTHER PARTY'S INTELLECTUAL PROPERTY RIGHTS, UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER IN TORT, CONTRACT, OR OTHERWISE, WILL EITHER PARTY BE LIABLE TO THE OTHER PARTY UNDER THIS AGREEMENT FOR ANY (A) INDIRECT, SPECIAL, INCIDENTAL, CONSEQUENTIAL OR PUNITIVE DAMAGES OF ANY CHARACTER, INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, LOST PROFITS, LOST SALES OR BUSINESS, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, LOST DATA, OR FOR ANY AND ALL OTHER INDIRECT DAMAGES OR LOSSES, EVEN IF SUCH PARTY HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY OF SUCH DAMAGES, OR (B) DIRECT DAMAGES, COSTS, OR LIABILITIES IN EXCESS OF THE AMOUNTS PAID BY CUSTOMER DURING THE TWELVE (12) MONTHS IMMEDIATELY PRECEDING THE INCIDENT OR CLAIM UNDER THE APPLICABLE ORDER FORM OR SOW. THESE LIMITATIONS SHALL APPLY NOTWITHSTANDING ANY FAILURE OF ESSENTIAL PURPOSE OF ANY REMEDY.

9. Data Use

9.1 System Data. Harness shall have the right collect and analyze System Data (including, without limitation, information concerning Customer and data derived therefrom), and Harness will be free (during and after the term hereof) to (i) use such information and data to improve and enhance the Harness Platform and for other development, diagnostic and corrective purposes in connection with the Harness Platform and other Harness offerings, and (ii) disclose such data solely in aggregate or other de-identified form in connection with its business. "System Data" means data collected by Harness regarding the Harness Platform that may be used to generate logs, statistics or reports regarding the performance, availability, usage, integrity or security of the Harness Platform.

9.2 Customer Data and License to Customer Data. "Customer Data" means electronic data submitted by Customer through the Harness Platform. As between the parties, Customer exclusively owns and reserves all rights, title, and interest in and to the Customer Data and Customer's software applications (and all derivatives, modifications, improvements, or enhancements related to any of the foregoing), which includes all copyright, patent, trade secret, trademark and all other intellectual property rights related to any of the foregoing. Customer hereby grants to Harness and its Affiliates a non-exclusive right to access, use, and process Customer Data, as necessary to provide the Harness Platform and the Services in accordance with this Agreement, and to improve the functioning and usability of the Harness Platform. Customer is solely responsible for the quality and integrity of Customer Data. Customer represents and warrants that it has obtained all the necessary consents to provide Harness with the foregoing license to Customer Data.

9.3 Personal Identifiable Information. If Customer provides Harness with any personally identifiable information ("Personal Data"), Customer represents and warrants that such Personal Information is not Prohibited Data, has been collected by Customer in accordance with the provisions of all applicable data protection laws and regulations, and that Customer has all right and consents necessary to provide such Personal Data to Harness.

10. Miscellaneous

10.1 Governing Law; Venue. This Agreement shall be governed by and construed under the laws of the State of California, U.S.A., as if performed wholly within the state and without giving effect to the principles of conflict of law. The parties consent to the exclusive jurisdiction and venue of the courts located in and serving San Francisco, California. The Uniform Computer Information Transactions Act (UCITA) nor the United Nations Convention for the International Sale of Goods will apply to this Agreement.

10.2 No Waiver. Failure by either party to exercise any of its rights under, or to enforce any provision of, this Agreement will not be deemed a waiver or forfeiture of such rights or ability to enforce such provision. If any provision of this Agreement is held by a court of competent jurisdiction to be illegal, invalid or unenforceable, such provision will be amended to achieve as nearly as possible the same economic effect of the original provision and the remainder of this Agreement will remain in full force and effect.Â

10.3 Entire Agreement. This Agreement (including each Order Form and SOW) represents the entire agreement between the parties and supersedes any previous or contemporaneous oral or written agreements or communications regarding the subject matter of this Agreement.Â

10.4 Order of Precedence. This Agreement shall control over additional or different terms of any purchase order, confirmation, invoice, statement of work or similar document (other than the Order Form or SOW, which will take precedence), even if accepted in writing by both parties, and waivers and amendments to this Agreement shall be effective only if made by non-pre-printed agreements clearly understood by both parties to be an amendment or waiver to this Agreement.Â

10.5 Interpretation. All capitalized terms used but not defined in an Order Form or SOW shall have the meanings provided to them in the Agreement. For purposes of this Agreement, âincludingâ means âincluding without limitation.âÂ

10.6 Severability. If any provision of this Agreement is held by a court of competent jurisdiction to be illegal, invalid or unenforceable, such provision will be amended to achieve as nearly as possible the same economic effect of the original provision and the remainder of this Agreement will remain in full force and effect.

10.7 Cumulative Remedy. The rights and remedies of the parties hereunder will be deemed cumulative and not exclusive of any other right or remedy conferred by this Agreement or by law or equity.Â

10.8 Independent Contractors. No joint venture, partnership, employment, or agency relationship exists between the parties as a result of this Agreement or use of the Harness Platform.Â

10.9 Assignment and Delegation. Customer may not assign this Agreement without the prior written consent of Harness, and any purported assignment in violation of this Section 10.9 shall be void. Harness may assign, transfer or subcontract this Agreement in whole or in part without Customerâs consent. Upon any assignment of this Agreement by Customer that is approved by Harness or other corporate transaction involving Customer that would materially increase its Licensee Unit usage, if the Order Form contains a subscription for an âunlimitedâ amount of Licensee Units, such subscription will, with respect to Customer or the successor entity, as applicable, be capped at the monthly average of authorized Licensee Units used by Customer under such Order Form during the three full calendar months prior to such assignment (or if the Harness Platform has been used for fewer than three full calendar months, then the monthly average based on a pro rata calculation of such use). Harness reserves the right to perform its obligations from locations and/or through use of Affiliates, contractors and subcontractors, worldwide, provided that Harness will be responsible for such parties.

10.10 Force Majeure. With the exception of Customerâs payment obligations, no delay, failure, or default, other than a failure to pay fees when due, will constitute a breach of this Agreement to the extent caused by epidemics, acts of war, terrorism, hurricanes, earthquakes, cyberattacks, other acts of God or of nature, strikes or other labor disputes, riots or other acts of civil disorder, embargoes, government orders responding to any of the foregoing, or other causes beyond the performing partyâs reasonable control.

10.11 Publicity. Customer agrees that Harness may refer to Customer by its trade name and logo, and may briefly describe Customerâs business, in Harnessâs marketing materials and website. Additionally, Customer and Harness shall collaborate in good faith for the purpose of executing various co-marketing activities (e.g., customer testimonial videos, case study write ups, conference speaking slots, serving as a referenceable customer, etc.). The parties agree that all co-marketing activities will be contingent on a successful deployment of the Harness Platform.

10.12 Notice. Harness may give notice to Customer by electronic mail to Customerâs email address as provided by Customer on the Order Form or on record in Customerâs account information, or by written communication sent by first class mail or pre-paid post to Customerâs address as provided by Customer on the Order Form or on record in Customerâs account information. Customer may give notice to Harness at any time by any letter delivered by nationally recognized overnight delivery service or first class postage prepaid mail to Harness at the following address or such other address as may be notified to Customer from time to time: Harness, 55 Stockton St., 8th Floor, San Francisco, CA 94108, Attn.: Legal Department. Notice under this Agreement shall be deemed given when received, if personally delivered; when receipt is electronically confirmed, if transmitted by email; the day after it is sent, if sent for next-day delivery by a recognized overnight delivery service; and upon receipt, if sent by certified or registered mail, return receipt requested.Â

10.13 Updates. Harness may update this Agreement from time to time by providing you with prior written notice of material updates at least thirty (30) days in advance of the effective date. Such notice will be given in accordance with this section. Except as otherwise specified by Harness, (a) updates will be effective upon the effective date indicated at the top of this Agreement or in such notice, (b) your continued access or use of the Harness Platform or Services on or after the effective date of such updates constitutes your acceptance of such updates and (c) if you do not agree to such updates, you should stop using the Harness Platform and Services. However, if you have paid for a subscription to the Harness Platform, and we update this Agreement during your License Term, the updates with respect to that subscription will be effective upon your next Renewal Term, if applicable, and in this case, if you object to the updates, as your sole and exclusive remedy, you may choose not to renew, in accordance with the terms hereof. The updated version of the Agreement will supersede all prior versions.

Source URL: <https://www.harness.io/support/support-code-of-conduct>

Harness Support code of conduct

Harness is committed to treating all customers, prospects, partners, and users with respect, empathy, and kindness. We expect to be treated the same way.

Misconduct toward Harness employees and support staff may result in support tickets being closed, a ban on a specific individual from future support requests, or in the case of repeat offenders, termination of the business relationship with Harness.

We are committed to making participation in the Harness Support experience a harassment-free experience for everyone, regardless of the level of experience, gender, gender identity and expression, sexual orientation, disability, personal appearance, body size, race, ethnicity, age, religion, or nationality.

Scope

This code of conduct applies to all participants in the Harness Support experience, as individuals and representatives of an organization/company/agency/non-profit.

Our Standards

Examples of behavior that contributes to a positive environment include:

- Demonstrating empathy and kindness toward other people
- Being respectful of differing opinions, viewpoints, and experiences
- Giving and gracefully accepting constructive feedback
- Accepting responsibility and apologizing to those affected by our mistakes, and learning from the experience

Examples of unacceptable behavior include:

- The use of sexualized language or imagery, and sexual attention or advances of any kind
- Trolling, insulting or derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or email address, without their explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

Reporting

To report non-compliance with this code of conduct, submit this form. You may choose to remain anonymous. You can expect a response within three business days. Use of an outside mediator can be requested and is at Harness staff's discretion.

Enforcement

Misconduct toward Harness employees and support staff will result in support requests being denied, and the ticket being closed.

Repeat offenders may be permanently banned from making a support request, and/or termination of the business relationship with Harness.

Acknowledgments

This Code of Conduct is adapted from the Contributor Covenant, version 2.0 available here.

Source URL: <https://www.harness.io/blog/argocd-terraform-and-harness>

ArgoCD, Terraform and Harness

GitOps with ArgoCD, Terraform and Harness

â

GitOps has taken the DevOps world by storm. GitOps is a DevOps strategy where teams use files in Git to declaratively define the desired state of an environment. An automated reconciler is powerful and applies changes to that state to whatever is being managed, typically application environments and infrastructure. GitOps is so powerful because it meets developers where they are, provides audit trails of changes, and with pull-requests, generates a layer of approvals.Â

Challenges

As attractive as GitOps is, it isn't without challenges. For example, the automated reconcilers tend to be domain-specific with tools like ArgoCD and Flux targeting applications in Kubernetes and infrastructure-as-code tools like Terraform operating at infrastructure levels. If you want to create a cluster and deploy an application to it, multiple automation tools need to be stitched together.

â

There is also a heavy governance burden pushed down to developers. When reviewing that PR, how do I know how well this version performed in tests? Are there security defects? If we're manipulating infrastructure, how would I understand the cost implications?

â

Developers either have to go digging to review each change in painstaking detail or approve after a cursory check and hope for the best. Neither approach is great.Â

â

Visibility and reporting can also be a challenge when you need to check multiple instances of multiple tools to understand the current state. The definitions in Git can help, but they can be tricky to display together in an easy-to-digest way.

â

Bootstrapping environments and managing GitOps infrastructure can be tricky as well. How many ArgoCD instances should we have? How will they be maintained?

A healthy approach for GitOps

In the five-minute video below, we demonstrate one approach to addressing these challenges. We have it baked into our Harness platform.Â

â

â

Challenges addressed:

- **Multiple reconcilers:** While still using ArgoCD for the application and Terraform for the infrastructure, Harness orchestrates between them, simplifying the experience.
- **Governance:** Harness decorated the pull request with key quality telemetry, including checks of organizational standards defined in OPA. This simplified the checking for the developer.Â
- **Views and reporting:** Harness provided a single reporting interface, making it clear what is deployed where.
- **Bootstrapping:** The ArgoCD reconciler is installed alongside the environment creation, minimizing the effort required to run a GitOps approach at scale.Â

Conclusion: Harness as a DevOps Game-Changer

â

As we've seen in the video demonstration, Harness's capabilities in managing infrastructure as code, integrating with GitOps workflows, and ensuring compliance and governance standards position it as a game-changer in the DevOps landscape. Its ability to streamline and automate processes not only saves time but also helps in maintaining high standards of efficiency and security, making it a powerful tool for any forward-thinking DevOps team.

â

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/docs/api/>

- Introduction
 - How it works
 - Get started
 - Authentication
 - Requests and Responses
 - Common Parameters [Beta]
 - Status Codes
 - Versioning [Beta]
 - Pagination [Beta]
- Organizations
 - - Organization
 - post
Create an organization [Beta]
 - get
List organizations [Beta]
 - get
Retrieve an organization [Beta]
 - put
Update an organization [Beta]
 - del
Delete an organization [Beta]
 - get
List Organizations by filter
 - post
Create an Organization
 - get
List Organization details
 - put
Update an Organization
 - del
Delete an Organization
 - Projects
 - - Project [Beta]
 - post
Create a project
 - get
List projects
 - get
Retrieve a project
 - put
Update a project
 - del

- Delete a project
- Project
 - get
 - List all Projects for a user
 - post
 - Create a Project
 - get
 - List Project details
 - put
 - Update a Project
 - del
 - Delete a Project
 - get
 - List user's project with support to filter by multiple organizations
- Secrets
 - Account Secrets [Beta]
 - post
 - Create a secret
 - get
 - List secrets
 - post
 - Validate secret reference
 - get
 - Retrieve a secret
 - del
 - Deletes a secret
 - put
 - Update a secret
 - Organization Secrets [Beta]
 - post
 - Create a secret
 - get
 - List secrets
 - post
 - Validate secret reference
 - get
 - Retrieve a secret
 - del
 - Delete a secret
 - put
 - Update a secret
 - Project Secrets [Beta]
 - post
 - Create a secret
 - get
 - List secrets
 - post
 - Validate secret reference
 - get
 - Retrieve a secret
 - del
 - Delete a secret
 - put
 - Update a secret
 - Secrets
 - get
 - Fetches the list of Secrets corresponding to the request's filter criteria.
 - post
 - Creates a Secret at given Scope
 - post
 - Creates a Secret File
 - post
 - Creates a secret via YAML
 - get
 - Get the Secret by ID and Scope
 - put
 - Updates the Secret by ID and Scope

- del
Deletes Secret by ID and Scope
 - post
Fetches the list of Secrets corresponding to the request's filter criteria.
 - put
Updates the Secret file by ID and Scope
 - put
Updates the Secret by ID and Scope via YAML
 - post
Validates Secret with the provided ID and Scope
 - get
Checks whether the identifier is unique or not
- Connectors
 - Account Connectors [Beta]
 - post
Create a Connector
 - get
Retrieve a connector
 - put
Update a connector
 - del
Delete a connector
 - get
Test a connector
 - Organization Connectors [Beta]
 - post
Create a Connector
 - get
Retrieve a connector
 - put
Update a connector
 - del
Delete a connector
 - get
Test a connector
 - Project Connectors [Beta]
 - post
Create a Connector
 - get
Retrieve a connector
 - put
Update a connector
 - del
Delete a connector
 - get
Test a connector
 - Connectors
 - post
Fetches the list of CMC K8S Connectors corresponding to the request's filter criteria.
 - get
List all Connectors using filters
 - put
Update a Connector
 - post
Create a Connector
 - get
Return Connector details
 - del
Delete a Connector
 - get
List all the configured field values for the given Connector type.
 - post
Get the Template URL of connector
 - get
Lists all Connectors for an account
 - get
Gets the connector's statistics by Account Identifier, Project Identifier and Organization Identifier
 - post

- Fetches the list of Connectors corresponding to the request's filter criteria.
 - post
Get list of Connectors by FQN
 - post
Test Harness Connector connection with third-party tool
 - post
Test Git Connector sync with repo
 - get
Test a Harness Connector
 - GoogleSecretManagerConnector
 - get
Get list of GCP Regions
- Roles
 - Account Roles [Beta]
 - get
List Roles
 - post
Create a Role
 - get
Retrieve a Role
 - put
Update a Role
 - del
Delete a Role
 - Organization Roles [Beta]
 - get
List Roles
 - post
Create a Role
 - get
Retrieve a Role
 - put
Update a Role
 - del
Delete a Role
 - Project Roles [Beta]
 - get
List Roles
 - post
Create a Role
 - get
Retrieve a Role
 - put
Update a Role
 - del
Delete a Role
 - Roles
 - get
List Roles
 - post
Create Role
 - get
Get Role
 - put
Update Role
 - del
Delete Role
- Resource Groups
 - Account Resource Groups [Beta]
 - get
List Resource Groups
 - post
Create a Resource Group
 - get
Retrieve a Resource Group
 - put

- Update a Resource Group
 - del
 - Delete a Resource Group
 - Organization Resource Groups [Beta]
 - get
 - List Resource Groups
 - post
 - Create a Resource Group
 - get
 - Retrieve a Resource Group
 - put
 - Update a Resource Group
 - del
 - Delete a Resource Group
 - Project Resource Groups [Beta]
 - get
 - List Resource Groups
 - post
 - Create a Resource Group
 - get
 - Retrieve a Resource Group
 - put
 - Update a Resource Group
 - del
 - Delete a Resource Group
 - Filter Resource Groups [Beta]
 - post
 - Filter Resource Groups
 - Harness Resource Group
 - get
 - List Resource Groups
 - post
 - Create Resource Group
 - get
 - Get Resource Group
 - put
 - Update Resource Group
 - del
 - Delete Resource Group
 - post
 - List Resource Groups by filter
 - Zendesk
 - post
 - create zendesk ticket for given user
 - get
 - get short live token for Coveo
- Role Assignments
 - Account Role Assignments [Beta]
 - get
 - List role assignments
 - post
 - Create a role assignment
 - get
 - Retrieve a role assignment
 - del
 - Delete a role assignment
 - Organization Role Assignments [Beta]
 - get
 - List role assignments
 - post
 - Create a role assignment
 - get
 - Retrieve a role assignment
 - del
 - Delete a role assignment

- Project Role Assignments [Beta]
 - get
 - List role assignments
 - post
 - Create a role assignment
 - get
 - Retrieve a role assignment
 - del
 - Delete a role assignment
- Role Assignments
 - post
 - Bulk Delete Role Assignment
 - post
 - Create Role Assignments
 - get
 - List Role Assignments
 - post
 - Create Role Assignment
 - get
 - Get Role Assignment
 - del
 - Delete Role Assignment
 - post
 - List Role Assignments by filter
 - post
 - List Aggregated Role Assignments by filter
 - post
 - List Role Assignments by scope filter
 - post
 - Validate Role Assignment
- Platform
 - Accounts
 - get
 - Gets an account
 - get
 - Checks if immutable delegate is enabled for account
 - put
 - Update Account Name
 - put
 - Update Default Experience
 - AccountSetting
 - get
 - Get the AccountSetting by accountIdentifier
 - put
 - Updates account settings
 - get
 - Get the AccountSetting by accountIdentifier
 - Access Control List
 - post
 - Check Permission
 - AuditFilters
 - get
 - Get the list of Filters of type Audit satisfying the criteria (if any) in the request
 - put
 - Updates the Filter of type Audit
 - post
 - Creates a Filter
 - get
 - Gets a Filter of type Audit by identifier
 - del
 - Delete a Filter of type Audit by identifier
 - Audit
 - post
 - List Audit Events

EULA [Beta]

- post
Sign an End User License Agreement
- get
Validate specified agreement is signed or not

-

Filter

- get
List Filters
- put
Update a Filter
- post
Create a Filter
- get
Return Filter Details
- del
Delete a Filter
- get
List Filters
- put
Update a Filter
- post
Create a Filter
- get
Return Filter Details
- del
Delete a Filter
- get
List Filters
- put
Update a Filter
- post
Create a Filter
- get
Return Filter Details
- del
Delete a Filter
- get
List Filters
- put
Update a Filter
- post
Create a Filter
- get
Return Filter Details
- del
Delete a Filter
- get
List Filters
- put
Update a Filter
- post
Create a Filter
- get
Return Filter Details
- del
Delete a Filter

-

Invite

- put
Resend invite
- del
Delete Invite
- get
Get Invite
- get
List Invites
- post
Get pending users

-

IP Allowlist [Beta]

- post
Create a IP Allowlist config
- get
List IP Allowlist Configs
- get
Retrieve a IP Allowlist config
- put
Update IP Allowlist config
- del
Delete an IP Allowlist config
- get

- Validate unique IP Allowlist config identifier
- get
 - Validate IP address lies in a specified range or not
- Oidc-Access-Token
 - post
 - Generate an OIDC IAM Role Credential for AWS
 - post
 - Generates an OIDC Service Account Access Token for GCP
 - post
 - Generates an OIDC Workload Access Token for GCP
- Oidc-ID-Token
 - post
 - Generates an OIDC ID Token for AWS
 - post
 - Generates an OIDC ID Token for GCP
- OIDC
 - get
 - Get the openid configuration for Harness
 - get
 - Get the openid configuration for Harness
- Canny
 - post
 - create Canny Post for given user
 - get
 - Get a list of boards available on Canny
- ApiKey
 - get
 - Fetches the list of API Keys corresponding to the request's filter criteria.
 - post
 - Creates an API key
 - put
 - Updates API Key for the provided ID
 - del
 - Deletes the API Key corresponding to the provided ID.
 - get
 - Fetches the API Keys details corresponding to the provided ID and Scope.
 - get
 - Fetches the list of Aggregated API Keys corresponding to the request's filter criteria.
- Source Code Manager
 - put
 - Updates Source Code Manager Details with the given Source Code Manager Id
 - del
 - Deletes the Source Code Manager corresponding to the specified Source Code Manager Id
 - get
 - Lists Source Code Managers for the given account
 - post
 - Creates Source Code Manager
- Nextgen Ldap
 - post
 - Test LDAP authentication
 - get
 - Return Ldap groups matching name
- Harness Resource Type
 - get
 - Gets all resource types available at this scope
- Authentication Settings
 - get
 - Return configured Ldap settings for the account
 - put
 - Updates Ldap setting
 - post
 - Create Ldap setting
 - del

- >Delete Ldap settings
 - del
 - >Delete SAML meta data
 - del
 - Delete SAML meta data for given SAML sso id
 - put
 - Update authentication enabled or not for given SAML setting
 - get
 - Gets authentication settings for the given Account ID
 - get
 - Gets authentication settings version 2 for the given Account ID
 - get
 - Get password strength
 - get
 - Test SAML connectivity
 - get
 - Test SAML connectivity
 - del
 - Delete OAuth Setting
 - put
 - Enable/disable public access at account level
 - put
 - Set session timeout at account level
 - put
 - Set two factor authorization
 - put
 - Update Auth mechanism
 - put
 - Update Oauth providers
 - put
 - Update SAML metadata
 - post
 - Upload SAML metadata
 - put
 - Update SAML metadata for a given SAML SSO Id
 - put
 - Updates the whitelisted domains
- Permissions
 - get
 - List Permissions
 - get
 - List Resource Types
- Secret Managers
 - post
 - Gets the metadata of Secret Manager
- Setting
 - get
 - Get a setting value by identifier
 - get
 - Get list of settings under the specified category
 - put
 - Update settings
- Service Account
 - get
 - Get Service Accounts
 - post
 - Create a Service Account
 - put
 - Update a Service Account
 - del
 - Delete a Service Account
 - get
 - Get Service Account In Scope
 - get
 - List aggregated Service Accounts
- SMTP
 - del

- >Delete Smtplib Config by identifier
 - get
 - Gets Smtplib config by accountId
 - put
 - Updates the Smtplib Config
 - post
 - Creates SMTP config
 - post
 - Tests the config's connectivity by sending a test email
 - post
 - Checks whether other connectors exist with the same name
- Token
 - post
 - Create a Token
 - put
 - Update a Token
 - del
 - Delete a Token
 - get
 - List all Tokens
 - post
 - Rotate a Token
 - post
 - Validate a Token
- User
 - post
 - Add user(s) to scope
 - put
 - Change user password
 - get
 - Check if user is last admin
 - put
 - Disable two factor authentication
 - get
 - Get detailed user information
 - post
 - Get list of users
 - get
 - Gets Two Factor Auth Settings
 - get
 - Get Current User Info
 - post
 - Get users list
 - put
 - Update User
 - del
 - Remove user from scope
 - get
 - Reset two factor authorization
 - put
 - Unlock user
 - put
 - Enable two factor authentication
 - put
 - Update User
- User Group
 - get
 - Check user membership
 - put
 - Add user to User Group
 - del
 - Remove user from User Group
 - put
 - Copy User Group
 - get
 - List the User Groups in an account/org/project
 - put
 - Update User Group
 - post

- Create User Group
 - get
 - Get User Group
 - del
 - Delete a User Group in an account/org/project
 - get
 - Get Inheriting Child Scopes
 - post
 - List users in User Group
 - put
 - Link LDAP Group to the User Group to an account/org/project
 - put
 - Link SAML Group to the User Group in an account/org/project
 - post
 - List User Groups by filter
 - post
 - Get filtered User Groups
 - put
 - Unlink SSO Group from the User Group in an account/org/project
 - put
 - Update User Group
 - post
 - Create User Group
- Variables
 - get
 - Fetches the list of Variables.
 - put
 - Updates the Variable.
 - post
 - Creates a Variable.
 - get
 - Get the Variable by scope identifiers and variable identifier.
 - del
 - Deletes Variable by ID.
- Delegate
 - Agent mTLS Endpoint Management
 - get
 - Gets the agent mTLS endpoint for an account.
 - put
 - Updates the existing agent mTLS endpoint for an account.
 - post
 - Creates the agent mTLS endpoint for an account.
 - del
 - Removes the agent mTLS endpoint for an account.
 - patch
 - Updates selected properties of the existing agent mTLS endpoint for an account.
 - get
 - Checks whether a given agent mTLS endpoint domain prefix is available.
 - Delegate Download Resource
 - post
 - Downloads a docker delegate yaml file.
 - post
 - Downloads a kubernetes delegate yaml file.
 - Delegate Group Tags Resource
 - get
 - Retrieves list of tags attached with Delegate group
 - put
 - Clears all existing tags with delegate group and attach given set of tags to delegate group.
 - post
 - Add given list of tags to the Delegate group
 - del
 - Deletes all tags from the Delegate group
 - post
 - List delegate groups that are having mentioned tags.
 - Delegate Setup Resource
 - del
 - Deletes a Delegate by its identifier.

- **get**
Generates delegate terraform example module file from the account
- **post**
Generates helm values yaml file from the data specified in request body (Delegate setup details).
- **post**
Lists all delegates in NG filtered by provided conditions
- **get**
Gets the latest supported delegate version. The version has YY.MM.XXXXX format. You can use any version lower than the returned results(upto 3 months old)
- **put**
Overrides delegate image tag for account
- Delegate Token Resource
 - **get**
Retrieves Delegate Tokens by Account, Organization, Project and status.
 - **put**
Revokes Delegate Token.
 - **post**
Creates Delegate Token.
 - **get**
Lists delegate groups that are using the specified delegate token.
- Pipelines
 - Pipelines [Beta]
 - **get**
List Pipelines
 - **post**
Create a Pipeline
 - **get**
Retrieve a Pipeline
 - **put**
Update a Pipeline
 - **del**
Delete a Pipeline
 - **patch**
Patch API for pipeline
 - **post**
Move Pipeline YAML from inline to remote
 - **post**
Get Pipeline YAML from Git Repository
 - **put**
Update GitMetadata for Remote Pipelines
 - Input Sets [Beta]
 - **post**
Create an Input Set
 - **get**
List Input Sets
 - **get**
Retrieve an Input Set
 - **put**
Update an Input Set
 - **del**
Delete an Input Set
 - **post**
Move InputSet YAML from inline to remote
 - **post**
Get Input Set YAML from Git Repository
 - **put**
Update GitMetadata for Remote InputSet
 - Approvals
 - **get**
Gets Approval Instances by Execution Id
 - **post**
Approve or Reject an Execution by Pipeline Execution ID
 - **post**
Approve or Reject a Pipeline Execution
 - Pipeline Execution [Beta]
 - **post**
Execute Pipeline

- **get**
Get Stages execution List for a given Pipeline execution.
- **post**
Re-run Stages Execution of a Pipeline
- **post**
Execute given Stages of a Pipeline
- Pipeline Dashboard
 - **get**
Fetch Execution Details for an Interval
- Pipeline Input Set
 - **get**
List Input Sets
 - **post**
Create an Input Set
 - **post**
Create an Overlay Input Set for a pipeline
 - **get**
Fetch an Input Set
 - **put**
Update an Input Set
 - **del**
Delete an Input Set
 - **get**
Gets an Overlay Input Set by identifier
 - **put**
Update an Overlay Input Set for a pipeline
 - **post**
Fetch Runtime Input Template
 - **put**
Update git-metadata in remote input-set
- Pipeline
 - **post**
Create a Pipeline
 - **post**
Create a Pipeline
 - **get**
Fetch a Pipeline
 - **put**
Update a Pipeline
 - **del**
Delete a Pipeline
 - **post**
List Pipelines
 - **get**
Fetch Pipeline Summary
 - **post**
Import and Create Pipeline from Git Repository
 - **post**
Import and Create Pipeline from Git Repository
 - **put**
Update git-metadata in remote pipeline Entity
 - **put**
Update a Pipeline
- Pipeline Execution Details
 - **get**
Fetch Execution Details
 - **get**
Fetch Execution Details
 - **get**
Fetch Execution SubGraph for a Given NodeExecution ID
 - **get**
Get the Input Set YAML used for given Plan Execution
 - **post**
List Execution Identifier
 - **post**
List Executions
 - **get**
Get Notes for a pipelineExecution

- **put**
Updates Notes for a pipelineExecution
- Pipeline Execute
 - **get**
Retry History for a given execution
 - **put**
Execute an Interrupt
 - **put**
Handles the interrupt for a given stage in a pipeline
 - **post**
Retry a executed pipeline with inputSet pipeline yaml
 - **post**
Execute a Pipeline with Input Set References
 - **post**
Execute a Pipeline with Runtime Input YAML
 - **post**
Execute given Stages of a Pipeline
- Pipeline Refresh
 - **get**
Validates template inputs in a pipeline's YAML specification.
- Triggers [Beta]
 - **get**
Gets the paginated list of triggers for accountIdentifier, orgIdentifier, projectIdentifier, targetIdentifier.
 - **post**
Creates Trigger for triggering target pipeline identifier.
 - **get**
Gets the trigger by accountIdentifier, orgIdentifier, projectIdentifier, targetIdentifier and triggerIdentifier.
 - **put**
Updates trigger for pipeline with target pipeline identifier.
 - **del**
Deletes Trigger by identifier.
 - **get**
Lists all Triggers
 - **get**
Fetches Trigger details for a specific accountIdentifier, orgIdentifier, projectIdentifier, targetIdentifier, triggerIdentifier.
 - **get**
Get event history for a trigger
- TriggersEvents
 - **get**
Get all the polled response for a given trigger
 - **get**
Get event history for a trigger
 - **get**
Get Trigger history event correlation
 - **get**
Get Trigger history event correlation V2
 - **get**
Get artifact and manifest trigger event history based on build source type
- Webhook Triggers
 - **get**
Gets webhook event processing details for input eventId.
 - **get**
Gets webhook event processing details for input eventId.
 - **post**
Handles event payload for webhook triggers.
 - **post**
Handles event payload for custom webhook triggers.
 - **post**
Handles event payload for custom webhook triggers.
 - **post**
Handles event payload for custom webhook triggers.
- Webhook Event Handler
 - **post**
Process event payload for webhook triggers.
- CD

- - K8s Release Service Mapping [Beta]
 - get
 - List service and environment details using namespace and releasesname
 - CustomDeployment
 - post
 - Gets Custom Deployment Entity References
 - post
 - Gets Custom Deployment Expression Variables
 - get
 - Gets Infra Variables from a Custom Deployment Template by identifier
 - post
 - Return the updated yaml for infrastructure based on Deployment template
 - get
 - This validates whether Infrastructure is valid or not
 - Environments
 - get
 - Gets Environment list for a project
 - put
 - Update an Environment by identifier
 - post
 - Create an Environment
 - get
 - Gets an Environment by identifier
 - del
 - Delete an Environment by identifier
 - get
 - Gets Service Overrides list
 - post
 - upsert a Service Override for an Environment
 - del
 - Delete a ServiceOverride entity
 - get
 - Gets Environment Access list
 - post
 - Move environment YAML from inline to remote
 - put
 - Upsert an Environment by identifier
 - EnvironmentGroup
 - post
 - Create an Environment Group
 - get
 - Gets an Environment Group by identifier
 - put
 - Update an Environment Group by Identifier
 - del
 - Delete en Environment Group by Identifier
 - post
 - Gets Environment Group list
 - Infrastructures
 - get
 - Gets Infrastructure list
 - put
 - Update an Infrastructure by identifier
 - post
 - Create an Infrastructure in an Environment
 - get
 - Gets an Infrastructure by identifier
 - del
 - Delete an Infrastructure by identifier
 - post
 - Move infra YAML from inline to remote
 - Usage
 - get
 - Download CSV Active Services report
 - get
 - Gets License Usage By Module, Timestamp, and Account Identifier

- **get**
Gets License Usage By Module, Timestamp, and Account Identifier
- **get**
Gets License Usage By Module, Timestamp, and Account Identifier
- **get**
Download CSV Active Services report
- **get**
Gets License Usage By Module, Timestamp, and Account Identifier
- **get**
Gets License Usage By Module, Timestamp, and Account Identifier
- **get**
Gets License Usage By Module, Timestamp, and Account Identifier
- **get**
Download CSV Active Monitored Services report
- **get**
Download CSV Active Services Monitored report
- **get**
CvgetLicenseUsage
- **get**
getSRMLicenseUsage
- **post**
Returns a List of active monitored services along with identifier,Active Monitored Services Count and other details
- File Store
 - **get**
List Files and Folders metadata
 - **post**
Create Folder or File including content
 - **post**
Creates File or Folder metadata via YAML
 - **get**
Get the Folder or File metadata
 - **put**
Update Folder or File including content
 - **del**
Delete File or Folder by identifier
 - **get**
Download File
 - **get**
Get list of created by user details
 - **get**
Get file content of scopedFilePath
 - **get**
Get list of entities where file is referenced by queried entity type
 - **get**
Get the list of supported entity types for files
 - **post**
Get filtered list of Files or Folders
 - **post**
Get folder nodes at first level, not including sub-nodes
 - **put**
Update File or Folder metadata via YAML
- Service Dashboard
 - **get**
Get pipeline execution count for a service with grouping support on artifact and deployment status
- ServiceOverrides
 - **put**
Update an ServiceOverride Entity
 - **post**
Create an ServiceOverride Entity
 - **get**
Gets Service Overrides by Identifier
 - **del**
Delete a ServiceOverride entity
- tas
 - **get**
Return the Tas organizations
 - **get**
Return the Tas spaces

- get
Return the Tas spaces
- Deployment Freeze
 - Freeze CRUD
 - post
Create a Freeze
 - get
Get a Freeze
 - put
Updates a Freeze
 - del
Delete a Freeze
 - post
Delete many Freezes
 - post
Gets Freeze list
 - get
Get list of freeze acted on a frozen execution
 - get
Get Global Freeze Yaml
 - post
Create Global Freeze
 - post
Update the status of Freeze to active or inactive
 - Freeze Evaluation
 - Freeze Schema
- Services
 - Account Services [Beta]
 - get
Retrieve a service
 - put
Update service
 - del
Delete a service
 - get
List services
 - post
Create a service
 - Organization Services [Beta]
 - get
Retrieve a service
 - put
Update Service
 - del
Delete a service
 - get
List Services
 - post
Create a service
 - Project Services [Beta]
 - get
Retrieve a service
 - put
Update Service
 - del
Delete a Service
 - get
List Services
 - post
Create a Service
 - Services
 - get
Gets Service list
 - put
Update a Service by identifier
 - post
Create a Service

- post
Create Services
 - get
Gets a Service by identifier
 - del
Delete a Service by identifier
 - get
Retrieving the list of Kubernetes Command Options
 - get
Retrieving the list of Kustomize Command Flags
 - get
Retrieving the list of actions available for service hooks
 - get
Gets Service Access list
 - put
Upsert a Service by identifier
- Rancher Infrastructures
 - Account Rancher Infrastructure [Beta]
 - get
List rancher clusters using account level connector
 - get
List rancher clusters using account level env and infra def
 - Org Rancher Infrastructure [Beta]
 - get
List rancher clusters using org level connector
 - get
List rancher clusters using org level env and infra def
 - Project Rancher Infrastructure [Beta]
 - get
List rancher clusters using project level connector
 - get
List rancher clusters using project level env and infra def
 - Templates
 - Account Templates [Beta]
 - post
Create Template
 - get
Get Templates List
 - put
Update Git details
 - get
Get Stable Template
 - get
Retrieve a Template
 - put
Update Template
 - del
Delete Template
 - put
Update Stable Template
 - post
Import Template
 - Organization Templates [Beta]
 - put
Update Git details
 - post
Create Template
 - get
Get Templates List
 - get
Get Stable Template
 - get
Retrieve a Template
 - put
Update Template
 - del
Delete Template

- put
 - Update Stable Template
 - post
 - Import template
 - Project Templates [Beta]
 - put
 - Update Git details
 - post
 - Create Template
 - get
 - Get Templates List
 - get
 - Get Stable Template
 - get
 - Retrieve a Template
 - put
 - Update Template
 - del
 - Delete Template
 - put
 - Update Stable Template
 - post
 - Import Template
 - Templates
 - post
 - Get YAML with updated Template Inputs
 - get
 - Validate Template Inputs in a YAML
 - post
 - Create a Template
 - del
 - Delete Template Version
 - get
 - Get Template
 - get
 - Gets Template Input Set YAML
 - post
 - Gets all metadata of template list
 - post
 - Move Template YAML from inline to remote
 - put
 - Update Template Version
 - post
 - Update git metadata details for a remote template
 - put
 - Update Stable Template Version
 - Global Templates
 - get
 - Gets Global Template Input Set YAML
- GitOps
 - Agents
 - get
 - AgentServiceForServer_List
 - post
 - AgentServiceForServer_Create
 - post
 - AgentServiceForServer_Search
 - put
 - AgentServiceForServer_Update
 - get
 - AgentServiceForServer_GetDeployYaml
 - post
 - AgentServiceForServer_PostDeployHelmChart
 - post
 - AgentServiceForServer_PostDeployYaml
 - get
 - AgentServiceForServer_GetDeployHelmChart
 - patch

- AgentServiceForServer_SetPrimaryNode
 - get
 - AgentServiceForServer_Get
 - del
 - AgentServiceForServer_Delete
 - post
 - AgentServiceForServer_RegenerateCredentials
 - post
 - AgentServiceForServer_Scale
 - get
 - AgentServiceForServer_Unique
- Application
 - post
 - ApplicationService_ListApps
- Applications
 - get
 - List returns list of applications for a specific agent
 - post
 - Create creates an application
 - get
 - ManagedResources returns list of managed resources
 - get
 - Get returns an application by name
 - get
 - ListResourceEvents returns a list of event resources
 - get
 - PodLogs returns stream of log entries for the specified pod(s).
 - get
 - GetManifests returns application manifests
 - get
 - PodLogs returns stream of log entries for the specified pod(s).
 - get
 - ResourceTree returns resource tree
 - get
 - Get the meta-data (author, date, tags, message) for a specific revision of the application
 - get
 - Get returns sync windows of the application
 - put
 - Update updates an application
 - del
 - Delete deletes an application
 - patch
 - Patch patch an application
 - del
 - TerminateOperation terminates the currently running operation
 - get
 - GetResource returns single application resource
 - del
 - DeleteResource deletes a single application resource
 - post
 - PatchResource patch single application resource
 - get
 - ListResourceActions returns list of resource actions
 - post
 - RunResourceAction run resource action
 - post
 - Rollback syncs an application to its target stateHarness Event type (rollback)
 - put
 - UpdateSpec updates an application spec
 - post
 - Sync syncs an application to its target stateHarness Event type (deploy)
 - get
 - Watch returns stream of application change events
 - get
 - WatchResourceTree returns stream of application resource tree
 - post
 - ListNs returns list of namespaces for a query.
 - post
 - List returns list of application sync status
 - get

- Checks whether an app with the given name exists
- Certificates
 - get
 - List returns list of certificates
- Clusters
 - get
 - Gets a Cluster by identifier
 - del
 - Delete a Cluster by identifier
 - get
 - Gets cluster list
 - post
 - link a Cluster
 - post
 - Link Clusters
 - post
 - Unlink Clusters
 - get
 - List returns list of clusters
 - post
 - Create creates a cluster
 - get
 - Get returns a cluster by identifier
 - del
 - Delete deletes a cluster
 - put
 - Update updates a cluster
 - get
 - List all available repository certificates
 - post
 - CreateHosted creates a harness hosted cluster
 - post
 - List returns list of Clusters
 - get
 - Checks for whether the cluster exists
- Dashboard Aggregates
 - post
 - Returns aggregate statistics of recent deployments
 - get
 - List phase status counts for top 5 most deployed apps
- Dashboards
 - get
 - List count of Cluster, Repos and Apps created within a time series
 - get
 - GetDashboradOverview gets dashboard overview
- GnuPG Keys
 - post
 - Create one or more GPG public keys in the server's configuration
 - get
 - Get information about specified GPG public key from the server
 - del
 - Delete specified GPG public key from the server's configuration
- GPG Keys
 - get
 - List all available repository certificates
- Hosts
 - post
 - Gets the list of hosts filtered by accountIdentifier and connectorIdentifier
 - post
 - Validates hosts connectivity credentials
- Project mappings
 - get
 - AppProjectMappingService_GetAppProjectMappingListByAgent
 - post

- CreateAppProjectMapping creates a new mapping between Harness Project and argo project

 - del
Delete an argo project to harness project mapping
 - get
AppProjectMappingService_GetAppProjectMappingList
 - post
CreateAppProjectMapping creates a new mapping between Harness Project and argo project
 - get
AppProjectMappingService_GetAppProjectMappingV2
 - del
Delete an argo project to harness project mapping
 - put
CreateAppProjectMapping creates a new mapping between Harness Project and argo project
 - get
V2
- Projects
 - get
List returns list of projects
 - post
Create a new project
 - get
Get returns a project by name
 - del
Delete deletes a project
 - put
Update updates a project
- Reconciler
 - post
Returns number of entities that exist in the cluster on the agent. Filter can be used to count only global entities (with empty project) and those specified by the filter.
 - post
Imports data from cluster via agent. There must be at least one project mapping in the database. Returns number of entities imported.
- Repositories
 - get
ListRepositories gets a list of all configured repositories
 - post
CreateRepository creates a new repository configuration
 - get
Checks whether External Secrets Operator is installed
 - get
Returns a list of ESO generators installed in agent namespace.
 - post
Returns the Repository type of OCI repo
 - post
ValidateAccess gets connection state for a repository
 - get
Get returns a repository or its credentials
 - del
DeleteRepository deletes a repository from the configuration
 - put
UpdateRepository updates a repository configuration
 - get
GetAppDetails returns application details by given path
 - get
ListApps returns list of apps in the repo
 - get
GetHelmCharts returns list of helm charts in the specified repository
 - get
Returns a list of refs (e.g. branches and tags) in the repo
 - get
List returns list of Repositories by repository credential template
 - post
List returns list of Repositories
 - get
Checks whether a repository with the given name exists
- Repository Certificates
 - get

- List all available repository certificates
 - del
Delete the certificates that match the RepositoryCertificateQuery
 - post
Creates repository certificates on the server
 - Repository credentials
 - post
Create creates a new repository credential
 - post
Get returns a repository credential given its url
 - get
Get returns a repository credential given its identifier
 - del
Delete deletes a repository credential
 - put
Update updates a repository credential
 - post
List repository credentials
 - ValidateHost
 - post
Validates hosts connectivity credentials
- CCM
 - Cloud Cost Anomalies
 - post
List Anomalies
 - post
Returns the list of distinct values for all the specified Anomaly fields.
 - post
List Anomalies
 - post
List Anomalies for Perspective
 - put
Report Anomaly feedback
 - Cloud Cost BI Dashboards
 - get
List all the BI Dashboards for CCM
 - Cloud Cost Budget Groups
 - get
Fetch Budget group details
 - put
Update an existing budget group
 - del
Delete a budget group
 - post
Get aggregated amount for given budget groups/budgets
 - get
Get list of budget and budget group summaries
 - get
List all the Budget groups
 - post
Create a Budget Group
 - Cloud Cost Budgets
 - get
Fetch Budget details
 - put
Update an existing budget
 - post
Clone a budget
 - del
Delete a budget
 - get
Fetch the cost details of a Budget
 - get
List all the Budgets associated with a Perspective
 - get
List all the Budgets

- post
Create a Budget
- Cloud Cost K8S Connectors Metadata
 - post
Get CCM K8S Metadata
- Cloud Cost Details
 - post
Returns an overview of the cost
 - post
Returns cluster data in a tabular format
 - post
Returns cost details in a tabular format
 - post
Returns cost details in a time series format
- Cloud Cost Overview
 - get
Fetch high level overview details about CCM feature.
- Cloud Cost Cost Categories
 - get
Fetch details of a Cost category
 - del
Delete a Cost category
 - get
Return details of all the Cost categories
 - put
Update a Cost category
 - post
Create Cost category
- RuleEnforcement
 - post
Add a new rule Enforcement
 - post
Fetch Rule Enforcement count for account
 - post
Fetch Rule Enforcement execution details for account
 - post
Fetch Rule Enforcements for account
- Rule
 - post
Clone a rule
 - del
Delete a rule
 - post
Enqueues job for execution
 - get
Get Schema for entity
 - post
Fetch rules for account
 - get
connectors with governance enabled and valid permission
 - put
Update a Rule
 - post
Validate a rule
- Cloud Cost Perspectives Folders
 - post
Create a Perspective folder
 - del
Delete a folder
 - get
Fetch folders for an account
 - put
Update a folder
 - get
Return details of all the Perspectives

- post
 - Move a Perspective
- Cloud Cost Perspective Reports
 - get
 - Fetch details of a cost Report
 - put
 - Update a cost Perspective Report
 - post
 - Create a schedule for a Report
 - del
 - Delete cost Perspective report
- Cloud Cost Perspectives
 - get
 - Fetch details of a Perspective
 - put
 - Update a Perspective
 - post
 - Create a Perspective
 - del
 - Delete a Perspective
 - get
 - Return details of all the Perspectives
 - get
 - Get the last period cost for a Perspective
 - get
 - Get the last twelve month cost for a Perspective
- Cloud Cost Recommendations Details
 - get
 - Return Azure VM Recommendation
 - get
 - Return EC2 Recommendation
 - get
 - Return ECS Recommendation
 - get
 - Return node pool Recommendation
 - get
 - Return workload Recommendation
- Cloud Cost Recommendations
 - post
 - Return void
 - post
 - Return the number of Recommendations
 - post
 - Return the list of filter values for the Recommendations
 - post
 - Return the list of Recommendations
 - post
 - Return Recommendations statistics Grouped on Resource Type
 - post
 - Return Recommendations statistics
- Cloud Cost Recommendation Jira
 - post
 - Create jira for recommendation
- Cloud Cost Recommendation Servicenow
 - post
 - Create servicenow ticket for recommendation
- Cloud Cost Recommendation Ignore List
 - post
 - Add resources to recommendations ignore list
 - get
 - Get resources in recommendations ignore list
 - post
 - Remove resources from recommendations ignore list
- Cloud Cost AutoStopping Rules

- get
 - List AutoStopping Rules
- post
 - Create an AutoStopping Rule
- get
 - Return AutoStopping Rule details
- del
 - Delete an AutoStopping Rule
- get
 - Return health status of an AutoStopping Rule
- get
 - Return savings details for an AutoStopping Rule
- get
 - List all the resources for an AutoStopping Rule
- get
 - Return diagnostics result of an AutoStopping Rule
- post
 - Cool down an AutoStopping Rule
- get
 - Return metadata of cool down of an AutoStopping Rule
- get
 - Return cumulative savings for all the AutoStopping Rules
- put
 - Disable/Enable an Autostopping Rule
- Cloud Cost AutoStopping Rules V2
 - post
 - Create an AutoStopping Rule
 - put
 - Update an existing AutoStopping Rule
- Cloud Cost AutoStopping Load Balancers
 - get
 - Return all the load balancers
 - put
 - Update a load balancer
 - post
 - Create a load balancer
 - del
 - Delete load balancers and the associated resources
 - get
 - Return details of a load balancer
 - get
 - Return all the AutoStopping Rules in a load balancer
 - get
 - Return last activity details of a load balancer
- Cloud Cost AutoStopping Fixed Schedules
 - get
 - Return all the AutoStopping Rule fixed schedules
 - post
 - Create a fixed schedule for an AutoStopping Rule
 - del
 - Delete a fixed schedule for AutoStopping Rule.
- Commitment Orchestrator Events APIs
 - post
 - List event logs
- Feature Flags
 - API Keys
 - get
 - Returns API Keys for an Environment
 - post
 - Creates an API key for the given Environment
 - del
 - Deletes an API Key
 - get
 - Returns API keys
 - put
 - Updates an API Key

- Feature Flags
 - get
 - Returns all Feature Flags for the project
 - post
 - Creates a Feature Flag
 - del
 - Delete a Feature Flag
 - get
 - Returns a Feature Flag
 - patch
 - Updates a Feature Flag
 - get
 - Return a list of dependant flags
 - post
 - Restore a Feature Flag
 - Targets
 - get
 - Returns all Targets
 - post
 - Creates a Target
 - post
 - Add Target details
 - del
 - Deletes a Target
 - get
 - Returns details of a Target
 - patch
 - Updates a Target
 - put
 - Modifies a Target
 - get
 - Returns Target Groups for the given Target
 - Target Groups
 - get
 - Returns all Target Groups
 - post
 - Creates a Target Group
 - del
 - Deletes a Target Group
 - get
 - Returns Target Group details for the given identifier
 - patch
 - Updates a Target Group
 - get
 - Returns Feature Flags that are available to be added to the given Target Group
 - get
 - Returns Feature Flags in a Target Group
 - Environment Perspectives
 - del
 - Delete a Perspective - Environment link.
 - put
 - Upsert a Perspective to an Environment.
- SRM
 - Monitored Services
 - post
 - createDefaultMonitoredService
 - get
 - Get monitored service data
 - put
 - Updates monitored service data
 - del
 - Delete monitored service data
 - put
 - delete template reference from monitored service
 - get
 - getAllMonitoredServicesWithHealthSources
 - get
 - CvgetAnomaliesSummary

- get
getCountOfServices
- get
getEnvironments
- get
getHealthSources
- get
getHealthSourcesForMonitoredServiceIdentifier
- get
getList
- get
getListV2
- get
getMSSecondaryEvents
- get
getMSSecondaryEventsDetails
- get
getMonitoredServiceChangeDetails
- get
getMonitoredServiceDetails
- get
getMonitoredServiceDetails_1
- get
getMonitoredServiceFromServiceAndEnvironment
- get
getMonitoredServiceLogs
- get
fetch reconciliation status for template referenced monitored services
- get
get monitored service resolved template inputs
- get
getMonitoredServiceScore
- get
Get notification rules for MonitoredService
- get
getOverAllHealthScore
- get
getServices
- get
getSloMetrics
- get
check if a template referenced monitored service(s) require reconciliation
- get
list
- post
Saves monitored service data
- post
Saves monitored service from template input
- post
saveMonitoredServiceFromYaml
- put
setHealthMonitoringFlag
- put
Update monitored service from yaml or template
- put
updateMonitoredServiceFromYaml
- get
yamlTemplate
- SLOs dashboard
 - get
getSLOAssociatedEnvironmentIdentifiers
 - get
getSLOAssociatedMonitoredServices
 - get
getSecondaryEventDetails
 - get
getSecondaryEvents
 - get
Get all SLOs count by risk
 - get
Get SLO consumption breakdown

- get
Get SLO dashboard details
- get
Get SLO list view
- post
Get SLO list view
- NG SLOs
 - get
Get SLO data
 - put
Update SLO data
 - del
Delete SLO data
 - post
Get onBoarding graph for composite slo
 - post
Get SLO list view
 - get
Get all SLOs
 - post
Saves SLO data
- SLOs
 - get
Get Error budget reset history
 - get
Get notification rules for SLO
 - get
Get SLO logs
 - post
Reset Error budget history
 - get
Get Metric Graph For SLO
 - get
List SLOs
- Downtime
 - get
getDowntime
 - put
updateDowntimeData
 - del
deleteDowntimeData
 - get
getAssociatedMonitoredServices
 - get
getDowntimeAssociatedMonitoredServices
 - get
getHistory
 - get
listDowntimes
 - post
saveDowntime
 - put
updateDowntimeEnabled
- Srm Notification
 - get
getNotificationRuleData
 - put
updateNotificationRuleData
 - del
deleteNotificationRuleData
 - get
getNotificationRuleData_1
 - post
saveNotificationRuleData
- IDP
 - Backstage Allow List
 - get

- Get backend url allow list
 - post
 - Save backend url allow list
- Backstage App Configurations
 - post
 - Toggle Plugin
 - post
 - Save Or Update Plugin Config
- Backstage Auth Information
 - post
 - Save Auth Info
- Backstage Environment Variable
 - put
 - Reload backstage env variables
- Connector Information
 - get
 - Get Connector Info
 - post
 - Create or Update Connector Info
 - get
 - Get Connector Info by Provider Type
- DataSource Information
 - get
 - Get Datasources Present In Account
 - get
 - Get DataPoints present in DataSources for an account
 - get
 - Get Data Sources and Data Points Map for Account
- Kubernetes DataPoints Information
 - post
 - Get data points data for kubernetes data source
- Layout Proxy
 - post
 - Ingest plugin layout
- Plugin Information
 - get
 - List Available Plugins
 - post
 - Save custom plugin info
 - get
 - Get Plugin
 - put
 - Update custom plugin info
 - post
 - Request for a Plugin
 - get
 - Get all plugin requests for an account
- Scores Information
 - get
 - Get Score Summary for Scorecards
 - get
 - Get Scores for Scorecards
 - post
 - Get Aggregated Scores for backstage entities
 - post
 - Trigger recalibration of scores for a scorecard
- Custom Dashboards
 - Custom Dashboards
 - get
 - Download data within a Dashboard
- Policy Management
 -

- dashboard
 - get
 - dashboard#metrics
- examples
 - get
 - examples#list
- policies
 - get
 - policies#list
 - post
 - policies#create
 - del
 - policies#delete
 - get
 - policies#find
 - patch
 - policies#update
- evaluate
 - post
 - evaluate#evaluate
- evaluations
 - get
 - evaluations#list
 - get
 - evaluations#find
- policysets
 - get
 - policysets#list
 - post
 - policysets#create
 - del
 - policysets#delete
 - get
 - policysets#find
 - patch
 - policysets#update
- system
 - get
 - system#health
 - get
 - system#version
- Git Sync (deprecated)
 - Git Branches
 - post
 - Sync the content of new Git Branch into harness with Git Sync Config Id
 - get
 - Lists branches with their status(Synced, Unsynced) by Git Sync Config Id for the given scope
 - Git Full Sync
 - get
 - Fetch Configuration for Git Full Sync for the provided scope
 - put
 - Update Configuration for Git Full Sync for the provided scope
 - post
 - Create Configuration for Git Full Sync for the provided scope
 - post
 - List files in full sync along with their status
 - post
 - Trigger Full Sync
 - Git Sync Settings
 - get
 - Get Git Sync Setting for the given scope
 - put
 - This updates the existing Git Sync settings within the scope. Only changing Connectivity Mode is allowed

- post
Creates Git Sync Setting in a scope
- - Git Sync
 - get
Lists Git Sync Config for the given scope
 - put
Update existing Git Sync Config by Identifier
 - post
Creates Git Sync Config in given scope
 - get
Check whether Git Sync is enabled for given scope or not
 - put
Update existing Git Sync Config default root folder by Identifier
 - - Git Sync Errors
 - get
Get Errors Count for the given scope, Repo and Branch
 - get
Lists Git to Harness Errors by file or connectivity errors for the given scope, Repo and Branch
 - get
Lists Git to Harness Errors for the given Commit Id
 - get
Lists Git to Harness Errors grouped by Commits for the given scope, Repo and Branch
- Error Models
 - Error Response [Beta]
 - Governance Metadata [Beta]
- Chaos Engineering
- Chaos Engineering
- How it works
- Get started
- Authentication
- Requests and Responses
- Common Parameters [Beta]
- Status Codes
- Versioning [Beta]
- Pagination [Beta]
- - Organization
 - post
Create an organization [Beta]
 - get
List organizations [Beta]
 - get
Retrieve an organization [Beta]
 - put
Update an organization [Beta]
 - del
Delete an organization [Beta]
 - get
List Organizations by filter
 - post
Create an Organization
 - get
List Organization details
 - put
Update an Organization
 - del
Delete an Organization
- post
Create an organization [Beta]
- get
List organizations [Beta]
- get
Retrieve an organization [Beta]
- put
Update an organization [Beta]
- del
Delete an organization [Beta]
- get

List Organizations by filter

- post
Create an Organization
- get
List Organization details
- put
Update an Organization
- del
Delete an Organization
- Project [Beta]
 - post
Create a project
 - get
List projects
 - get
Retrieve a project
 - put
Update a project
 - del
Delete a project
- Project
 - get
List all Projects for a user
 - post
Create a Project
 - get
List Project details
 - put
Update a Project
 - del
Delete a Project
 - get
List user's project with support to filter by multiple organizations
- post
Create a project
- get
List projects
- get
Retrieve a project
- put
Update a project
- del
Delete a project
- get
List all Projects for a user
- post
Create a Project
- get
List Project details
- put
Update a Project
- del
Delete a Project
- get
List user's project with support to filter by multiple organizations
- Account Secrets [Beta]
 - post
Create a secret
 - get
List secrets
 - post
Validate secret reference
 - get
Retrieve a secret
 - del
Deletes a secret

- put
Update a secret
- Organization Secrets [Beta]
 - post
Create a secret
 - get
List secrets
 - post
Validate secret reference
 - get
Retrieve a secret
 - del
Delete a secret
 - put
Update a secret
- Project Secrets [Beta]
 - post
Create a secret
 - get
List secrets
 - post
Validate secret reference
 - get
Retrieve a secret
 - del
Delete a secret
 - put
Update a secret
- Secrets
 - get
Fetches the list of Secrets corresponding to the request's filter criteria.
 - post
Creates a Secret at given Scope
 - post
Creates a Secret File
 - post
Creates a secret via YAML
 - get
Get the Secret by ID and Scope
 - put
Updates the Secret by ID and Scope
 - del
Deletes Secret by ID and Scope
 - post
Fetches the list of Secrets corresponding to the request's filter criteria.
 - put
Updates the Secret file by ID and Scope
 - put
Updates the Secret by ID and Scope via YAML
 - post
Validates Secret with the provided ID and Scope
 - get
Checks whether the identifier is unique or not
- post
Create a secret
- get
List secrets
- post
Validate secret reference
- get
Retrieve a secret
- del
Delete a secret
- put
Update a secret
- post
Create a secret

- get
 - List secrets
- post
 - Validate secret reference
- get
 - Retrieve a secret
- del
 - Delete a secret
- put
 - Update a secret

- post
 - Create a secret
- get
 - List secrets
- post
 - Validate secret reference
- get
 - Retrieve a secret
- del
 - Delete a secret
- put
 - Update a secret

- get
 - Fetches the list of Secrets corresponding to the request's filter criteria.
- post
 - Creates a Secret at given Scope
- post
 - Creates a Secret File
- post
 - Creates a secret via YAML
- get
 - Get the Secret by ID and Scope
- put
 - Updates the Secret by ID and Scope
- del
 - Deletes Secret by ID and Scope
- post
 - Fetches the list of Secrets corresponding to the request's filter criteria.
- put
 - Updates the Secret file by ID and Scope
- put
 - Updates the Secret by ID and Scope via YAML
- post
 - Validates Secret with the provided ID and Scope
- get
 - Checks whether the identifier is unique or not

- Account Connectors [Beta]
 - post
 - Create a Connector
 - get
 - Retrieve a connector
 - put
 - Update a connector
 - del
 - Delete a connector
 - get
 - Test a connector

- Organization Connectors [Beta]
 - post
 - Create a Connector
 - get
 - Retrieve a connector
 - put
 - Update a connector
 - del
 - Delete a connector
 - get
 - Test a connector

- Project Connectors [Beta]
 - post
 - Create a Connector
 - get
 - Retrieve a connector
 - put
 - Update a connector
 - del
 - Delete a connector
 - get
 - Test a connector
- Connectors
 - post
 - Fetches the list of CMC K8S Connectors corresponding to the request's filter criteria.
 - get
 - List all Connectors using filters
 - put
 - Update a Connector
 - post
 - Create a Connector
 - get
 - Return Connector details
 - del
 - Delete a Connector
 - get
 - List all the configured field values for the given Connector type.
 - post
 - Get the Template URL of connector
 - get
 - Lists all Connectors for an account
 - get
 - Gets the connector's statistics by Account Identifier, Project Identifier and Organization Identifier
 - post
 - Fetches the list of Connectors corresponding to the request's filter criteria.
 - post
 - Get list of Connectors by FQN
 - post
 - Test Harness Connector connection with third-party tool
 - post
 - Test Git Connector sync with repo
 - get
 - Test a Harness Connector
- GoogleSecretManagerConnector
 - get
 - Get list of GCP Regions
- post
 - Create a Connector
- get
 - Retrieve a connector
- put
 - Update a connector
- del
 - Delete a connector
- get
 - Test a connector
- post
 - Create a Connector
- get
 - Retrieve a connector
- put
 - Update a connector
- del
 - Delete a connector
- get
 - Test a connector
- post
 - Create a Connector

- get
Retrieve a connector
- put
Update a connector
- del
Delete a connector
- get
Test a connector
- post
Fetches the list of CMC K8S Connectors corresponding to the request's filter criteria.
- get
List all Connectors using filters
- put
Update a Connector
- post
Create a Connector
- get
Return Connector details
- del
Delete a Connector
- get
List all the configured field values for the given Connector type.
- post
Get the Template URL of connector
- get
Lists all Connectors for an account
- get
Gets the connector's statistics by Account Identifier, Project Identifier and Organization Identifier
- post
Fetches the list of Connectors corresponding to the request's filter criteria.
- post
Get list of Connectors by FQN
- post
Test Harness Connector connection with third-party tool
- post
Test Git Connector sync with repo
- get
Test a Harness Connector
- get
Get list of GCP Regions
- Account Roles [Beta]
 - get
List Roles
 - post
Create a Role
 - get
Retrieve a Role
 - put
Update a Role
 - del
Delete a Role
- Organization Roles [Beta]
 - get
List Roles
 - post
Create a Role
 - get
Retrieve a Role
 - put
Update a Role
 - del
Delete a Role
- Project Roles [Beta]
 - get
List Roles
 - post
Create a Role

- get
 Retrieve a Role
 - put
 Update a Role
 - del
 Delete a Role
- Roles
 - get
 List Roles
 - post
 Create Role
 - get
 Get Role
 - put
 Update Role
 - del
 Delete Role
- get
List Roles
- post
Create a Role
- get
Retrieve a Role
- put
Update a Role
- del
Delete a Role
- get
List Roles
- post
Create a Role
- get
Retrieve a Role
- put
Update a Role
- del
Delete a Role
- get
List Roles
- post
Create a Role
- get
Retrieve a Role
- put
Update a Role
- del
Delete a Role
- get
List Roles
- post
Create a Role
- get
Get Role
- put
Update Role
- del
Delete Role
- Account Resource Groups [Beta]
 - get
 List Resource Groups
 - post
 Create a Resource Group
 - get
 Retrieve a Resource Group
 - put
 Update a Resource Group

- del
Delete a Resource Group
- Organization Resource Groups [Beta]
 - get
List Resource Groups
 - post
Create a Resource Group
 - get
Retrieve a Resource Group
 - put
Update a Resource Group
 - del
Delete a Resource Group
- Project Resource Groups [Beta]
 - get
List Resource Groups
 - post
Create a Resource Group
 - get
Retrieve a Resource Group
 - put
Update a Resource Group
 - del
Delete a Resource Group
- Filter Resource Groups [Beta]
 - post
Filter Resource Groups
- Harness Resource Group
 - get
List Resource Groups
 - post
Create Resource Group
 - get
Get Resource Group
 - put
Update Resource Group
 - del
Delete Resource Group
 - post
List Resource Groups by filter
- Zendesk
 - post
create zendesk ticket for given user
 - get
get short live token for Coveo
- get
List Resource Groups
- post
Create a Resource Group
- get
Retrieve a Resource Group
- put
Update a Resource Group
- del
Delete a Resource Group
- get
List Resource Groups
- post
Create a Resource Group
- get
Retrieve a Resource Group
- put
Update a Resource Group
- del
Delete a Resource Group

- get
List Resource Groups
- post
Create a Resource Group
- get
Retrieve a Resource Group
- put
Update a Resource Group
- del
Delete a Resource Group
- post
Filter Resource Groups
- get
List Resource Groups
- post
Create Resource Group
- get
Get Resource Group
- put
Update Resource Group
- del
Delete Resource Group
- post
List Resource Groups by filter
- post
create zendesk ticket for given user
- get
get short live token for Coveo
- Account Role Assignments [Beta]
 - get
List role assignments
 - post
Create a role assignment
 - get
Retrieve a role assignment
 - del
Delete a role assignment
- Organization Role Assignments [Beta]
 - get
List role assignments
 - post
Create a role assignment
 - get
Retrieve a role assignment
 - del
Delete a role assignment
- Project Role Assignments [Beta]
 - get
List role assignments
 - post
Create a role assignment
 - get
Retrieve a role assignment
 - del
Delete a role assignment
- Role Assignments
 - post
Bulk Delete Role Assignment
 - post
Create Role Assignments
 - get
List Role Assignments
 - post
Create Role Assignment
 - get

- Get Role Assignment
 - o del
Delete Role Assignment
 - o post
List Role Assignments by filter
 - o post
List Aggregated Role Assignments by filter
 - o post
List Role Assignments by scope filter
 - o post
Validate Role Assignment
- get
List role assignments
- post
Create a role assignment
- get
Retrieve a role assignment
- del
Delete a role assignment
- get
List role assignments
- post
Create a role assignment
- get
Retrieve a role assignment
- del
Delete a role assignment
- get
List role assignments
- post
Create a role assignment
- get
Retrieve a role assignment
- del
Delete a role assignment
- post
Bulk Delete Role Assignment
- post
Create Role Assignments
- get
List Role Assignments
- post
Create Role Assignment
- get
Get Role Assignment
- del
Delete Role Assignment
- post
List Role Assignments by filter
- post
List Aggregated Role Assignments by filter
- post
List Role Assignments by scope filter
- post
Validate Role Assignment
- - Accounts
 - o get
Gets an account
 - o get
Checks if immutable delegate is enabled for account
 - o put
Update Account Name
 - o put
Update Default Experience
 - AccountSetting
 - o get

- Get the AccountSetting by accountIdentifier
 - put
Updates account settings
 - get
Get the AccountSetting by accountIdentifier
- Access Control List
 - post
Check Permission
- AuditFilters
 - get
Get the list of Filters of type Audit satisfying the criteria (if any) in the request
 - put
Updates the Filter of type Audit
 - post
Creates a Filter
 - get
Gets a Filter of type Audit by identifier
 - del
Delete a Filter of type Audit by identifier
- Audit
 - post
List Audit Events
- EULA [Beta]
 - post
Sign an End User License Agreement
 - get
Validate specified agreement is signed or not
- Filter
 - get
List Filters
 - put
Update a Filter
 - post
Create a Filter
 - get
Return Filter Details
 - del
Delete a Filter
 - get
List Filters
 - put
Update a Filter
 - post
Create a Filter
 - get
Return Filter Details
 - del
Delete a Filter
 - get
List Filters
 - put
Update a Filter
 - post
Create a Filter
 - get
Return Filter Details
 - del
Delete a Filter
 - get
List Filters
 - put
Update a Filter
 - post
Create a Filter
 - get
Return Filter Details
 - del
Delete a Filter

- Delete a Filter
- Invite
 - put Resend invite
 - del Delete Invite
 - get Get Invite
 - get List Invites
 - post Get pending users
- IP Allowlist [Beta]
 - post Create a IP Allowlist config
 - get List IP Allowlist Configs
 - get Retrieve a IP Allowlist config
 - put Update IP Allowlist config
 - del Delete an IP Allowlist config
 - get Validate unique IP Allowlist config identifier
 - get Validate IP address lies in a specified range or not
- Oidc-Access-Token
 - post Generate an OIDC IAM Role Credential for AWS
 - post Generates an OIDC Service Account Access Token for GCP
 - post Generates an OIDC Workload Access Token for GCP
- Oidc-ID-Token
 - post Generates an OIDC ID Token for AWS
 - post Generates an OIDC ID Token for GCP
- OIDC
 - get Get the openid configuration for Harness
 - get Get the openid configuration for Harness
- Canny
 - post create Canny Post for given user
 - get Get a list of boards available on Canny
- ApiKey
 - get Fetches the list of API Keys corresponding to the request's filter criteria.
 - post Creates an API key
 - put Updates API Key for the provided ID
 - del Deletes the API Key corresponding to the provided ID.
 - get Fetches the API Keys details corresponding to the provided ID and Scope.
 - get Fetches the list of Aggregated API Keys corresponding to the request's filter criteria.
- Source Code Manager
 - put

- Updates Source Code Manager Details with the given Source Code Manager Id
 - o del
Deletes the Source Code Manager corresponding to the specified Source Code Manager Id
 - o get
Lists Source Code Managers for the given account
 - o post
Creates Source Code Manager
- Nextgen Ldap
 - o post
Test LDAP authentication
 - o get
Return Ldap groups matching name
- Harness Resource Type
 - o get
Gets all resource types available at this scope
- Authentication Settings
 - o get
Return configured Ldap settings for the account
 - o put
Updates Ldap setting
 - o post
Create Ldap setting
 - o del
Delete Ldap settings
 - o del
Delete SAML meta data
 - o del
Delete SAML meta data for given SAML sso id
 - o put
Update authentication enabled or not for given SAML setting
 - o get
Gets authentication settings for the given Account ID
 - o get
Gets authentication settings version 2 for the given Account ID
 - o get
Get password strength
 - o get
Test SAML connectivity
 - o get
Test SAML connectivity
 - o del
Delete OAuth Setting
 - o put
Enable/disable public access at account level
 - o put
Set session timeout at account level
 - o put
Set two factor authorization
 - o put
Update Auth mechanism
 - o put
Update Oauth providers
 - o put
Update SAML metadata
 - o post
Upload SAML metadata
 - o put
Update SAML metadata for a given SAML SSO Id
 - o put
Updates the whitelisted domains
- Permissions
 - o get
List Permissions
 - o get
List Resource Types
- Secret Managers
 - o post

Gets the metadata of Secret Manager

- Setting
 - get
Get a setting value by identifier
 - get
Get list of settings under the specified category
 - put
Update settings

- Service Account
 - get
Get Service Accounts
 - post
Create a Service Account
 - put
Update a Service Account
 - del
Delete a Service Account
 - get
Get Service Account In Scope
 - get
List aggregated Service Accounts

- SMTP
 - del
Delete Smtplib Config by identifier
 - get
Gets Smtplib config by accountId
 - put
Updates the Smtplib Config
 - post
Creates SMTP config
 - post
Tests the config's connectivity by sending a test email
 - post
Checks whether other connectors exist with the same name

- Token
 - post
Create a Token
 - put
Update a Token
 - del
Delete a Token
 - get
List all Tokens
 - post
Rotate a Token
 - post
Validate a Token

- User
 - post
Add user(s) to scope
 - put
Change user password
 - get
Check if user is last admin
 - put
Disable two factor authentication
 - get
Get detailed user information
 - post
Get list of users
 - get
Gets Two Factor Auth Settings
 - get
Get Current User Info
 - post
Get users list
 - put

- Update User
 - del Remove user from scope
 - get Reset two factor authorization
 - put Unlock user
 - put Enable two factor authentication
 - put Update User
- User Group
 - get Check user membership
 - put Add user to User Group
 - del Remove user from User Group
 - put Copy User Group
 - get List the User Groups in an account/org/project
 - put Update User Group
 - post Create User Group
 - get Get User Group
 - del Delete a User Group in an account/org/project
 - get Get Inheriting Child Scopes
 - post List users in User Group
 - put Link LDAP Group to the User Group to an account/org/project
 - put Link SAML Group to the User Group in an account/org/project
 - post List User Groups by filter
 - post Get filtered User Groups
 - put Unlink SSO Group from the User Group in an account/org/project
 - put Update User Group
 - post Create User Group
- Variables
 - get Fetches the list of Variables.
 - put Updates the Variable.
 - post Creates a Variable.
 - get Get the Variable by scope identifiers and variable identifier.
 - del Deletes Variable by ID.
- get Gets an account
- get Checks if immutable delegate is enabled for account
- put Update Account Name
- put Update Default Experience
- get

Get the AccountSetting by accountIdentifier

- put
Updates account settings
- get
Get the AccountSetting by accountIdentifier

- post
Check Permission

- get
Get the list of Filters of type Audit satisfying the criteria (if any) in the request

- put
Updates the Filter of type Audit

- post
Creates a Filter

- get
Gets a Filter of type Audit by identifier

- del
Delete a Filter of type Audit by identifier

- post
List Audit Events

- post
Sign an End User License Agreement

- get
Validate specified agreement is signed or not

- get
List Filters

- put
Update a Filter

- post
Create a Filter

- get
Return Filter Details

- del
Delete a Filter

- get
List Filters

- put
Update a Filter

- post
Create a Filter

- get
Return Filter Details

- del
Delete a Filter

- get
List Filters

- put
Update a Filter

- post
Create a Filter

- get
Return Filter Details

- del
Delete a Filter

- get
List Filters

- put
Update a Filter

- post
Create a Filter

- get
Return Filter Details

- del
Delete a Filter

- put
Resend invite
- del
Delete Invite

- get
Get Invite
- get
List Invites
- post
Get pending users
- post
Create a IP Allowlist config
- get
List IP Allowlist Configs
- get
Retrieve a IP Allowlist config
- put
Update IP Allowlist config
- del
Delete an IP Allowlist config
- get
Validate unique IP Allowlist config identifier
- get
Validate IP address lies in a specified range or not
- post
Generate an OIDC IAM Role Credential for AWS
- post
Generates an OIDC Service Account Access Token for GCP
- post
Generates an OIDC Workload Access Token for GCP
- post
Generates an OIDC ID Token for AWS
- post
Generates an OIDC ID Token for GCP
- get
Get the openid configuration for Harness
- get
Get the openid configuration for Harness
- post
create Canny Post for given user
- get
Get a list of boards available on Canny
- get
Fetches the list of API Keys corresponding to the request's filter criteria.
- post
Creates an API key
- put
Updates API Key for the provided ID
- del
Deletes the API Key corresponding to the provided ID.
- get
Fetches the API Keys details corresponding to the provided ID and Scope.
- get
Fetches the list of Aggregated API Keys corresponding to the request's filter criteria.
- put
Updates Source Code Manager Details with the given Source Code Manager Id
- del
Deletes the Source Code Manager corresponding to the specified Source Code Manager Id
- get
Lists Source Code Managers for the given account
- post
Creates Source Code Manager
- post
Test LDAP authentication
- get
Return Ldap groups matching name
- get
Gets all resource types available at this scope

- get
 - Return configured Ldap settings for the account
- put
 - Updates Ldap setting
- post
 - Create Ldap setting
- del
 - Delete Ldap settings
- del
 - Delete SAML meta data
- del
 - Delete SAML meta data for given SAML sso id
- put
 - Update authentication enabled or not for given SAML setting
- get
 - Gets authentication settings for the given Account ID
- get
 - Gets authentication settings version 2 for the given Account ID
- get
 - Get password strength
- get
 - Test SAML connectivity
- get
 - Test SAML connectivity
- del
 - Delete OAuth Setting
- put
 - Enable/disable public access at account level
- put
 - Set session timeout at account level
- put
 - Set two factor authorization
- put
 - Update Auth mechanism
- put
 - Update Oauth providers
- put
 - Update SAML metadata
- post
 - Upload SAML metadata
- put
 - Update SAML metadata for a given SAML SSO Id
- put
 - Updates the whitelisted domains
- get
 - List Permissions
- get
 - List Resource Types
- post
 - Gets the metadata of Secret Manager
- get
 - Get a setting value by identifier
- get
 - Get list of settings under the specified category
- put
 - Update settings
- get
 - Get Service Accounts
- post
 - Create a Service Account
- put
 - Update a Service Account
- del
 - Delete a Service Account
- get
 - Get Service Account In Scope
- get
 - List aggregated Service Accounts

- del
Delete Smtplib Config by identifier
 - get
Gets Smtplib config by accountId
 - put
Updates the Smtplib Config
 - post
Creates SMTP config
 - post
Tests the config's connectivity by sending a test email
 - post
Checks whether other connectors exist with the same name
 - post
Create a Token
 - put
Update a Token
 - del
Delete a Token
 - get
List all Tokens
 - post
Rotate a Token
 - post
Validate a Token
- post
Add user(s) to scope
 - put
Change user password
 - get
Check if user is last admin
 - put
Disable two factor authentication
 - get
Get detailed user information
 - post
Get list of users
 - get
Gets Two Factor Auth Settings
 - get
Get Current User Info
 - post
Get users list
 - put
Update User
 - del
Remove user from scope
 - get
Reset two factor authorization
 - put
Unlock user
 - put
Enable two factor authentication
 - put
Update User
- get
Check user membership
 - put
Add user to User Group
 - del
Remove user from User Group
 - put
Copy User Group
 - get
List the User Groups in an account/org/project
 - put
Update User Group
 - post
Create User Group
 - get
Get User Group

- **del**
Delete a User Group in an account/org/project
- **get**
Get Inheriting Child Scopes
- **post**
List users in User Group
- **put**
Link LDAP Group to the User Group to an account/org/project
- **put**
Link SAML Group to the User Group in an account/org/project
- **post**
List User Groups by filter
- **post**
Get filtered User Groups
- **put**
Unlink SSO Group from the User Group in an account/org/project
- **put**
Update User Group
- **post**
Create User Group

- **get**
Fetches the list of Variables.
- **put**
Updates the Variable.
- **post**
Creates a Variable.
- **get**
Get the Variable by scope identifiers and variable identifier.
- **del**
Deletes Variable by ID.

- Agent mTLS Endpoint Management
 - **get**
Gets the agent mTLS endpoint for an account.
 - **put**
Updates the existing agent mTLS endpoint for an account.
 - **post**
Creates the agent mTLS endpoint for an account.
 - **del**
Removes the agent mTLS endpoint for an account.
 - **patch**
Updates selected properties of the existing agent mTLS endpoint for an account.
 - **get**
Checks whether a given agent mTLS endpoint domain prefix is available.

- Delegate Download Resource
 - **post**
Downloads a docker delegate yaml file.
 - **post**
Downloads a kubernetes delegate yaml file.

- Delegate Group Tags Resource
 - **get**
Retrieves list of tags attached with Delegate group
 - **put**
Clears all existing tags with delegate group and attach given set of tags to delegate group.
 - **post**
Add given list of tags to the Delegate group
 - **del**
Deletes all tags from the Delegate group
 - **post**
List delegate groups that are having mentioned tags.

- Delegate Setup Resource
 - **del**
Deletes a Delegate by its identifier.
 - **get**
Generates delegate terraform example module file from the account
 - **post**
Generates helm values yaml file from the data specified in request body (Delegate setup details).

- post
Lists all delegates in NG filtered by provided conditions
 - get
Gets the latest supported delegate version. The version has YY.MM.XXXXX format. You can use any version lower than the returned results(upto 3 months old)
 - put
Overrides delegate image tag for account
- Delegate Token Resource
 - get
Retrieves Delegate Tokens by Account, Organization, Project and status.
 - put
Revokes Delegate Token.
 - post
Creates Delegate Token.
 - get
Lists delegate groups that are using the specified delegate token.
- get
Gets the agent mTLS endpoint for an account.
- put
Updates the existing agent mTLS endpoint for an account.
- post
Creates the agent mTLS endpoint for an account.
- del
Removes the agent mTLS endpoint for an account.
- patch
Updates selected properties of the existing agent mTLS endpoint for an account.
- get
Checks whether a given agent mTLS endpoint domain prefix is available.
- post
Downloads a docker delegate yaml file.
- post
Downloads a kubernetes delegate yaml file.
- get
Retrieves list of tags attached with Delegate group
- put
Clears all existing tags with delegate group and attach given set of tags to delegate group.
- post
Add given list of tags to the Delegate group
- del
Deletes all tags from the Delegate group
- post
List delegate groups that are having mentioned tags.
- del
Deletes a Delegate by its identifier.
- get
Generates delegate terraform example module file from the account
- post
Generates helm values yaml file from the data specified in request body (Delegate setup details).
- post
Lists all delegates in NG filtered by provided conditions
- get
Gets the latest supported delegate version. The version has YY.MM.XXXXX format. You can use any version lower than the returned results(upto 3 months old)
- put
Overrides delegate image tag for account
- get
Retrieves Delegate Tokens by Account, Organization, Project and status.
- put
Revokes Delegate Token.
- post
Creates Delegate Token.
- get
Lists delegate groups that are using the specified delegate token.
- Pipelines [Beta]
 - get

- List Pipelines
 - post Create a Pipeline
 - get Retrieve a Pipeline
 - put Update a Pipeline
 - del Delete a Pipeline
 - patch Patch API for pipeline
 - post Move Pipeline YAML from inline to remote
 - post Get Pipeline YAML from Git Repository
 - put Update GitMetadata for Remote Pipelines
- Input Sets [Beta]
 - post Create an Input Set
 - get List Input Sets
 - get Retrieve an Input Set
 - put Update an Input Set
 - del Delete an Input Set
 - post Move InputSet YAML from inline to remote
 - post Get Input Set YAML from Git Repository
 - put Update GitMetadata for Remote InputSet
- Approvals
 - get Gets Approval Instances by Execution Id
 - post Approve or Reject an Execution by Pipeline Execution ID
 - post Approve or Reject a Pipeline Execution
- Pipeline Execution [Beta]
 - post Execute Pipeline
 - get Get Stages execution List for a given Pipeline execution.
 - post Re-run Stages Execution of a Pipeline
 - post Execute given Stages of a Pipeline
- Pipeline Dashboard
 - get Fetch Execution Details for an Interval
- Pipeline Input Set
 - get List Input Sets
 - post Create an Input Set
 - post Create an Overlay Input Set for a pipeline
 - get Fetch an Input Set
 - put Update an Input Set
 - del Delete an Input Set
 - get

- Gets an Overlay Input Set by identifier
 - put Update an Overlay Input Set for a pipeline
 - post Fetch Runtime Input Template
 - put Update git-metadata in remote input-set
- Pipeline
 - post Create a Pipeline
 - post Create a Pipeline
 - get Fetch a Pipeline
 - put Update a Pipeline
 - del Delete a Pipeline
 - post List Pipelines
 - get Fetch Pipeline Summary
 - post Import and Create Pipeline from Git Repository
 - post Import and Create Pipeline from Git Repository
 - put Update git-metadata in remote pipeline Entity
 - put Update a Pipeline
- Pipeline Execution Details
 - get Fetch Execution Details
 - get Fetch Execution Details
 - get Fetch Execution SubGraph for a Given NodeExecution ID
 - get Get the Input Set YAML used for given Plan Execution
 - post List Execution Identifier
 - post List Executions
 - get Get Notes for a pipelineExecution
 - put Updates Notes for a pipelineExecution
- Pipeline Execute
 - get Retry History for a given execution
 - put Execute an Interrupt
 - put Handles the interrupt for a given stage in a pipeline
 - post Retry a executed pipeline with inputSet pipeline yaml
 - post Execute a Pipeline with Input Set References
 - post Execute a Pipeline with Runtime Input YAML
 - post Execute given Stages of a Pipeline
- Pipeline Refresh
 - get Validates template inputs in a pipeline's YAML specification.
- Triggers [Beta]
 - get

- Gets the paginated list of triggers for accountIdentifier, orgIdentifier, projectIdentifier, targetIdentifier.
- post
Creates Trigger for triggering target pipeline identifier.
- get
Gets the trigger by accountIdentifier, orgIdentifier, projectIdentifier, targetIdentifier and triggerIdentifier.
- put
Updates trigger for pipeline with target pipeline identifier.
- del
Deletes Trigger by identifier.
- get
Lists all Triggers
- get
Fetches Trigger details for a specific accountIdentifier, orgIdentifier, projectIdentifier, targetIdentifier, triggerIdentifier.
- get
Get event history for a trigger
- TriggersEvents
 - get
Get all the polled response for a given trigger
 - get
Get event history for a trigger
 - get
Get Trigger history event correlation
 - get
Get Trigger history event correlation V2
 - get
Get artifact and manifest trigger event history based on build source type
- Webhook Triggers
 - get
Gets webhook event processing details for input eventId.
 - get
Gets webhook event processing details for input eventId.
 - post
Handles event payload for webhook triggers.
 - post
Handles event payload for custom webhook triggers.
 - post
Handles event payload for custom webhook triggers.
 - post
Handles event payload for custom webhook triggers.
- Webhook Event Handler
 - post
Process event payload for webhook triggers.
- get
List Pipelines
- post
Create a Pipeline
- get
Retrieve a Pipeline
- put
Update a Pipeline
- del
Delete a Pipeline
- patch
Patch API for pipeline
- post
Move Pipeline YAML from inline to remote
- post
Get Pipeline YAML from Git Repository
- put
Update GitMetadata for Remote Pipelines
- post
Create an Input Set
- get
List Input Sets
- get
Retrieve an Input Set
- put

Update an Input Set

- del
Delete an Input Set
 - post
Move InputSet YAML from inline to remote
 - post
Get Input Set YAML from Git Repository
 - put
Update GitMetadata for Remote InputSet
-
- get
Gets Approval Instances by Execution Id
 - post
Approve or Reject an Execution by Pipeline Execution ID
 - post
Approve or Reject a Pipeline Execution
-
- post
Execute Pipeline
 - get
Get Stages execution List for a given Pipeline execution.
 - post
Re-run Stages Execution of a Pipeline
 - post
Execute given Stages of a Pipeline
-
- get
Fetch Execution Details for an Interval
-
- get
List Input Sets
 - post
Create an Input Set
 - post
Create an Overlay Input Set for a pipeline
 - get
Fetch an Input Set
 - put
Update an Input Set
 - del
Delete an Input Set
 - get
Gets an Overlay Input Set by identifier
 - put
Update an Overlay Input Set for a pipeline
 - post
Fetch Runtime Input Template
 - put
Update git-metadata in remote input-set
-
- post
Create a Pipeline
 - post
Create a Pipeline
 - get
Fetch a Pipeline
 - put
Update a Pipeline
 - del
Delete a Pipeline
 - post
List Pipelines
 - get
Fetch Pipeline Summary
 - post
Import and Create Pipeline from Git Repository
 - post
Import and Create Pipeline from Git Repository
 - put
Update git-metadata in remote pipeline Entity
 - put
Update a Pipeline

- **get**
Fetch Execution Details
- **get**
Fetch Execution Details
- **get**
Fetch Execution SubGraph for a Given NodeExecution ID
- **get**
Get the Input Set YAML used for given Plan Execution
- **post**
List Execution Identifier
- **post**
List Executions
- **get**
Get Notes for a pipelineExecution
- **put**
Updates Notes for a pipelineExecution

- **get**
Retry History for a given execution
- **put**
Execute an Interrupt
- **put**
Handles the interrupt for a given stage in a pipeline
- **post**
Retry a executed pipeline with inputSet pipeline yaml
- **post**
Execute a Pipeline with Input Set References
- **post**
Execute a Pipeline with Runtime Input YAML
- **post**
Execute given Stages of a Pipeline

- **get**
Validates template inputs in a pipeline's YAML specification.

- **get**
Gets the paginated list of triggers for accountIdentifier, orgIdentifier, projectIdentifier, targetIdentifier.
- **post**
Creates Trigger for triggering target pipeline identifier.
- **get**
Gets the trigger by accountIdentifier, orgIdentifier, projectIdentifier, targetIdentifier and triggerIdentifier.
- **put**
Updates trigger for pipeline with target pipeline identifier.
- **del**
Deletes Trigger by identifier.
- **get**
Lists all Triggers
- **get**
Fetches Trigger details for a specific accountIdentifier, orgIdentifier, projectIdentifier, targetIdentifier, triggerIdentifier.
- **get**
Get event history for a trigger

- **get**
Get all the polled response for a given trigger
- **get**
Get event history for a trigger
- **get**
Get Trigger history event correlation
- **get**
Get Trigger history event correlation V2
- **get**
Get artifact and manifest trigger event history based on build source type

- **get**
Gets webhook event processing details for input eventId.
- **get**
Gets webhook event processing details for input eventId.
- **post**
Handles event payload for webhook triggers.
- **post**
Handles event payload for custom webhook triggers.
- **post**
Handles event payload for custom webhook triggers.

- post
 - Handles event payload for custom webhook triggers.
- post
 - Process event payload for webhook triggers.
- K8s Release Service Mapping [Beta]
 - get
 - List service and environment details using namespace and releasesname
- CustomDeployment
 - post
 - Gets Custom Deployment Entity References
 - post
 - Gets Custom Deployment Expression Variables
 - get
 - Gets Infra Variables from a Custom Deployment Template by identifier
 - post
 - Return the updated yaml for infrastructure based on Deployment template
 - get
 - This validates whether Infrastructure is valid or not
- Environments
 - get
 - Gets Environment list for a project
 - put
 - Update an Environment by identifier
 - post
 - Create an Environment
 - get
 - Gets an Environment by identifier
 - del
 - Delete an Environment by identifier
 - get
 - Gets Service Overrides list
 - post
 - upsert a Service Override for an Environment
 - del
 - Delete a ServiceOverride entity
 - get
 - Gets Environment Access list
 - post
 - Move environment YAML from inline to remote
 - put
 - Upsert an Environment by identifier
- EnvironmentGroup
 - post
 - Create an Environment Group
 - get
 - Gets an Environment Group by identifier
 - put
 - Update an Environment Group by Identifier
 - del
 - Delete en Environment Group by Identifier
 - post
 - Gets Environment Group list
- Infrastructures
 - get
 - Gets Infrastructure list
 - put
 - Update an Infrastructure by identifier
 - post
 - Create an Infrastructure in an Environment
 - get
 - Gets an Infrastructure by identifier
 - del
 - Delete an Infrastructure by identifier
 - post
 - Move infra YAML from inline to remote

- Usage
 - get
Download CSV Active Services report
 - get
Gets License Usage By Module, Timestamp, and Account Identifier
 - get
Gets License Usage By Module, Timestamp, and Account Identifier
 - get
Gets License Usage By Module, Timestamp, and Account Identifier
 - get
Download CSV Active Services report
 - get
Gets License Usage By Module, Timestamp, and Account Identifier
 - get
Gets License Usage By Module, Timestamp, and Account Identifier
 - get
Gets License Usage By Module, Timestamp, and Account Identifier
 - get
Download CSV Active Monitored Services report
 - get
Download CSV Active Services Monitored report
 - get
CvgetLicenseUsage
 - get
getSRMLicenseUsage
 - post
Returns a List of active monitored services along with identifier,Active Monitored Services Count and other details
- File Store
 - get
List Files and Folders metadata
 - post
Create Folder or File including content
 - post
Creates File or Folder metadata via YAML
 - get
Get the Folder or File metadata
 - put
Update Folder or File including content
 - del
Delete File or Folder by identifier
 - get
Download File
 - get
Get list of created by user details
 - get
Get file content of scopedFilePath
 - get
Get list of entities where file is referenced by queried entity type
 - get
Get the list of supported entity types for files
 - post
Get filtered list of Files or Folders
 - post
Get folder nodes at first level, not including sub-nodes
 - put
Update File or Folder metadata via YAML
- Service Dashboard
 - get
Get pipeline execution count for a service with grouping support on artifact and deployment status
- ServiceOverrides
 - put
Update an ServiceOverride Entity
 - post
Create an ServiceOverride Entity
 - get
Gets Service Overrides by Identifier
 - del
Delete a ServiceOverride entity

- tas
 - get
 - Return the Tas organizations
 - get
 - Return the Tas spaces
 - get
 - Return the Tas spaces
- get
 - List service and environment details using namespace and releasename
- post
 - Gets Custom Deployment Entity References
- post
 - Gets Custom Deployment Expression Variables
- get
 - Gets Infra Variables from a Custom Deployment Template by identifier
- post
 - Return the updated yaml for infrastructure based on Deployment template
- get
 - This validates whether Infrastructure is valid or not
- get
 - Gets Environment list for a project
- put
 - Update an Environment by identifier
- post
 - Create an Environment
- get
 - Gets an Environment by identifier
- del
 - Delete an Environment by identifier
- get
 - Gets Service Overrides list
- post
 - upsert a Service Override for an Environment
- del
 - Delete a ServiceOverride entity
- get
 - Gets Environment Access list
- post
 - Move environment YAML from inline to remote
- put
 - Upsert an Environment by identifier
- post
 - Create an Environment Group
- get
 - Gets an Environment Group by identifier
- put
 - Update an Environment Group by Identifier
- del
 - Delete en Environment Group by Identifier
- post
 - Gets Environment Group list
- get
 - Gets Infrastructure list
- put
 - Update an Infrastructure by identifier
- post
 - Create an Infrastructure in an Environment
- get
 - Gets an Infrastructure by identifier
- del
 - Delete an Infrastructure by identifier
- post
 - Move infra YAML from inline to remote
- get
 - Download CSV Active Services report
- get

- **get**
Gets License Usage By Module, Timestamp, and Account Identifier
- **get**
Gets License Usage By Module, Timestamp, and Account Identifier
- **get**
Gets License Usage By Module, Timestamp, and Account Identifier
- **get**
Download CSV Active Services report
- **get**
Gets License Usage By Module, Timestamp, and Account Identifier
- **get**
Gets License Usage By Module, Timestamp, and Account Identifier
- **get**
Gets License Usage By Module, Timestamp, and Account Identifier
- **get**
Download CSV Active Monitored Services report
- **get**
Download CSV Active Services Monitored report
- **get**
`CvgetLicenseUsage`
- **get**
`getSRMLicenseUsage`
- **post**
Returns a List of active monitored services along with identifier,Active Monitored Services Count and other details
- **get**
List Files and Folders metadata
- **post**
Create Folder or File including content
- **post**
Creates File or Folder metadata via YAML
- **get**
Get the Folder or File metadata
- **put**
Update Folder or File including content
- **del**
Delete File or Folder by identifier
- **get**
Download File
- **get**
Get list of created by user details
- **get**
Get file content of scopedFilePath
- **get**
Get list of entities where file is referenced by queried entity type
- **get**
Get the list of supported entity types for files
- **post**
Get filtered list of Files or Folders
- **post**
Get folder nodes at first level, not including sub-nodes
- **put**
Update File or Folder metadata via YAML
- **get**
Get pipeline execution count for a service with grouping support on artifact and deployment status
- **put**
Update an ServiceOverride Entity
- **post**
Create an ServiceOverride Entity
- **get**
Gets Service Overrides by Identifier
- **del**
Delete a ServiceOverride entity
- **get**
Return the Tas organizations
- **get**
Return the Tas spaces
- **get**
Return the Tas spaces
-

Freeze CRUD

- post
 Create a Freeze
- get
 Get a Freeze
- put
 Updates a Freeze
- del
 Delete a Freeze
- post
 Delete many Freezes
- post
 Gets Freeze list
- get
 Get list of freeze acted on a frozen execution
- get
 Get Global Freeze Yaml
- post
 Create Global Freeze
- post
 Update the status of Freeze to active or inactive

- Freeze Evaluation
- Freeze Schema

- post
 Create a Freeze
- get
 Get a Freeze
- put
 Updates a Freeze
- del
 Delete a Freeze
- post
 Delete many Freezes
- post
 Gets Freeze list
- get
 Get list of freeze acted on a frozen execution
- get
 Get Global Freeze Yaml
- post
 Create Global Freeze
- post
 Update the status of Freeze to active or inactive

- Account Services [Beta]

- get
 Retrieve a service
- put
 Update service
- del
 Delete a service
- get
 List services
- post
 Create a service

- Organization Services [Beta]

- get
 Retrieve a service
- put
 Update Service
- del
 Delete a service
- get
 List Services
- post
 Create a service

- Project Services [Beta]

- get

- Retrieve a service
 - put
 - Update Service
 - del
 - Delete a Service
 - get
 - List Services
 - post
 - Create a Service
- Services
 - get
 - Gets Service list
 - put
 - Update a Service by identifier
 - post
 - Create a Service
 - post
 - Create Services
 - get
 - Gets a Service by identifier
 - del
 - Delete a Service by identifier
 - get
 - Retrieving the list of Kubernetes Command Options
 - get
 - Retrieving the list of Kustomize Command Flags
 - get
 - Retrieving the list of actions available for service hooks
 - get
 - Gets Service Access list
 - put
 - Upsert a Service by identifier
- get
 - Retrieve a service
- put
 - Update service
- del
 - Delete a service
- get
 - List services
- post
 - Create a service
- get
 - Retrieve a service
- put
 - Update Service
- del
 - Delete a service
- get
 - List Services
- post
 - Create a service
- get
 - Retrieve a service
- put
 - Update Service
- del
 - Delete a Service
- get
 - List Services
- post
 - Create a Service
- get
 - Gets Service list
- put
 - Update a Service by identifier
- post
 - Create a Service

- post
Create Services
- get
Gets a Service by identifier
- del
Delete a Service by identifier
- get
Retrieving the list of Kubernetes Command Options
- get
Retrieving the list of Kustomize Command Flags
- get
Retrieving the list of actions available for service hooks
- get
Gets Service Access list
- put
Upsert a Service by identifier
- Account Rancher Infrastructure [Beta]
 - get
List rancher clusters using account level connector
 - get
List rancher clusters using account level env and infra def
- Org Rancher Infrastructure [Beta]
 - get
List rancher clusters using org level connector
 - get
List rancher clusters using org level env and infra def
- Project Rancher Infrastructure [Beta]
 - get
List rancher clusters using project level connector
 - get
List rancher clusters using project level env and infra def
- get
List rancher clusters using account level connector
- get
List rancher clusters using account level env and infra def
- get
List rancher clusters using org level connector
- get
List rancher clusters using org level env and infra def
- get
List rancher clusters using project level connector
- get
List rancher clusters using project level env and infra def
- Account Templates [Beta]
 - post
Create Template
 - get
Get Templates List
 - put
Update Git details
 - get
Get Stable Template
 - get
Retrieve a Template
 - put
Update Template
 - del
Delete Template
 - put
Update Stable Template
 - post
Import Template
- Organization Templates [Beta]

- put
 - Update Git details
 - post
 - Create Template
 - get
 - Get Templates List
 - get
 - Get Stable Template
 - get
 - Retrieve a Template
 - put
 - Update Template
 - del
 - Delete Template
 - put
 - Update Stable Template
 - post
 - Import template
- Project Templates [Beta]
 - put
 - Update Git details
 - post
 - Create Template
 - get
 - Get Templates List
 - get
 - Get Stable Template
 - get
 - Retrieve a Template
 - put
 - Update Template
 - del
 - Delete Template
 - put
 - Update Stable Template
 - post
 - Import Template
- Templates
 - post
 - Get YAML with updated Template Inputs
 - get
 - Validate Template Inputs in a YAML
 - post
 - Create a Template
 - del
 - Delete Template Version
 - get
 - Get Template
 - get
 - Gets Template Input Set YAML
 - post
 - Gets all metadata of template list
 - post
 - Move Template YAML from inline to remote
 - put
 - Update Template Version
 - post
 - Update git metadata details for a remote template
 - put
 - Update Stable Template Version
- Global Templates
 - get
 - Gets Global Template Input Set YAML
- post
 - Create Template
- get
 - Get Templates List
- put

- Update Git details
 - get Get Stable Template
 - get Retrieve a Template
 - put Update Template
 - del Delete Template
 - put Update Stable Template
 - post Import Template
- put Update Git details
 - post Create Template
 - get Get Templates List
 - get Get Stable Template
 - get Retrieve a Template
 - put Update Template
 - del Delete Template
 - put Update Stable Template
 - post Import template
- put Update Git details
 - post Create Template
 - get Get Templates List
 - get Get Stable Template
 - get Retrieve a Template
 - put Update Template
 - del Delete Template
 - put Update Stable Template
 - post Import Template
- post Get YAML with updated Template Inputs
 - get Validate Template Inputs in a YAML
 - post Create a Template
 - del Delete Template Version
 - get Get Template
 - get Gets Template Input Set YAML
 - post Gets all metadata of template list
 - post Move Template YAML from inline to remote
 - put Update Template Version
 - post Update git metadata details for a remote template
 - put

Update Stable Template Version

- get
Gets Global Template Input Set YAML
- Agents
 - get
AgentServiceForServer_List
 - post
AgentServiceForServer_Create
 - post
AgentServiceForServer_Search
 - put
AgentServiceForServer_Update
 - get
AgentServiceForServer_GetDeployYaml
 - post
AgentServiceForServer_PostDeployHelmChart
 - post
AgentServiceForServer_PostDeployYaml
 - get
AgentServiceForServer_GetDeployHelmChart
 - patch
AgentServiceForServer_SetPrimaryNode
 - get
AgentServiceForServer_Get
 - del
AgentServiceForServer_Delete
 - post
AgentServiceForServer_RegenerateCredentials
 - post
AgentServiceForServer_Scale
 - get
AgentServiceForServer_Unique
- Application
 - post
ApplicationService_ListApps
- Applications
 - get
List returns list of applications for a specific agent
 - post
Create creates an application
 - get
ManagedResources returns list of managed resources
 - get
Get returns an application by name
 - get
ListResourceEvents returns a list of event resources
 - get
PodLogs returns stream of log entries for the specified pod(s).
 - get
GetManifests returns application manifests
 - get
PodLogs returns stream of log entries for the specified pod(s).
 - get
ResourceTree returns resource tree
 - get
Get the meta-data (author, date, tags, message) for a specific revision of the application
 - get
Get returns sync windows of the application
 - put
Update updates an application
 - del
Delete deletes an application
 - patch
Patch patch an application
 - del
TerminateOperation terminates the currently running operation
 - get

- GetResource returns single application resource
 - o del
 - DeleteResource deletes a single application resource
 - o post
 - PatchResource patch single application resource
 - o get
 - ListResourceActions returns list of resource actions
 - o post
 - RunResourceAction run resource action
 - o post
 - Rollback syncs an application to its target stateHarness Event type (rollback)
 - o put
 - UpdateSpec updates an application spec
 - o post
 - Sync syncs an application to its target stateHarness Event type (deploy)
 - o get
 - Watch returns stream of application change events
 - o get
 - WatchResourceTree returns stream of application resource tree
 - o post
 - ListNs returns list of namespaces for a query.
 - o post
 - List returns list of application sync status
 - o get
 - Checks whether an app with the given name exists
- Certificates
 - o get
 - List returns list of certificates
- Clusters
 - o get
 - Gets a Cluster by identifier
 - o del
 - Delete a Cluster by identifier
 - o get
 - Gets cluster list
 - o post
 - link a Cluster
 - o post
 - Link Clusters
 - o post
 - Unlink Clusters
 - o get
 - List returns list of clusters
 - o post
 - Create creates a cluster
 - o get
 - Get returns a cluster by identifier
 - o del
 - Delete deletes a cluster
 - o put
 - Update updates a cluster
 - o get
 - List all available repository certificates
 - o post
 - CreateHosted creates a harness hosted cluster
 - o post
 - List returns list of Clusters
 - o get
 - Checks for whether the cluster exists
- Dashboard Aggregates
 - o post
 - Returns aggregate statistics of recent deployments
 - o get
 - List phase status counts for top 5 most deployed apps
- Dashboards
 - o get
 - List count of Cluster, Repos and Apps created within a time series
 - o get

- GetDashboradOverview gets dashboard overview
- GnuPG Keys
 - post
Create one or more GPG public keys in the server's configuration
 - get
Get information about specified GPG public key from the server
 - del
Delete specified GPG public key from the server's configuration
- GPG Keys
 - get
List all available repository certificates
- Hosts
 - post
Gets the list of hosts filtered by accountIdentifier and connectorIdentifier
 - post
Validates hosts connectivity credentials
- Project mappings
 - get
AppProjectMappingService_GetAppProjectMappingListByAgent
 - post
CreateAppProjectMapping creates a new mapping between Harness Project and argo project
 - del
Delete an argo project to harness project mapping
 - get
AppProjectMappingService_GetAppProjectMappingList
 - post
CreateAppProjectMapping creates a new mapping between Harness Project and argo project
 - get
AppProjectMappingService_GetAppProjectMappingV2
 - del
Delete an argo project to harness project mapping
 - put
CreateAppProjectMapping creates a new mapping between Harness Project and argo project
 - get
V2
- Projects
 - get
List returns list of projects
 - post
Create a new project
 - get
Get returns a project by name
 - del
Delete deletes a project
 - put
Update updates a project
- Reconciler
 - post
Returns number of entities that exist in the cluster on the agent. Filter can be used to count only global entities (with empty project) and those specified by the filter.
 - post
Imports data from cluster via agent. There must be at least one project mapping in the database. Returns number of entities imported.
- Repositories
 - get
ListRepositories gets a list of all configured repositories
 - post
CreateRepository creates a new repository configuration
 - get
Checks whether External Secrets Operator is installed
 - get
Returns a list of ESO generators installed in agent namespace.
 - post
Returns the Repository type of OCI repo
 - post

- ValidateAccess gets connection state for a repository
 - o get
 - Get returns a repository or its credentials
 - o del
 - DeleteRepository deletes a repository from the configuration
 - o put
 - UpdateRepository updates a repository configuration
 - o get
 - GetAppDetails returns application details by given path
 - o get
 - ListApps returns list of apps in the repo
 - o get
 - GetHelmCharts returns list of helm charts in the specified repository
 - o get
 - Returns a list of refs (e.g. branches and tags) in the repo
 - o get
 - List returns list of Repositories by repository credential template
 - o post
 - List returns list of Repositories
 - o get
 - Checks whether a repository with the given name exists
- Repository Certificates
 - o get
 - List all available repository certificates
 - o del
 - Delete the certificates that match the RepositoryCertificateQuery
 - o post
 - Creates repository certificates on the server
- Repository credentials
 - o post
 - Create creates a new repository credential
 - o post
 - Get returns a repository credential given its url
 - o get
 - Get returns a repository credential given its identifier
 - o del
 - Delete deletes a repository credential
 - o put
 - Update updates a repository credential
 - o post
 - List repository credentials
- ValidateHost
 - o post
 - Validates hosts connectivity credentials
- get
 - AgentServiceForServer_List
- post
 - AgentServiceForServer_Create
- post
 - AgentServiceForServer_Search
- put
 - AgentServiceForServer_Update
- get
 - AgentServiceForServer_GetDeployYaml
- post
 - AgentServiceForServer_PostDeployHelmChart
- post
 - AgentServiceForServer_PostDeployYaml
- get
 - AgentServiceForServer_GetDeployHelmChart
- patch
 - AgentServiceForServer_SetPrimaryNode
- get
 - AgentServiceForServer_Get
- del
 - AgentServiceForServer_Delete
- post
 - AgentServiceForServer_RegenerateCredentials

- post
 - AgentServiceForServer_Scale
- get
 - AgentServiceForServer_Under
- post
 - ApplicationService_ListApps
- get
 - List returns list of applications for a specific agent
- post
 - Create creates an application
- get
 - ManagedResources returns list of managed resources
- get
 - Get returns an application by name
- get
 - ListResourceEvents returns a list of event resources
- get
 - PodLogs returns stream of log entries for the specified pod(s).
- get
 - GetManifests returns application manifests
- get
 - PodLogs returns stream of log entries for the specified pod(s).
- get
 - ResourceTree returns resource tree
- get
 - Get the meta-data (author, date, tags, message) for a specific revision of the application
- get
 - Get returns sync windows of the application
- put
 - Update updates an application
- del
 - Delete deletes an application
- patch
 - Patch patch an application
- del
 - TerminateOperation terminates the currently running operation
- get
 - GetResource returns single application resource
- del
 - DeleteResource deletes a single application resource
- post
 - PatchResource patch single application resource
- get
 - ListResourceActions returns list of resource actions
- post
 - RunResourceAction run resource action
- post
 - Rollback syncs an application to its target stateHarness Event type (rollback)
- put
 - UpdateSpec updates an application spec
- post
 - Sync syncs an application to its target stateHarness Event type (deploy)
- get
 - Watch returns stream of application change events
- get
 - WatchResourceTree returns stream of application resource tree
- post
 - ListNs returns list of namespaces for a query.
- post
 - List returns list of application sync status
- get
 - Checks whether an app with the given name exists
- get
 - List returns list of certificates
- get
 - Gets a Cluster by identifier
- del
 - Delete a Cluster by identifier
- get

- Gets cluster list
- post
link a Cluster
- post
Link Clusters
- post
Unlink Clusters
- get
List returns list of clusters
- post
Create creates a cluster
- get
Get returns a cluster by identifier
- del
Delete deletes a cluster
- put
Update updates a cluster
- get
List all available repository certificates
- post
CreateHosted creates a harness hosted cluster

- post
List returns list of Clusters
- get
Checks for whether the cluster exists

- post
Returns aggregate statistics of recent deployments
- get
List phase status counts for top 5 most deployed apps

- get
List count of Cluster, Repos and Apps created within a time series
- get
GetDashboradOverview gets dashboard overview

- post
Create one or more GPG public keys in the server's configuration
- get
Get information about specified GPG public key from the server
- del
Delete specified GPG public key from the server's configuration
- get
List all available repository certificates

- post
Gets the list of hosts filtered by accountIdentifier and connectorIdentifier
- post
Validates hosts connectivity credentials

- get
AppProjectMappingService _GetAppProjectMappingListByAgent
- post
CreateAppProjectMapping creates a new mapping between Harness Project and argo project
- del
Delete an argo project to harness project mapping
- get
AppProjectMappingService _GetAppProjectMappingList
- post
CreateAppProjectMapping creates a new mapping between Harness Project and argo project
- get
AppProjectMappingService _GetAppProjectMappingV2
- del
Delete an argo project to harness project mapping
- put
CreateAppProjectMapping creates a new mapping between Harness Project and argo project
- get
V2
- get
List returns list of projects
- post

- Create a new project
 - get
 - Get returns a project by name
 - del
 - Delete deletes a project
 - put
 - Update updates a project
- post
 - Returns number of entities that exist in the cluster on the agent. Filter can be used to count only global entities (with empty project) and those specified by the filter.
- post
 - Imports data from cluster via agent. There must be at least one project mapping in the database. Returns number of entities imported.
- get
 - ListRepositories gets a list of all configured repositories
- post
 - CreateRepository creates a new repository configuration
- get
 - Checks whether External Secrets Operator is installed
- get
 - Returns a list of ESO generators installed in agent namespace.
- post
 - Returns the Repository type of OCI repo
- post
 - ValidateAccess gets connection state for a repository
- get
 - Get returns a repository or its credentials
- del
 - DeleteRepository deletes a repository from the configuration
- put
 - UpdateRepository updates a repository configuration
- get
 - GetAppDetails returns application details by given path
- get
 - ListApps returns list of apps in the repo
- get
 - GetHelmCharts returns list of helm charts in the specified repository
- get
 - Returns a list of refs (e.g. branches and tags) in the repo
- get
 - List returns list of Repositories by repository credential template
- post
 - List returns list of Repositories
- get
 - Checks whether a repository with the given name exists
- get
 - List all available repository certificates
- del
 - Delete the certificates that match the RepositoryCertificateQuery
- post
 - Creates repository certificates on the server
- post
 - Create creates a new repository credential
- post
 - Get returns a repository credential given its url
- get
 - Get returns a repository credential given its identifier
- del
 - Delete deletes a repository credential
- put
 - Update updates a repository credential
- post
 - List repository credentials
- post
 - Validates hosts connectivity credentials
- Cloud Cost Anomalies

- post
List Anomalies
- post
Returns the list of distinct values for all the specified Anomaly fields.
- post
List Anomalies
- post
List Anomalies for Perspective
- put
Report Anomaly feedback
- Cloud Cost BI Dashboards
 - get
List all the BI Dashboards for CCM
- Cloud Cost Budget Groups
 - get
Fetch Budget group details
 - put
Update an existing budget group
 - del
Delete a budget group
 - post
Get aggregated amount for given budget groups/budgets
 - get
Get list of budget and budget group summaries
 - get
List all the Budget groups
 - post
Create a Budget Group
- Cloud Cost Budgets
 - get
Fetch Budget details
 - put
Update an existing budget
 - post
Clone a budget
 - del
Delete a budget
 - get
Fetch the cost details of a Budget
 - get
List all the Budgets associated with a Perspective
 - get
List all the Budgets
 - post
Create a Budget
- Cloud Cost K8S Connectors Metadata
 - post
Get CCM K8S Metadata
- Cloud Cost Details
 - post
Returns an overview of the cost
 - post
Returns cluster data in a tabular format
 - post
Returns cost details in a tabular format
 - post
Returns cost details in a time series format
- Cloud Cost Overview
 - get
Fetch high level overview details about CCM feature.
- Cloud Cost Cost Categories
 - get
Fetch details of a Cost category
 - del
Delete a Cost category

- get
 - Return details of all the Cost categories
 - put
 - Update a Cost category
 - post
 - Create Cost category
- RuleEnforcement
 - post
 - Add a new rule Enforcement
 - post
 - Fetch Rule Enforcement count for account
 - post
 - Fetch Rule Enforcement execution details for account
 - post
 - Fetch Rule Enforcements for account
- Rule
 - post
 - Clone a rule
 - del
 - Delete a rule
 - post
 - Enqueues job for execution
 - get
 - Get Schema for entity
 - post
 - Fetch rules for account
 - get
 - connectors with governance enabled and valid permission
 - put
 - Update a Rule
 - post
 - Validate a rule
- Cloud Cost Perspectives Folders
 - post
 - Create a Perspective folder
 - del
 - Delete a folder
 - get
 - Fetch folders for an account
 - put
 - Update a folder
 - get
 - Return details of all the Perspectives
 - post
 - Move a Perspective
- Cloud Cost Perspective Reports
 - get
 - Fetch details of a cost Report
 - put
 - Update a cost Perspective Report
 - post
 - Create a schedule for a Report
 - del
 - Delete cost Perspective report
- Cloud Cost Perspectives
 - get
 - Fetch details of a Perspective
 - put
 - Update a Perspective
 - post
 - Create a Perspective
 - del
 - Delete a Perspective
 - get
 - Return details of all the Perspectives
 - get
 - Get the last period cost for a Perspective

- get
Get the last twelve month cost for a Perspective
- Cloud Cost Recommendations Details
 - get
Return Azure VM Recommendation
 - get
Return EC2 Recommendation
 - get
Return ECS Recommendation
 - get
Return node pool Recommendation
 - get
Return workload Recommendation
- Cloud Cost Recommendations
 - post
Return void
 - post
Return the number of Recommendations
 - post
Return the list of filter values for the Recommendations
 - post
Return the list of Recommendations
 - post
Return Recommendations statistics Grouped on Resource Type
 - post
Return Recommendations statistics
- Cloud Cost Recommendation Jira
 - post
Create jira for recommendation
- Cloud Cost Recommendation Servicenow
 - post
Create servicenow ticket for recommendation
- Cloud Cost Recommendation Ignore List
 - post
Add resources to recommendations ignore list
 - get
Get resources in recommendations ignore list
 - post
Remove resources from recommendations ignore list
- Cloud Cost AutoStopping Rules
 - get
List AutoStopping Rules
 - post
Create an AutoStopping Rule
 - get
Return AutoStopping Rule details
 - del
Delete an AutoStopping Rule
 - get
Return health status of an AutoStopping Rule
 - get
Return savings details for an AutoStopping Rule
 - get
List all the resources for an AutoStopping Rule
 - get
Return diagnostics result of an AutoStopping Rule
 - post
Cool down an AutoStopping Rule
 - get
Return metadata of cool down of an AutoStopping Rule
 - get
Return cumulative savings for all the AutoStopping Rules
 - put
Disable/Enable an Autostopping Rule
- Cloud Cost AutoStopping Rules V2

- post
Create an AutoStopping Rule
 - put
Update an existing AutoStopping Rule
- Cloud Cost AutoStopping Load Balancers
 - get
Return all the load balancers
 - put
Update a load balancer
 - post
Create a load balancer
 - del
Delete load balancers and the associated resources
 - get
Return details of a load balancer
 - get
Return all the AutoStopping Rules in a load balancer
 - get
Return last activity details of a load balancer
- Cloud Cost AutoStopping Fixed Schedules
 - get
Return all the AutoStopping Rule fixed schedules
 - post
Create a fixed schedule for an AutoStopping Rule
 - del
Delete a fixed schedule for AutoStopping Rule.
- Commitment Orchestrator Events APIs
 - post
List event logs
 - post
List Anomalies
 - post
Returns the list of distinct values for all the specified Anomaly fields.
 - post
List Anomalies
 - post
List Anomalies for Perspective
 - put
Report Anomaly feedback
 - get
List all the BI Dashboards for CCM
 - get
Fetch Budget group details
 - put
Update an existing budget group
 - del
Delete a budget group
 - post
Get aggregated amount for given budget groups/budgets
 - get
Get list of budget and budget group summaries
 - get
List all the Budget groups
 - post
Create a Budget Group
 - get
Fetch Budget details
 - put
Update an existing budget
 - post
Clone a budget
 - del
Delete a budget
 - get
Fetch the cost details of a Budget
 - get

List all the Budgets associated with a Perspective

- get
List all the Budgets

- post

Create a Budget

- post

Get CCM K8S Metadata

- post

Returns an overview of the cost

- post

Returns cluster data in a tabular format

- post

Returns cost details in a tabular format

- post

Returns cost details in a time series format

- get

Fetch high level overview details about CCM feature.

- get

Fetch details of a Cost category

- del

Delete a Cost category

- get

Return details of all the Cost categories

- put

Update a Cost category

- post

Create Cost category

- post

Add a new rule Enforcement

- post

Fetch Rule Enforcement count for account

- post

Fetch Rule Enforcement execution details for account

- post

Fetch Rule Enforcements for account

- post

Clone a rule

- del

Delete a rule

- post

Enqueues job for execution

- get

Get Schema for entity

- post

Fetch rules for account

- get

connectors with governance enabled and valid permission

- put

Update a Rule

- post

Validate a rule

- post

Create a Perspective folder

- del

Delete a folder

- get

Fetch folders for an account

- put

Update a folder

- get

Return details of all the Perspectives

- post

Move a Perspective

- get

Fetch details of a cost Report

- put
 - Update a cost Perspective Report
- post
 - Create a schedule for a Report
- del
 - Delete cost Perspective report

- get
 - Fetch details of a Perspective
- put
 - Update a Perspective
- post
 - Create a Perspective
- del
 - Delete a Perspective
- get
 - Return details of all the Perspectives
- get
 - Get the last period cost for a Perspective
- get
 - Get the last twelve month cost for a Perspective

- get
 - Return Azure VM Recommendation
- get
 - Return EC2 Recommendation
- get
 - Return ECS Recommendation
- get
 - Return node pool Recommendation
- get
 - Return workload Recommendation

- post
 - Return void
- post
 - Return the number of Recommendations
- post
 - Return the list of filter values for the Recommendations
- post
 - Return the list of Recommendations
- post
 - Return Recommendations statistics Grouped on Resource Type
- post
 - Return Recommendations statistics

- post
 - Create jira for recommendation

- post
 - Create servicenow ticket for recommendation

- post
 - Add resources to recommendations ignore list
- get
 - Get resources in recommendations ignore list
- post
 - Remove resources from recommendations ignore list

- get
 - List AutoStopping Rules
- post
 - Create an AutoStopping Rule
- get
 - Return AutoStopping Rule details
- del
 - Delete an AutoStopping Rule
- get
 - Return health status of an AutoStopping Rule
- get
 - Return savings details for an AutoStopping Rule
- get
 - List all the resources for an AutoStopping Rule

- get
 - Return diagnostics result of an AutoStopping Rule
- post
 - Cool down an AutoStopping Rule
- get
 - Return metadata of cool down of an AutoStopping Rule
- get
 - Return cumulative savings for all the AutoStopping Rules
- put
 - Disable/Enable an Autostopping Rule

- post
 - Create an AutoStopping Rule
- put
 - Update an existing AutoStopping Rule

- get
 - Return all the load balancers
- put
 - Update a load balancer
- post
 - Create a load balancer
- del
 - Delete load balancers and the associated resources
- get
 - Return details of a load balancer
- get
 - Return all the AutoStopping Rules in a load balancer
- get
 - Return last activity details of a load balancer

- get
 - Return all the AutoStopping Rule fixed schedules
- post
 - Create a fixed schedule for an AutoStopping Rule
- del
 - Delete a fixed schedule for AutoStopping Rule.

- post
 - List event logs

- API Keys
 - get
 - Returns API Keys for an Environment
 - post
 - Creates an API key for the given Environment
 - del
 - Deletes an API Key
 - get
 - Returns API keys
 - put
 - Updates an API Key

- Feature Flags
 - get
 - Returns all Feature Flags for the project
 - post
 - Creates a Feature Flag
 - del
 - Delete a Feature Flag
 - get
 - Returns a Feature Flag
 - patch
 - Updates a Feature Flag
 - get
 - Return a list of dependant flags
 - post
 - Restore a Feature Flag

- Targets
 - get
 - Returns all Targets

- post
Creates a Target
 - post
Add Target details
 - del
Deletes a Target
 - get
Returns details of a Target
 - patch
Updates a Target
 - put
Modifies a Target
 - get
Returns Target Groups for the given Target
- Target Groups
 - get
Returns all Target Groups
 - post
Creates a Target Group
 - del
Deletes a Target Group
 - get
Returns Target Group details for the given identifier
 - patch
Updates a Target Group
 - get
Returns Feature Flags that are available to be added to the given Target Group
 - get
Returns Feature Flags in a Target Group
- Environment Perspectives
 - del
Delete a Perspective - Environment link.
 - put
Upsert a Perspective to an Environment.
- get
Returns API Keys for an Environment
- post
Creates an API key for the given Environment
- del
Deletes an API Key
- get
Returns API keys
- put
Updates an API Key
- get
Returns all Feature Flags for the project
- post
Creates a Feature Flag
- del
Delete a Feature Flag
- get
Returns a Feature Flag
- patch
Updates a Feature Flag
- get
Return a list of dependant flags
- post
Restore a Feature Flag
- get
Returns all Targets
- post
Creates a Target
- post
Add Target details
- del
Deletes a Target
- get
Returns details of a Target

- patch
Updates a Target
- put
Modifies a Target
- get
Returns Target Groups for the given Target
- get
Returns all Target Groups
- post
Creates a Target Group
- del
Deletes a Target Group
- get
Returns Target Group details for the given identifier
- patch
Updates a Target Group
- get
Returns Feature Flags that are available to be added to the given Target Group
- get
Returns Feature Flags in a Target Group
- del
Delete a Perspective - Environment link.
- put
Upsert a Perspective to an Environment.
- Monitored Services
 - post
createDefaultMonitoredService
 - get
Get monitored service data
 - put
Updates monitored service data
 - del
Delete monitored service data
 - put
delete template reference from monitored service
 - get
getAllMonitoredServicesWithHealthSources
 - get
CvgetAnomaliesSummary
 - get
getCountOfServices
 - get
getEnvironments
 - get
getHealthSources
 - get
getHealthSourcesForMonitoredServiceIdentifier
 - get
getList
 - get
getListV2
 - get
getMSSecondaryEvents
 - get
getMSSecondaryEventsDetails
 - get
getMonitoredServiceChangeDetails
 - get
getMonitoredServiceDetails
 - get
getMonitoredServiceDetails_1
 - get
getMonitoredServiceFromServiceAndEnvironment
 - get
getMonitoredServiceLogs
 - get
fetch reconciliation status for template referenced monitored services
 - get
get monitored service resolved template inputs

- get
getMonitoredServiceScore
 - get
Get notification rules for MonitoredService
 - get
getOverAllHealthScore
 - get
getServices
 - get
getSloMetrics
 - get
check if a template referenced monitored service(s) require reconciliation
 - get
list
 - post
Saves monitored service data
 - post
Saves monitored service from template input
 - post
saveMonitoredServiceFromYaml
 - put
setHealthMonitoringFlag
 - put
Update monitored service from yaml or template
 - put
updateMonitoredServiceFromYaml
 - get
yamlTemplate
- SLOs dashboard
 - get
getSLOAssociatedEnvironmentIdentifiers
 - get
getSLOAssociatedMonitoredServices
 - get
getSecondaryEventDetails
 - get
getSecondaryEvents
 - get
Get all SLOs count by risk
 - get
Get SLO consumption breakdown
 - get
Get SLO dashboard details
 - get
Get SLO list view
 - post
Get SLO list view
- NG SLOs
 - get
Get SLO data
 - put
Update SLO data
 - del
Delete SLO data
 - post
Get onBoarding graph for composite slo
 - post
Get SLO list view
 - get
Get all SLOs
 - post
Saves SLO data
- SLOs
 - get
Get Error budget reset history
 - get
Get notification rules for SLO
 - get
Get SLO logs

- post
Reset Error budget history
 - get
Get Metric Graph For SLO
 - get
List SLOs
- Downtime
 - get
getDowntime
 - put
updateDowntimeData
 - del
deleteDowntimeData
 - get
getAssociatedMonitoredServices
 - get
getDowntimeAssociatedMonitoredServices
 - get
getHistory
 - get
listDowntimes
 - post
saveDowntime
 - put
updateDowntimeEnabled
- Srm Notification
 - get
getNotificationRuleData
 - put
updateNotificationRuleData
 - del
deleteNotificationRuleData
 - get
getNotificationRuleData_1
 - post
saveNotificationRuleData
- post
createDefaultMonitoredService
- get
Get monitored service data
- put
Updates monitored service data
- del
Delete monitored service data
- put
delete template reference from monitored service
- get
getAllMonitoredServicesWithHealthSources
- get
CvgetAnomaliesSummary
- get
getCountOfServices
- get
getEnvironments
- get
getHealthSources
- get
getHealthSourcesForMonitoredServiceIdentifier
- get
getList
- get
getListV2
- get
getMSSecondaryEvents
- get
getMSSecondaryEventsDetails
- get
getMonitoredServiceChangeDetails
- get

- getMonitoredServiceDetails
- get
getMonitoredServiceDetails_1
- get
getMonitoredServiceFromServiceAndEnvironment
- get
getMonitoredServiceLogs
- get
fetch reconciliation status for template referenced monitored services
- get
get monitored service resolved template inputs
- get
getMonitoredServiceScore
- get
Get notification rules for MonitoredService
- get
getOverAllHealthScore
- get
getServices
- get
getSloMetrics
- get
check if a template referenced monitored service(s) require reconciliation
- get
list
- post
Saves monitored service data
- post
Saves monitored service from template input
- post
saveMonitoredServiceFromYaml
- put
setHealthMonitoringFlag
- put
Update monitored service from yaml or template
- put
updateMonitoredServiceFromYaml
- get
yamlTemplate
- get
getSLOAssociatedEnvironmentIdentifiers
- get
getSLOAssociatedMonitoredServices
- get
getSecondaryEventDetails
- get
getSecondaryEvents
- get
Get all SLOs count by risk
- get
Get SLO consumption breakdown
- get
Get SLO dashboard details
- get
Get SLO list view
- post
Get SLO list view
- get
Get SLO data
- put
Update SLO data
- del
Delete SLO data
- post
Get onBoarding graph for composite slo
- post
Get SLO list view
- get
Get all SLOs
- post

Saves SLO data

- get
Get Error budget reset history
- get
Get notification rules for SLO
- get
Get SLO logs
- post
Reset Error budget history
- get
Get Metric Graph For SLO
- get
List SLOs
- get
getDowntime
- put
updateDowntimeData
- del
deleteDowntimeData
- get
getAssociatedMonitoredServices
- get
getDowntimeAssociatedMonitoredServices
- get
getHistory
- get
listDowntimes
- post
saveDowntime
- put
updateDowntimeEnabled
- get
getNotificationRuleData
- put
updateNotificationRuleData
- del
deleteNotificationRuleData
- get
getNotificationRuleData_1
- post
saveNotificationRuleData
- Backstage Allow List
 - get
Get backend url allow list
 - post
Save backend url allow list
- Backstage App Configurations
 - post
Toggle Plugin
 - post
Save Or Update Plugin Config
- Backstage Auth Information
 - post
Save Auth Info
- Backstage Environment Variable
 - put
Reload backstage env variables
- Connector Information
 - get
Get Connector Info
 - post
Create or Update Connector Info
 - get
Get Connector Info by Provider Type

- DataSource Information
 - get
Get Datasources Present In Account
 - get
Get DataPoints present in DataSources for an account
 - get
Get Data Sources and Data Points Map for Account
- Kubernetes DataPoints Information
 - post
Get data points data for kubernetes data source
- Layout Proxy
 - post
Ingest plugin layout
- Plugin Information
 - get
List Available Plugins
 - post
Save custom plugin info
 - get
Get Plugin
 - put
Update custom plugin info
 - post
Request for a Plugin
 - get
Get all plugin requests for an account
- Scores Information
 - get
Get Score Summary for Scorecards
 - get
Get Scores for Scorecards
 - post
Get Aggregated Scores for backstage entities
 - post
Trigger recalibration of scores for a scorecard
- get
Get backend url allow list
- post
Save backend url allow list
- post
Toggle Plugin
- post
Save Or Update Plugin Config
- post
Save Auth Info
- put
Reload backstage env variables
- get
Get Connector Info
- post
Create or Update Connector Info
- get
Get Connector Info by Provider Type
- get
Get Datasources Present In Account
- get
Get DataPoints present in DataSources for an account
- get
Get Data Sources and Data Points Map for Account
- post
Get data points data for kubernetes data source

- post
Ingest plugin layout
- get
List Available Plugins
- post
Save custom plugin info
- get
Get Plugin
- put
Update custom plugin info
- post
Request for a Plugin
- get
Get all plugin requests for an account
- get
Get Score Summary for Scorecards
- get
Get Scores for Scorecards
- post
Get Aggregated Scores for backstage entities
- post
Trigger recalibration of scores for a scorecard
- - Custom Dashboards
 - get
Download data within a Dashboard
- get
Download data within a Dashboard
- - dashboard
 - get
dashboard#metrics
- - examples
 - get
examples#list
- - policies
 - get
policies#list
 - post
policies#create
 - del
policies#delete
 - get
policies#find
 - patch
policies#update
- - evaluate
 - post
evaluate#evaluate
- - evaluations
 - get
evaluations#list
 - get
evaluations#find
- - policysets
 - get
policysets#list
 - post
policysets#create
 - del
policysets#delete
 - get
policysets#find

- patch
policysets#update
 - system
 - get
system#health
 - get
system#version
 - get
dashboard#metrics
 - get
examples#list
 - get
policies#list
 - post
policies#create
 - del
policies#delete
 - get
policies#find
 - patch
policies#update
 - post
evaluate#evaluate
 - get
evaluations#list
 - get
evaluations#find
 - get
policysets#list
 - post
policysets#create
 - del
policysets#delete
 - get
policysets#find
 - patch
policysets#update
 - get
system#health
 - get
system#version
-
- Git Branches
 - post
Sync the content of new Git Branch into harness with Git Sync Config Id
 - get
Lists branches with their status(Synced, Unsynced) by Git Sync Config Id for the given scope
 - Git Full Sync
 - get
Fetch Configuration for Git Full Sync for the provided scope
 - put
Update Configuration for Git Full Sync for the provided scope
 - post
Create Configuration for Git Full Sync for the provided scope
 - post
List files in full sync along with their status
 - post
Trigger Full Sync
 - Git Sync Settings
 - get
Get Git Sync Setting for the given scope
 - put

This updates the existing Git Sync settings within the scope. Only changing Connectivity Mode is allowed

- post
Creates Git Sync Setting in a scope

- Git Sync

- get
Lists Git Sync Config for the given scope
- put
Update existing Git Sync Config by Identifier
- post
Creates Git Sync Config in given scope
- get
Check whether Git Sync is enabled for given scope or not
- put
Update existing Git Sync Config default root folder by Identifier

- Git Sync Errors

- get
Get Errors Count for the given scope, Repo and Branch
- get
Lists Git to Harness Errors by file or connectivity errors for the given scope, Repo and Branch
- get
Lists Git to Harness Errors for the given Commit Id
- get
Lists Git to Harness Errors grouped by Commits for the given scope, Repo and Branch

- post
Sync the content of new Git Branch into harness with Git Sync Config Id

- get
Lists branches with their status(Synced, Unsynced) by Git Sync Config Id for the given scope

- get
Fetch Configuration for Git Full Sync for the provided scope

- put
Update Configuration for Git Full Sync for the provided scope

- post
Create Configuration for Git Full Sync for the provided scope

- post
List files in full sync along with their status

- post
Trigger Full Sync

- get
Get Git Sync Setting for the given scope

- put
This updates the existing Git Sync settings within the scope. Only changing Connectivity Mode is allowed

- post
Creates Git Sync Setting in a scope

- get
Lists Git Sync Config for the given scope

- put
Update existing Git Sync Config by Identifier

- post
Creates Git Sync Config in given scope

- get
Check whether Git Sync is enabled for given scope or not

- put
Update existing Git Sync Config default root folder by Identifier

- get
Get Errors Count for the given scope, Repo and Branch

- get
Lists Git to Harness Errors by file or connectivity errors for the given scope, Repo and Branch

- get
Lists Git to Harness Errors for the given Commit Id

- get
Lists Git to Harness Errors grouped by Commits for the given scope, Repo and Branch

- Error Response [Beta]

- Governance Metadata [Beta]

Harness NextGen Software Delivery Platform API Reference

(1.0)

Download OpenAPI specification:[Download](#)

The Harness Software Delivery Platform uses OpenAPI Specification v3.0. Harness constantly improves these APIs. Please be aware that some improvements could cause breaking changes.

Introduction

The Harness API allows you to integrate and use all the services and modules we provide on the Harness Platform. If you use client-side SDKs, Harness functionality can be integrated with your client-side automation, helping you reduce manual efforts and deploy code faster.

For more information about how Harness works, read our documentation or visit the Harness Developer Hub.

How it works

The Harness API is a RESTful API that uses standard HTTP verbs. You can send requests in JSON, YAML, or form-data format. The format of the response matches the format of your request. You must send a single request at a time and ensure that you include your authentication key. For more information about this, go to [Authentication](#).

Get started

Before you start integrating, get to know our API better by reading the following topics:

- Harness key concepts
- Authentication
- Requests and responses
- Common Parameters
- Status Codes
- Errors
- Versioning
- Pagination

The methods you need to integrate with depend on the functionality you want to use. Work with your Harness Solutions Engineer to determine which methods you need.

Authentication

To authenticate with the Harness API, you need to:

Generate an API token

To generate an API token, complete the following steps:

Important: Make sure to save your token securely. Harness does not store the API token for future reference, so make sure to save your token securely before you leave the page.

Send the API token in your requests

Send the token you created in the Harness Platform in the x-api-key header. For example: `x-api-key: YOUR_API_KEY_HERE`

Requests and Responses

The structure for each request and response is outlined in the API documentation. We have examples in JSON and YAML for every request and response. You can use our online editor to test the examples.

Common Parameters [Beta]

Field Name	Type	Default	Description
identifier	string	none	URL-friendly version of the name, used to identify a resource within its scope and so needs to be unique within the scope.
name	string	none	Human-friendly name for the resource.

Field Name	Type	Default	Description
org	string	none	Limit to provided org identifiers.
project	string	none	Limit to provided project identifiers.
description	string	none	More information about the specific resource.
tags	map[string]string	none	List of labels applied to the resource.
order	string	desc	Order to use when sorting the specified fields. Type: enum(asc,desc).
sort	string	none	Fields on which to sort. Note: Specify the fields that you want to use for sorting. When doing so, consider the operational overhead of sorting fields.
limit	int	30	Pagination: Number of items to return.
page	int	1	Pagination page number strategy: Specify the page number within the paginated collection related to the number of items in each page.
created	int64	none	Unix timestamp that shows when the resource was created (in milliseconds).
updated	int64	none	Unix timestamp that shows when the resource was last edited (in milliseconds).

Status Codes

Harness uses conventional HTTP status codes to indicate the status of an API request. Generally, 2xx responses are reserved for success and 4xx status codes are reserved for failures. A 5xx response code indicates an error on the Harness server.

Error Code	Description
200	OK
201	Created
202	Accepted
204	No Content
400	Bad Request
401	Unauthorized
403	Forbidden
412	Precondition Failed
415	Unsupported Media Type
500	Server Error

To view our error response structures, go [here](#).

Versioning [Beta]

Harness Version

The current version of our Beta APIs is yet to be announced. The version number will use the date-header format and will be valid only for our Beta APIs.

Generation

All our beta APIs are versioned as a Generation, and this version is included in the path to every API resource. For example, v1 beta APIs begin with `app.harness.io/v1/`, where v1 is the API Generation.

The version number represents the core API and does not change frequently. The version number changes only if there is a significant departure from the basic underpinnings of the existing API. For example, when Harness performs a system-wide refactoring of core concepts or resources.

Pagination [Beta]

We use pagination to place limits on the number of responses associated with list endpoints. Pagination is achieved by the use of limit query parameters. The limit defaults to 30. Its maximum value is 100.

Following are the pagination headers supported in the response bodies of paginated APIs:

For example:

Source URL: <https://www.harness.io/products/continuous-integration/migrating-to-ci>

Slash your CI bill up to 30%

Get your builds in production in just 6 weeks

Calculate your savings *

With CircleCI

Annual cost

\$325,929

With Harness

Annual cost

\$228,150

Build time saved

45,630

Select your current cloud CI provider

How many builds do you run weekly?

What is your average build time in minutes?

Select build machine type

+ Add an additional machine type

* The savings calculations above are for informational purposes only. Actual savings may vary due to a number of factors, including but not limited to applicable discounts, promotions, contract terms, market conditions. Competitor pricing is based on publicly available data at the time of publishing.

How much can you save from switching to Harness? *

Build up to 50% faster

Our optimized cloud build machines, intelligent caching, and exclusive Test Intelligenceâ¢ ensure builds run up to 4x faster than other CI solutions.

3 months of free credits

Get up to 3 months of complimentary credits valued at up to \$30,000.

Migrate to prod in just 6 weeks

Our dedicated team will partner with you Â to get your builds in production in just 6 weeks. Your success is our mission.

Additional terms apply. See terms and conditions.

Rev up your development with Harness Cl:Drive velocity while slashing your CI bill by up to 30%

Why make the switch?

Build up to 50% faster

Our optimized cloud build machines, intelligent caching, and exclusive Test Intelligenceâ¢ ensure builds run up to 4x faster than other CI solutions.

Save up to 30% on your CI bill

Our competitive cloud pricing, combined with reduced build times, offers unparalleled value.

3 months of free credits

Get 3 months of complimentary credits valued at \$30,000 to apply towards complementary onboarding.

Complimentary onboarding package

Our dedicated team will partner with youÂ to get your builds in production in just 6 weeks. Your success is our mission.

Additional terms apply. See terms and conditions.

* The savings calculations above are for informational purposes only. Actual savings may vary due to a number of factors, including but not limited to applicable discounts, promotions, contract terms, market conditions. Competitor pricing is based on publicly available data at the time of publishing.

How much can you save from switching to Harness? *

Our Migration Plan

Risk-free evaluation

Risk-free evaluation

Make your decision based on tangible results, without any binding commitments.

Week 1-2

Scoping and proof of value (POV)

- Evaluate our solution with no commitment from your side Â
- Get a free SDLC diagnostic of your current state

Complimentary SDLC diagnostics

Benefit from a thorough analysis of your current SDLC, provided with our premium white-glove service at no cost.

Week 3

Decision time

Swift transition

Transition from your existing vendor to us in just 6 weeks, ensuring minimal disruption to your operations.

Week 4-6

Complete migration

Our dedicated team ensures you're up and running in production without a hitch.

Ready to Make the Switch?

Please fill out the form and an expert will reach out to you shortly.

Ready to Make the Switch?

Please fill out the form and an expert will reach out to you shortly.

Thank you for submitting your information, an expert will reach out to you shortly.

Terms and conditions

- The Harness CI SaaS Migration Offer (âOfferâ) is valid until 04/30/2024 (âExpiration Dateâ), and will expire thereafter.
- This Offer is exclusively available to customers that purchase Harness Hosted CI for the first time, prior to the end of the Expiration Date.
- To apply, please fill out the form via the landing page.
- The offer includes 3 free months of complimentary Harness cloud credits, up to a maximum of 6 million free credits, to be used over the first year of contract. Unused credits do not rollover.

Save Costs by Running Only Tests That Matter

AI-powered Test Intelligence

Why run all tests all the time?

Use AI-powered Test Intelligence to run only Unit Tests related to your code change, slashing test cycle time by up to 80%. Welcome to smarter testing.

Cut testing costs

Cut Build Cost

By skipping irrelevant tests, builds are faster, streamlining development and reducing cloud and infrastructure expenses. Allocate your testing budget effectively.

Cut Troubleshooting Cost

Prioritizing relevant tests reduces unrelated test failures, preventing developers from being derailed by irrelevant issues and boosting productivity.

Cut Context-switch Cost

With less time spent waiting for tests, developers can maintain their focus on coding tasks, increasing productivity.

Ship faster without compromising quality

Cut Tests Cycle Time

- Run only tests that matter with Contextual test selection.
- Optimize with parallel testing

Faster issues resolution

- Avoid irrelevant test failures. Trim tests to hit fewer snags and focus on what truly matters.
- View test results for faster discovery of failures and triaging issues.
- Use AI-based remediation suggestions to accelerate issue resolution.

Understand selected tests

Get full visibility into which tests were selected and why, to identify negative trends and gain insights to improve test quality and coverage.

Test suite insights

Use our dashboard to turn testing metrics into actionable insights. Refine your test cycle for top performance.

Understand code quality with Test Insights

- **Failure Rate Analysis:** Target tests with frequent failures for proactive troubleshooting and resolution.
- **Test Duration Insights:** Spot and assess your lengthiest tests, streamlining them for faster execution and better resource management.
- **Historical Trends:** Monitor test performance and reliability across different timeframes, unveiling patterns and spotlighting areas for improvement.

Flexible Dashboards

Personalize and craft dashboards that cater to your unique requirements.

Harness Builds

Product Documentation

Learn how to connect SEI with your existing tech stack and get insights. How to remove bottlenecks and improve planning and sprint hygiene

Product Updates

See our latest feature releases, product improvements and announcements

White Papers

See our latest feature releases, product improvements and announcements

Case Studies

Sign up for a free 14 day trial and take your software development to the next level

Source URL: <https://www.harness.io/products/continuous-delivery/cd-visualize-devops-data>

Visualize DevOps Data

Harness dashboards provide visibility into your continuous delivery performance. Harness CD meticulously tracks applications across environments. Use Harness deployment dashboards to understand what is where as well as the performance of your continuous delivery efforts.

Unified Deployment Dashboards

Unified Deployment Dashboards give a consolidated view of deployments

Counts

Understand how many successful and failed deployments have happened for a project.

Trending

Harness visualizes deployment frequency and indicates how much counts are increasing or decreasing.

Failure Rate

Identify which services experience deployment failures most often to prioritize improvements and understand risk.

Execution Frequency

Track deployment frequency on a daily basis.

Service Dashboard

Drill down into a service's dashboard to see what version is deployed in each environment when it was last updated in those environments and a history of recent deployments of the service.

Environment Dashboard

Test and production environments typically contain many services. Drill down into an environment dashboard to see what services are deployed and when they were last updated

Custom Dashboards

Tailor dashboards to your organization's needs leveraging Looker technology. Custom dashboards define tiles with their own queries and visualizations. Slice and dice data across teams or projects to deliver the understanding that you need.

Send reports when it matters

Send dashboards to interested parties on a schedule or based on a conditional alert. If a report is supporting a status meeting or retrospective, schedule it to be sent to the team before the meeting so they don't have to come looking for it. For those reports that are most interesting when things are starting to go wrong, eliminate the noise by setting an alert to fire out a report only when conditions are met.

DORA Dashboards

Harness CD provides DORA metrics out of the box.

DORA Dashboards provides the following:

- **Deployments Frequency:** Frequency of deployments to production. Shipping more often indicates smaller batch sizes which reduces risk and overhead.
- **Lead time to Production:** Average time for a code commit to reach production. Faster indicates more automated testing and less friction.
- **Mean time to restore(MTTR):** Time to revert a deployment. Faster reversions reduce the impact of a problem, reducing risk and enabling more frequent delivery.
- **Change Failure Rate:** Frequency with which a deployment failed and needed to be reverted. Lower failure is better.

Ride the wave of Modern Software Delivery

Have a question? We are here to help!

Source URL: <https://www.harness.io/products/software-engineering-insights/features>

Engineering Excellence Platform For The Enterprise

Data-led Insights to Remove DevOps Bottlenecks, Automate Workflows,
Improve Dev Productivity

Collections allow users to define specific grouping entities such as Teams, Projects, or Sprints.

Out-of-the-box KPIs to unlock metrics such as DORA, Code-Commit and others to track aspects of engineering excellence

Profiles are a way to visualize and understand the performance of your engineering teams. They provide a comprehensive view of your team's strengths and weaknesses, and can help you identify areas for improvement.

Actionable Insights to analyze across different collections to pinpoint bottlenecks quickly

Low code automation engine that allows to
drive continuous improvement and enforce
good team habits.

Custom Metrics that allows to author custom
Javascript code and calculate metrics leveraging metadata collected by SEI.

Get insights from 40+ tools

Do More. Ship More. Celebrate More

[Learn more about](#)

Harness Software Engineering Insights

Product Documentation

Learn how to connect SEI with your existing tech stack and get insights. How to remove bottlenecks and improve planning and sprint hygiene

Product Updates

See our latest feature releases, product improvements and announcements

Blogs

See our latest feature releases, product improvements and announcements

Source URL: <https://www.harness.io/public-sector/find-a-federal-reseller>

Find a Federal Partner

Want to learn more?

Reach out to us at publicsector@harness.io

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/blog/cicd-for-serverless>

CI/CD for Serverless

Ah àserverlessâ... the promise of running applications without worrying about servers! The term "serverless" often sparks skepticism among those rooted in traditional software development practices. The idea of handing over control over infrastructure may seem daunting, and developers might wonder if serverless is even right for their use case. My goal in this blog is to demystify serverless computing (in particular, serverless deployment) without unnecessary complexity.

Embracing Abstraction - From Infrastructure to Serverless

Evolution to SaaS and the Kubernetes Leap

The software development landscape has witnessed a fundamental transformation through the lens of abstraction. The rise of Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) marked a pivotal shift, freeing engineering teams from the nitty-gritty of infrastructure concerns with the advent of Software as a Service (SaaS).

The massive adoption and growth of Kubernetes, a powerful container orchestrator, redefined how the underlying infrastructure can be abstracted away from engineers who build and deploy containerized applications. In a concise analogy, Kelsey Hightower tweeted: "**Kubernetes is not the kernel; it's systemd.**" This emphasizes its role as a higher-level orchestrator, adept at handling the deployment, scaling, and management of containerized applications.

Serverless and FaaS: The Ultimate Abstraction

As we delve into Serverless computing, it represents the apex of abstraction, pushing the boundaries even further. Beyond Software as a Service (SaaS), Serverless encapsulates Functions as a Service (FaaS), providing a powerful tool for engineers to zero in on their code. It simplifies by abstracting away server management complexities, embracing a dynamic, event-driven model for resource efficiency.

In the realm of FaaS, a function acts as "glue code," seamlessly connecting and extending existing services. This approach utilizes a runtime framework where you deploy a compact snippet of code, not an entire container. Within this snippet, you implement a single function or override a method, specifically designed to handle incoming requests or events. Importantly, there's no need to start an HTTP Server manually.

The beauty of FaaS lies in its serverless nature, offering a straightforward developer experience where concerns about the runtime of your code fade away. Scaling is inherently built in, ensuring a seamless and hassle-free experience.

It's crucial to note that Serverless extends beyond the handling of HTTP requests. You can think of FaaS as a subset of Serverless but Serverless is not FaaS.

Benefits of Serverless Computing

Using Serverless Computing comes with some great advantages that make a difference on cost, scalability, overhead, and more in how software is delivered.

Cost Efficiency

With Serverless, you only pay for what you use, like paying for the energy you use at home. No big upfront costs for infrastructure à it's a flexible and budget-friendly way to handle computing resources.

Scalability and Flexibility

Serverless adjusts automatically to handle more or fewer tasks. When your app gets busy, it scales up. When things slow down, it scales down. This flexibility keeps your app running smoothly without wasting resources.

Simplified Development and Deployment

For developers, Serverless means less hassle with servers. You can concentrate on writing code, and when it's time to put it into action, Serverless takes care of the technical details. The burden of setting up compute infrastructure is abstracted away, letting you focus on your code without dealing with the nitty-gritty of server management.

Use Cases for Serverless Computing

Serverless computing is a flexible approach that can be applied to many different areas. Let's explore key use cases where serverless computing proves to be a valuable solution.

Auto-scaling Websites and APIs

Serverless computing is tailor-made for dynamically changing workloads. For websites and APIs with fluctuating traffic, serverless allows automatic scaling up or down based on demand. This ensures optimal performance without the need for manual intervention, making it an ideal solution for handling varying user loads.

Event Streaming

In event-driven architectures, serverless excels at processing and responding to events in real-time. By leveraging serverless capabilities, organizations can effortlessly manage event streaming, reacting promptly to changes and ensuring seamless communication between services.

Processing Events and SaaS

Serverless is a natural fit for processing events, especially in scenarios where events trigger specific actions. Integrating with Software as a Service (SaaS) applications becomes more streamlined, allowing organizations to automate tasks, synchronize data, and enhance overall workflow efficiency.

Hybrid Cloud Applications

Serverless computing seamlessly integrates with hybrid cloud models, where applications span both on-premises and cloud environments. This flexibility ensures that organizations can deploy and manage applications without being constrained by the limitations of a single hosting environment.

Internet of Things (IoT)

The scalability and event-driven nature of serverless make it an excellent choice for IoT applications. Handling data from numerous devices and responding to events in real-time is simplified, allowing for efficient processing and analysis in IoT ecosystems.

Continuous Integration and Continuous Deployment (CI/CD)

In the fast-paced landscape of software development, the ability to iterate rapidly is more crucial than ever. CI/CD pipelines play a pivotal role in enabling organizations to ship code in small, incremental updates, ensuring that bug fixes and other enhancements can be seamlessly delivered on a daily basis.

Serverless computing amplifies the power of CI/CD by automating key processes in the software development lifecycle. Code check-ins can trigger the automatic building and redeployment of websites, while pull requests (PRs) can initiate the execution of automated tests, ensuring thorough code testing before human review. This automation not only accelerates the delivery pipeline but also enhances the reliability and quality of the software being deployed.

When exploring the automation possibilities within Serverless Applications, the potential to eliminate manual tasks from the workflow becomes apparent. From triggering builds and tests to orchestrating seamless deployments, serverless computing empowers development teams to focus on coding and innovation rather than getting bogged down by manual, time-consuming tasks. This shift toward automation not only increases efficiency but also reduces the risk of human errors, ultimately contributing to a more agile and responsive development process.

The dark sides of Serverless

While Serverless computing offers a multitude of benefits and has a number of use cases, it's crucial to acknowledge that it's not a one-size-fits-all solution. Like any technology, Serverless has its own set of challenges.Â

Unpredictable Costs

With rapid autoscaling comes unpredictable billing. Just as you consider factors like performance, security, and scalability, now, costs are an essential aspect of your code in serverless architecture. It's crucial for you as a developer to be aware of these costs and, importantly, to have control over them.

Portability Problems

Serverless platforms may not offer the portability freedom you imagine. While your code is portable, the triggers and scaling mechanisms are often specific to each cloud provider. For instance, the supported versions of languages may vary, impacting the migration of code across different platforms. Solutions like Kubernetes-based offerings provide some relief but come with their own set of choices and opinions.

Smells Like Infrastructure

Despite the promise of abstraction, many Serverless implementations still involve choices in infrastructure. Options like Apache OpenWhisk, Open FaaS, or KNative offer more generic serverless infrastructure but require maintenance. The rapid and sometimes unpredictable scaling of functions demands careful cluster capacity management, introducing complexities of its own.

Shifting Complexity

Serverless shifts complexity, especially in short-lived and rapidly scaling functions. While it simplifies certain aspects, the management of state becomes critical as functions may need to interact with external systems to maintain continuity. However, these challenges don't have to be insurmountable. Leveraging tools like the Serverless Framework can provide a solution by abstracting away some of the complexities.¹

Your Function Has a Function

With the ease of invoking serverless functions, there's a risk of falling into the Big Ball of Mud antipattern. The rapid instantiation of "serverless containers" can lead to the inclusion of excessive logic in a single function. Advanced tools like Amazon Step Functions make orchestrating multiple functions easier but require careful design to avoid complexity pitfalls.

Cold Starts

Unlike traditional systems, Serverless introduces the concept of cold starts, where every request triggers the instantiation of a new serverless container. This is particularly critical in languages like JAVA, demanding thoughtful design and infrastructure considerations.

To mitigate this challenge, hyperscalers like AWS have introduced solutions to enhance the performance of serverless functions. For instance, AWS has introduced SnapStart for AWS Lambda functions running on Java Corretto 11. This feature significantly improves function startup performance, providing up to 10x faster initialization, without incurring additional costs.

Why CI/CD for Serverless Deployment

The principles that govern traditional production controls also apply to serverless deployment. While modifying or patching a Lambda is remarkably straightforward, similar to a traditional system, it's essential to maintain disciplined practices. Hot patching in production, even if it's relatively easy for serverless functions, raises concerns about maintaining control and adhering to Software Development Life Cycle (SDLC) principles.

Enterprises employ production controls to prevent unauthorized modifications and ensure a structured approach to changes. This discipline is equally applicable to serverless functions, and while it's tempting to make in-line modifications using tools like the AWS Lambda editor, it's crucial to integrate serverless deployment into a robust CI/CD pipeline.

A well-orchestrated CI/CD pipeline that incorporates serverless functions offers several advantages. Firstly, it extends the same level of diligence and confidence established by the SDLC and Continuous Delivery processes to serverless workloads. This ensures that modifications to serverless functions follow a systematic approach, aligning with established organizational standards.

Moreover, integrating serverless deployment into your CI/CD pipeline enhances early defect detection, increases productivity, and promotes faster release cycles. Advanced deployment strategies become feasible within a comprehensive CI/CD pipeline, including canary releases, blue-green deployments, and feature toggles, allowing for controlled testing and gradual rollouts.

By extending the CI/CD pipeline to encompass both serverless and non-serverless workloads, organizations can achieve consistency and efficiency across their entire application landscape. Utilizing a CI/CD increases early defect detection, increases productivity, and enables faster release cycles, reflecting the agility and efficiency that serverless computing promises within a disciplined development and deployment framework. This approach not only streamlines the management of serverless functions but also aligns with best practices in software delivery, promoting reliability and maintainability.

Serverless Deployment with Harness

Now that we've explored the fundamentals and nuances of serverless deployment, it's time to put theory into practice. Harness Continuous Delivery (CD) empowers you to seamlessly integrate serverless deployment into your CI/CD pipeline, ensuring a disciplined and controlled approach to managing your serverless functions.

To begin your hands-on journey and unlock the potential of Harness CD for serverless deployment, delve into our tutorials:

- Deploy on AWS Lambda using Harness Continuous Delivery (CD)
- Deploy Google Cloud Functions to Google Cloud using a Harness CD pipeline

These comprehensive guides will walk you through the process of deploying sample functions seamlessly using Harness pipelines.

â

Similar Blogs

ArgoCD, Terraform and Harness

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Hygiene in SDLC: A Key to Engineering Efficiency

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/harness-product-modules/infrastructure-as-code-management>

Source URL: <https://www.harness.io/harness-product-modules/continuous-delivery>

Source URL: <https://www.harness.io/event/devsecops-events-with-harness>

DevSecOps Events with Harness

Sid Choudhury

VP, Product

Nick Durkin

Field CTO

Sean Roth

Director, Solutions

Marketing, DevSecOps

Building a meaningful DevSecOps practice is likely top of mind for your organization. But given the current threat landscape, you're probably struggling to identify the right security strategies to adopt that won't impact developer productivity. To make things more interesting, even when you have the right strategies, it'll take time to execute them by implementing new processes, upgrading your technology stack, and ultimately changing the culture. Sounds like quite an ordeal, right? Well, we'd like to help you expedite the learning curve.

Join Harness in a series of events around DevSecOps. We'll spend each event networking and diving into important DevSecOps discussion topics such as:

- DevSecOps program best practices
- Governance of open source artifacts in the pipeline
- Ensuring software integrity
- Zero-day vulnerability remediation
- The use of generative AI in DevSecOps processes

Space is very limited, request your seat today.

Upcoming events

Register today

Source URL: <https://www.harness.io/events/harnessdevdays-feb>

Harness CI/CD DevDays February

Creating experiences to efficiently develop and deploy code is what DevOps and Platform Engineering teams do. The key is to improve the experience internal teams have and how fast and how efficient they iterate. Which is why Harness is providing a free hands-on workshop to teach you how to go from code to deployed in multiple environments using a modern GitOps approach.

In this three hour workshop, we'll teach you how to build a pipeline that promotes artifacts and manifests to multiple environments with the ability to introduce approvals. You'll also learn how to:

- Build and publish docker artifacts from source
- Scan artifacts for common CVEs
- Facilitate deploying to multiple Kubernetes environments with a GitOps approach
- And so much more!

Harness will also provide a browser based CloudShell Environment with a Kubernetes Cluster and Ubuntu compute instance for use during the duration of the workshop.Â

Pre-Req's

We'll be interacting with a few systems during the class so please have access to the following before the workshop:

- A GitHub account that you have the ability to write to a repository to.Â
- A DockerHub account that you have the ability to push to.Â
- A Harness account that you are an admin for. Sign up here for a new account.Â

Some helpful material to be familiar with ahead of time:

- What is GitOps
- CD and the Modern Software Shop

Nick Lotz

Developer Advocate

Dewan Ahmed

Principal Developer Advocate at Harness

Source URL: <https://www.harness.io/harness-product-modules/continuous-integration>

Source URL: <https://www.harness.io/harness-product-modules/software-engineering-insights>

Source URL: <https://www.harness.io/harness-product-modules/platform>

Source URL: <https://www.harness.io/blog/harness-acquires-armory-assets>

Harness Acquires Armory Assets

We're excited to announce that Harness has acquired key intellectual property and technology from Armory. As a modern software delivery pioneer, Harness provides customers with a robust, scalable, and intelligent platform that supports the development, deployment, and operation of their applications. Armory customers will experience a seamless transition as key Armory engineering and customer support roles are also joining Harness to continue supporting existing implementations.Â

Harness and Armory have led the revolution towards modern continuous deployment. The acquisition will help Harness accelerate its innovation, enhancing the experience of the developer and DevOps teams served by Harness and Armory.

Why this acquisition?

The Armory team has a strong track record of innovation and has built exciting continuous deployment technologies. The opportunity to acquire this technology and recruit people who built it excites us tremendously. At Harness, we are building the best software delivery platform, supporting numerous cloud-native and traditional workloads. This acquisition will accelerate that work.

Harness has found great success in building and acquiring technology that embraces open-source projects, including Drone and Chaos Native. We continue to contribute heavily to a range of projects, including Drone, Litmus Chaos, OpenTofu, and Gitness. Bolstered by notable Spinnaker contributors from Armory, we expect to continue to lead and support the Spinnaker community.Â

âWe are incredibly excited about this opportunity as this will further enable Harness to deliver a best-in-class modern software delivery platform to the developer community. We view this acquisition as a testament to our commitment to the developer community and to simplifying and improving the developer experience,â said Harness CEO, Jyoti Bansal.

What will change (and what wonât)

Armory Continuous Delivery Self-Hosted will continue to be fully supported. Support will move over to Harnessâs systems, which will improve the overall experience.

âHarness has a great reputation for providing world-class support to its customers, and I am confident in their ability to continue to support us during this transition,â said Wei Li, Member of Technical Staff, Pure Storage. âI look forward to developing a potential partnership with the Harness team.â

Moving onto the Harness Platform

As we integrate Armory technology into Harness, Armory customers will have the opportunity to move their deployments onto the Harness Platform. We will work to make this transition as easy and automated as possible. Once on the Harness platform, teams have increased flexibility with a greater breadth of integrations.

Welcome to Harness

We have already had the opportunity to speak to many Armory customers and would like to express our gratitude for the trust you are putting in us. We are excited about what the future holds for modern continuous delivery and are thrilled to share the journey with you.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

5 Reasons to Attend Chaos Carnival

Harness is excited to host the fourth annual Chaos Carnival, a FREE two-day virtual conference on January 24-25, 2024. So, have you registered yet? Here are five reasons why you should register today!

â

- The urgency of adopting chaos engineering practices to give the best developer experience and customer satisfaction
- Case studies of leveraging chaos engineering to launch reliable products and services
- How shift-left reliability can be learned from shift-left security practices
- Implementing chaos engineering throughout the developer lifecycle

â

â

- Day 1 Keynote by Adrian Cockcroft telling about his journey from Netflix to AWS
- Learn how Harness SREs use chaos engineering to ensure customer SLAs are met

â

- SRE enablement and best practices
- Observability
- Shift-left chaos testing
- Cloud-native chaos engineering
- Resilience vs. chaos engineering

You donât want to miss it.

â

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/harness-product-modules/chaos-engineering>

Source URL: <https://www.harness.io/blog/ephemeral-ci-environments-gitness>

Ephemeral CI environments using ttl.sh and Gitness

In the realm of software development, balancing continuous integration with maintaining quality is a significant challenge. ttl.sh and Gitness offer a solution. ttl.sh is an ephemeral Docker image registry that allows for the creation of temporary image tags with built-in expiry. Gitness, an open-source platform by Harness, simplifies the management of source code repositories and development pipelines. This blog post will explore how these tools can be used to create temporary CI environments to expedite development processes.

The Problem

Multiple developers working on different features can lead to merge conflicts and a bloated image registry. Traditional CI environments often retain Docker images from feature branches too long, which complicates image management and increases the likelihood of testing against outdated or incorrect images.

The Solution

The workflow in the diagram leverages ttl.sh and Gitness to manage these challenges:

1. **Feature Branch Workflow:** The creation of a pull request (PR) for a feature branch triggers a build in Gitness.
2. **Image Creation:** Gitness constructs a Docker image from the feature branch, using ttl.sh for tagging with a UUID and a time-to-live (TTL) limit. This process does not require credentials and maintains privacy.
3. **Automated Testing:** The image is put through automated tests. Should the tests fail, developers are notified; if they succeed, the process proceeds.
4. **Image Promotion:** After a PR passes all checks and is merged, the image is given a permanent tag and pushed to a central image registry, which at this point, requires proper credentials.

This approach ensures that only quality-assured images make it to the central registry, reducing the clutter and promoting a cleaner CI process.

Demo

Let's construct this setup step by step. Starting with pushing an image to ttl.sh through Gitness, I'll guide you to resources for completing the remaining components of the system.

Begin with the Gitness documentation to set up your first project. You can start a new repository or link an existing one from GitHub or GitLab. The specific source code repository can be any; the essential requirement is the presence of a Dockerfile.

Navigate to "Pipelines" within your repository and create a new pipeline. Gitness will suggest a sample pipeline. Replace it with the following YAML:

You can find a gist version of the above YAML here.

Observe the image repo and tag. ttl.sh is the image repository name and the UUID (`xxxx-yyyy-nnnn-2a2222-4b44`) is the image name. You can replace the hard coded image name with a Gitness secret for a dynamic image name. Click on Secrets and add a new Gitness secret called `random_image_name`. Now you can update your pipeline YAML so that the image name is not fixed:

```
repo: ttl.sh/${{ secrets.get("random_image_name") }}
```

Save the pipeline and click on **Run**. After executing the pipeline, your output should resemble the following:

Next add the following steps yourself:

- Add a run step to execute tests on the container you just built and pushed.
- Add a trigger so that when a pull request is opened, Gitness can automatically trigger pipeline execution.
- Add a slack plugin step so that failed tests trigger slack webhook and notification.
- Add another run step so that when all tests pass, the image tag is retagged and pushed to a private image registry. You'll need authentication at this step.

Security Considerations

While ttl.sh's no-credential, ephemeral approach offers flexibility and simplicity for CI environments, it introduces unique security considerations. The convenience of pushing images without credentials is counterbalanced by potential security risks â namely, the

possibility of unauthorized image pulls. You can mitigate these risks through short-lived images and using UUIDs for image names:

Short-Lived Images: By design, images in ttl.sh are ephemeral. Setting a short expiration time for an image means it's available for a limited time window, reducing the risk exposure period.

Use of UUIDs: Incorporating UUIDs into image tags significantly lowers the risk of unauthorized access. The randomness and complexity of UUIDs make it exceedingly difficult for someone to guess the image name and pull it without authorization.

However, for production environments, organizations might consider deploying a private version of ttl.sh. This allows for more control over the security aspects, such as network isolation, access control, and auditing capabilities.

Takeaway

Integrating ttl.sh and Gitness creates an ephemeral CI environment that streamlines the build and test process. It helps to avoid the accumulation of outdated images and keeps the pipeline lean. This method is not just about speed; it's about maintaining a manageable and efficient development workflow. Adopting these tools can lead to more frequent and dependable software delivery for your engineering teams. Check out and follow Harness YouTube channel for technical content on Gitness and more.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/shiptalk-podcast-starting-an-sre-practice-in-a-critical-industry>

ShipTalk Podcast - Starting an SRE Practice in a Critical Industry

In this episode of ShipTalk, we are joined by Dr. Vlad Ukis, who is the Head of R&D for Siemens Healthineers and author of Establishing SRE Foundations.

Podcast [Audio]:

Podcast [Video]:

SRE Journey at Siemens and Your Organization

Medical devices and healthcare technology can be viewed as a critical industry. Lives and quality of care can be at stake if systems or platforms malfunction. Dr. Ukis walks us through the journey that Siemens has gone through from sending USB sticks hospital to hospital to operating a modern SaaS.

Operating a SaaS, Siemens early on recognized the need to scale the science of reliability. Dr. Ukis gives great advice to individuals or organizations starting out on the journey. For example if you had 90 days or a quarter to experiment:

Taking a baseline is critical when starting out. You can not improve what you can not measure. With newly started SRE teams, a good amount of time is taken creating baselines and building trends off of what observability data is available. At Harness, we help bubble this information up quickly to be taken into account when executing your delivery pipelines.

Harness, Your Partner on Your Reliability Journey

The Harness Platform is robust in helping extend your resilience capabilities. From having the ability to organize and manage your SLOs to experimenting with chaos experiments, Harness has a broad set of capabilities. The Harness Platform can also validate deployments with observability data and automatically roll back if needed. Feel free to sign up for the Harness Platform to start to further your resilience goals.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/helm-container-registries-harness-ci>

Push Helm Charts to Container Registries with Harness CI

The Open Container Initiative (OCI) is a Linux Foundation project started in 2015, to create open industry standards around container formats and runtimes. Artifact hosting services such as Docker Hub, Google Artifact Registry and Azure all support OCI artifact in their products.

In the dynamic landscape of Continuous Integration (CI) and Continuous Deployment (CD), a user recently faced the challenge of streamlining the process of publishing OCI Helm charts into a Container registry. The user highlighted the need for a native solution

within Harness CI, eliminating the manual scripting required by each user to accomplish this task. Specifically, the user aimed to publish OCI Helm charts to Google Artifact Registry, and the desire was to enhance Harness CI's already robust support for Helm charts consumption.

Identifying the Challenge and Solution

The user's request was to implement a native component within Harness CI to seamlessly publish OCI Helm charts into a Google Artifact Registry. The motivation behind this feature was to provide users with a standardised and efficient solution, eliminating the need for custom scripts. The objective was clear: enhance the CI pipeline by offering a dedicated component that simplifies the process of publishing Helm charts.

Introducing the drone-helm-chart-container-registry Plugin

Responding to this user's need, we are excited to introduce the drone-helm-chart-container-registry plugin. This is a Drone plugin, which is natively supported by Harness CI.

The plugin is designed to package and push Helm charts to a Docker Hub or Google Artifact Registry effortlessly. By integrating with Harness CI, this plugin eliminates the need for manual scripting, providing users with a seamless solution for publishing Helm charts.

Integration in Harness CI

Integrating the plugin into your Harness CI pipeline is straightforward. Define the plugin as a step in your pipeline YAML:

The plugin provides a range of customisation options, ensuring flexibility for various use cases:

Conclusion

Whether you are a seasoned DevOps professional or new to CI/CD, the drone-helm-chart-container-registry plugin simplifies the Helm chart publishing process. It seamlessly integrates with Harness CI, offering a native solution for packaging and pushing Helm charts to a Container registry.

For additional information, updates, and community support, explore the repository on GitHub.

Enhance your CI/CD pipelines with the native Helm chart publishing component in Harness CI, sign up for your free Harness account today!

If you would like to learn more, schedule a demo to see how Harness can revolutionise your software development process.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

[Documentation](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[Case studies](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[We want to hear from you](#)

Enjoyed reading this blog post or have questions or feedback?
Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/ci-ruby-test-intelligence>

Run Ruby Builds Up To 4x Faster Using Harness CI Test Intelligence

Companies strive to deliver software of the highest quality at the lowest possible cost. The quality of every software project is determined by balancing time, resources, and scope, otherwise known as the project management triangle. Since these three factors are linked, delivering high quality software quickly and efficiently almost always means sacrificing scope (features).

The exception to this rule is **innovation**.

Harness Test Intelligence (TI) helps companies innovate by decreasing test execution time, without reducing quality, sacrificing features, or increasing costs.

Harness Test Intelligence, which debuted in October 2022, is one of the features that makes Harness CI up to 4x faster than other CI solutions. TI reduces test time by running **only the subset of tests relevant to the code changes that triggered the pipeline**. TI supports Java, Kotlin, Scala, C#, and today we are thrilled to announce that support for Ruby is currently in public preview.

Demo

Vagrant is a popular open source project with over 5,000 Ruby rspec tests. Check out this demo to see how TI reduces Vagrant's Ruby test times over 90% compared to other CI providers.

In this post, we will give you an overview of TI, along with steps to reproduce the above demo in your own Harness account.

Where Test Intelligence Fits in Your SDLC

Testing is a vital component of every Continuous Integration pipeline. Testing ensures the quality of your application before code advances to the next phase in your SDLC. Running all tests on every code change is expensive and time-consuming. This leaves developers with the difficult decision of running all tests when introducing changes or to deferring tests until later in the development cycle.

Author Robert Galanakis says, "The fastest code is the code which does not run," in the spirit of writing simple, maintainable code. We can extend this perspective to software testing: *The fastest tests are the tests which do not run.*

How Does Test Intelligence Work?

Using Machine Learning (ML), Test Intelligence examines changes to source code and tests to determine what tests should be run for a given change. When a code change is pushed to a repository, TI uses the following metrics to select tests:

- **Changed code:** TI uses the list of files changed in the commit range to select tests that are associated directly or indirectly with the changes.
- **Changed tests:** When a test is changed, TI selects the test for execution, even if the code related to the test hasn't changed.
- **New tests:** When new tests are added, TI finds any correlations between any new or existing code, and selects and runs those tests.

TI is always up to date and syncs when you merge code to any branch.

Try Ruby Test Intelligence Yourself

To reproduce the above demo in your Harness account, follow these steps.

If you'd like to learn more, schedule a demo to see how Harness can revolutionize your software development process.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/hygiene-in-sdlc-a-key-to-engineering-efficiency>

Hygiene in SDLC: A Key to Engineering Efficiency

In the realm of software development, ensuring a robust and streamlined Software Development Life Cycle (SDLC) is paramount. While many focus on the technical intricacies and methodologies, there's an underlying aspect that holds equal importance: hygiene in SDLC processes. This blog delves into the significance of hygiene within SDLC and how it can pave the way for deriving valuable insights and metrics.

What is Hygiene in SDLC?

In essence, hygiene in SDLC refers to the practices, procedures, and protocols adopted to maintain the integrity, reliability, and effectiveness of the software development process. It is important to maintain hygiene across all the aspects of SDLC. It starts from the very beginning of understanding the requirements from various stakeholders and documenting them. It then flows through the different phases from design to implementation where several decisions will be made based on the best practices and ensuring the code quality is maintained.

Why is Hygiene important?

Hygiene in SDLC serves as a foundational pillar that significantly influences the quality, reliability, and sustainability of software solutions. By emphasizing standardized practices, fostering cross-functional collaboration, and proactively addressing risks, hygiene paves the way for delivering software solutions that are robust, secure, and aligned with stakeholder expectations. This adherence not only enhances software quality and security but also fosters a culture of excellence, innovation, and accountability.

The Link between Hygiene and Insights

Maintaining hygiene in SDLC is not merely about adherence to protocols; it's about fostering a culture of excellence and continuous improvement. Here's how hygiene directly contributes to generating valuable insights:

- **Accurate Tracking:** Maintaining hygiene across tools ensures that issues are accurately categorized, prioritized, and tracked throughout the development process.
- **Enhanced Insights:** A well-maintained system facilitates the extraction of meaningful metrics and analytics, providing deeper insights into recurring challenges and areas for optimization.
- **Proactive Problem Resolution:** Hygiene practices such as regular updates and clear documentation enable organizations to proactively address potential pitfalls and optimize resource allocation.
- **Data-Driven Decision Making:** By consistently documenting processes and maintaining code quality, and having data hygiene results in more clarity, better efficiency and less context switching that can lead to trusting the data and making informed decisions. For instance, tracking code quality metrics over time can highlight areas for improvement and guide resource allocation.

How can hygiene be improved?

As the saying goes, "only the things that get measured can be improved," this underscores the importance of establishing metrics and benchmarks to enhance hygiene within the Software Development Life Cycle (SDLC). By systematically evaluating and optimizing key areas, organizations can foster a culture of excellence and continuous improvement.

- **Define Clear Metrics:** Begin by identifying relevant metrics that align with the desired hygiene standards. This may include some of the key elements such as Insufficient Acceptance Criteria, Sprint, Fix Version etc.
- **Promote Accountability and Ownership:** Foster a culture of accountability where teams take ownership of maintaining hygiene standards. Encourage regular reviews, discussions, and collaborative efforts to uphold these standards across the entire process.
- **Encourage Continuous Feedback:** Establish a feedback-rich environment where teams can share insights, identify challenges, and propose solutions to enhance hygiene. Regular retrospectives, stakeholder feedback, and collaborative discussions can facilitate continuous improvement.
- **Revisit Thresholds as and when Required:** Regularly revisit and reassess established thresholds and benchmarks to ensure their relevance and effectiveness. Adjust thresholds based on evolving requirements, feedback, and organizational objectives to maintain alignment with hygiene standards.

Embracing Hygiene for Long-Term Success

To harness the full potential of SDLC hygiene, organizations must cultivate a culture that prioritizes quality, collaboration, and continuous improvement. This entails:

- **Training and Development:** Investing in training programs and fostering a culture of knowledge sharing empowers teams to stay abreast of industry best practices and emerging trends.
- **Collaborative Approach:** Encouraging collaboration among cross-functional teams, including developers, testers, product managers, and analysts, fosters a holistic understanding of the software's requirements and objectives.â
- **Feedback Loop:** Establishing a robust feedback loop, encompassing regular retrospectives and stakeholder feedback, ensures that lessons learned are integrated into future projects, driving continuous improvement.

Conclusion

Hygiene in SDLC is not a mere procedural aspect; it's a foundational pillar that underpins the success and sustainability of software development endeavors. By prioritizing hygiene and leveraging it as a catalyst for generating valuable insights and metrics, organizations can navigate the complexities of software development with confidence, agility, and foresight.Â

To explore how SEI can transform your software development process, we invite you to schedule a demo with our experts.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?
Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/architecting-for-cost-savings-on-bigquery>

Architecting for Cost Savings on BigQuery

This blog was co-authored by Nikunj Bajtaya, Principal Software Engineer and Abhijeet Sharma, Senior Software Engineer at Harness.

â

At Harness, we leverage our own products extensively in our software development lifecycle, as would be expected from a company such as ours. Our engineering team is responsible for our CCM software architecture, and has direct visibility into our production costs by team and by module (among other views), and we review our cloud costs regularly to find areas for optimization for both our product costs and product performance.Â Â

It was during one of these reviews that we began to see our BigQuery analysis costs rising beyond what we were expecting from our rapidly growing installed base. Our CCM product dashboards and reporting are built on top of BigQuery, so each time our users view dashboards or run reports, it consumes on-demand compute capacity (or compute slots by the new pricing), alongside the monthly charges for storing the cost data itself.

What led to starting the project?

As the Harness CCM installed base has grown in both quantity and size of customers, we are seeing an increase in customers for whom we are ingesting GBs of cost data daily. As we began ingesting historical data for a major new customer, we were watching both our query and storage costs spike at an alarming rate. As you can see in the Perspective below, our BigQuery costs were up 73% over a 6 month period!

Customers are getting more mature in their FinOps practices, expanding their reporting among product teams, so we were also seeing an increase in usage of our Perspectives and BI Dashboards. Bigger, longer term dashboard views led to larger and more frequent calls to BigQuery, which was also impacting our costs. For example, one customer wanted to see 2 years of cost data in a single dashboard view. Thatâs a high volume of data!Â Â

The quantity of data was also beginning to impact the performance of our dashboards, as queries were taking longer and dashboard response times were slower, especially for customers with large datasets. Our customer experience is extremely important to us, so we knew we needed to take a hard look at our BigQuery structure to improve user experience, while also making a positive impact on our productâs bottom line.Â

BigQuery Pricing

As an aside, when we started our project to optimize our database structure, we also needed to assess our transition to the new BQ Editions pricing, and how it would impact us. When we first introduced Harness CCM, on-demand compute pricing was the only available option, but Google introduced BQ Editions earlier this year, and given the growth of our dataset, it made sense to assess making a shift.Â

This became an urgent change once Google announced a 25% increase in on-demand pricing, which would directly impact our bottom line. Our queries were growing in size of data retrieved, but we were looking relatively healthy for our slot usage as itâs billed in BQ Editions. So we made the switch, avoiding an immediate 25% cost increase!

What the Engineering Team Thought

As we brainstormed several different areas for improvement, we realized that we needed even more granular cost information by customer and use case, so we started by building a new BI dashboard to provide deeper insights into BQ costs by customer. This helped us segment BQ costs by customer size as well as activity levels, which allowed us to better characterize our cost drivers.Â

After reviewing the data, the team determined that we would get the best improvements for both cost and performance if we tackled the following challenges:

- Flatten out nested labels to reduce UNNEST operations
- Caching data in Looker with Persistent Data Tables
- Review data ingestion frequency
- Improve gatekeeping by license type

Flattening Out Nested Labels

We ingest customers labels from each of the cloud service providers our customers use, storing them in a BigQuery table as nested labels, as using nesting is a Google best practice to denormalize data storage and increase query performance while maintaining hierarchical relationships between labels.

During our investigations we discovered two things: first, that label queries were the single biggest contributor to our BigQuery costs; and second, that the way we had originally implemented our label storage was causing a high amount of UNNEST calls, increasing both response times and data transfer costs.

By flattening out how we store labels into a columnar format, we could eliminate the need for UNNEST calls which would improve both query performance and data transfer costs. This wasn't a trivial change, but the effort yielded significant returns for us. We not only improved our immediate cost concerns, but ensured that this part of our database architecture was ready to handle the acceleration of customers adopting Harness CCM.

Looker Persistent Data Tables

Our customer facing Business Intelligence (BI) dashboards are powered by Looker, so that customers have a powerful BI tool at their fingertips for analyzing cloud costs. When we discovered that label queries were driving so much of our BQ costs, we realized that we could reduce a portion of that workload by caching data within Looker by using their persistent data tables (PDT).

We started by implementing them manually for our biggest customers in terms of dashboard usage to assess the impact on query costs and performance. It made a material impact on the frequency of BQ retrievals, so the next step is to automate the creation of PDTs when customers reach a certain threshold of dashboard usage within their account.Â

Review Data Ingestion Frequency

We always want to provide the most up to date information for our customers, so we pull cost data from the cloud providers multiple times per day to give our customers the most up to date costs. However, as part of this inquiry, we took a look at exactly how often customers themselves were looking at this data, whether through directly accessing the Harness console, or by sending automated, scheduled reports via email or Slack.Â

We realized that we had anticipated a use case that no one actually needed. Reports are generally sent weekly, or at most every day or two. Our own anomaly detection generally runs once per day well. So we were ingesting data (with the associated data transfer fees to BQ), more often than was needed / wanted, running up our own cloud bills unnecessarily.Â

We decided that once per day data ingestion from the cloud providers fit the actual use case our customers wanted. Implementing this change contributed a fair amount to our cost reduction efforts.Â

Improve Gatekeeping by License Type

Gatekeeping is easily one of the most impactful, but sometimes neglected, tasks for managing data, and as we reviewed our dashboards by customer usage, we realized that we were retrieving data into BigQuery that didn't need to be.Â

For context, only our Enterprise license customers have access to our BI Dashboards feature for advanced analytics. Anyone using our Free tier, free trial, or engaged in a proof of value (POV) can't access the BI Dashboards in our console. But we weren't gatekeeping the data ingestion for our non-paid customers, and were ingesting data that wouldn't be used.

Implementing a license check prior to ingesting data into BigQuery for Looker reduced the amount of data we were storing, which in turn also reduced our data ingestion fees for our existing customers. More importantly, it is also cost avoidance for the future, especially as our POV activity continues to grow.

Results Speak for Themselves!

After we implemented all of these BigQuery architectural changes, we were very pleased to share with the company that we had **reduced our direct BigQuery costs by nearly 45%**, while improving dashboard performance for our customers, especially our largest customers with large datasets. And don't forget that we also had an indirect 25% savings by moving over to BQ Editions pricing.

These changes were critical for us as we continue to grow and scale our customer base. We're now in a better position to keep our BQ costs in control, while giving our customers a better user experience.

Closing Thoughts

This exercise involved numerous technical discussions within our internal teams, several different POCs to prove our theories and test changes, as well working closely with the Google BigQuery team to drive the correct implementation. Without a concentrated effort on the larger groups part, this change wouldn't have been as successful as it is. We are very appreciative of the work that everyone involved put into this project.Â

This was just one example of BigQuery architecture changes that can be made to reduce cloud costs and improve performance. You can read about more cloud cost savings strategies we implemented a few years ago in this blog. If you want to learn more about Harness CloudÂ Cost Management, read about it here!

â

Editors Note: This is one of a new series of blogs that are focused on how Harness Cloud Cost Management users are âArchitecting for Cost Savingsâ, highlighting their experiences where cloud architectural decisions and changes have positively impacted cloud costs. Other blog posts in the series: Architecting for Cost Savings at Tyler Technologies

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/kubecon-2023-recap>

KubeCon 2023 NA Recap - Developer Experience is Monumental

KubeCon + CloudNativeCon North America has wrapped up, and the buzz it generated is still in the air. With the halls of Chicago's convention center now quiet, those of us who attended are left with a wealth of new insights and ideas. The conference proved to be a fertile ground for learning and networking, with cloud native professionals / technologists from leading open projects from around the world sharing their knowledge.

At Harness, we were right there in the thick of it, participating in the Co-Located events and engaging with the community. Our team dove into conversations, contributed to discussions, and absorbed a lot of feedback on everything from technical workflows to project management.

For those who couldn't make it, or for attendees who want to revisit the highlights, our recap will bring you the key points from KubeCon 2023. We'll cover the sessions that made us think, the keynotes that motivated us, and the informal chats that often lead to the best ideas.

The Pulse of Cloud Native: KubeCon Session Breakdown

As cloud native technology matures, the focus of KubeCon sessions has honed in on a few key areas: the construction of layered abstractions over Kubernetes, addressing the unique challenges within specific industry verticals, and the role of AI in cloud native spaces. While it's impossible to cover all the sessions within this recap, we've curated a subset of the extensive and enlightening discussions for you. Here are some highlights:

Argo and Flux

Since deploying Kubernetes to cloud or on-prem environments has largely been standardized, the community is now working on adding orchestration on top of orchestration. The word âScaleâ has featured in the session title at no fewer than 30 talks. Argo and Flux have taken center stage as key orchestration tools for declarative GitOps management of Kubernetes workloads.Â

These projects have been in the Cloud Native Computing Foundation for a few years now. Organizations are beginning to report their success in using them to manage not just hundreds or thousands of nodes, but hundreds - to - thousands of clusters and deployed applications. The piling up of abstraction layers is leading to an everything-as-code approach for which a multitude of vendors and community projects have stepped up to offer their resources and solutions.

Challenging Verticals

With the maturing of deployment architectures, a marked shift has also taken place to solve problems in industry verticals where cloud native has been a challenge. One example is deploying to secure or air-gapped environments. Crossing the air gap remains a wrench in the automation process. Organizations that operate in that space developed a couple of practical solutions:

- Develop as much as possible in unclassified environments while ensuring the application is as self-contained and portable.
- Closely engage with the open source community so that CVEâs are identified and acted on quickly prior to crossing the air gap.

AI

As might be expected, artificial intelligence was prominent in several sessions. The industry is still determining whether AI (namely LLMs) has reached the peak of the hype cycle, and what role it plays in cloud native. Compared to the general buzz, there were relatively few sessions specifically on AI/ML. Even then, those sessions were less about consuming generative AI, and more about deploying custom AI/ML applications.

The focus then was largely on the massive compute resources required for AI training and deployment. The community is still catching up when it comes to developing open source alternatives to the dominant proprietary stacks, but some projects have emerged to propose community-driven approaches to model training (federated LLMs for example).

On the Decline: Yesterday's Hot Topics Take a Backseat

Since last yearâs KubeCon, a few parts of the ecosystem have started to either mature or decrease in popularity. While by no means on the way out, we noticed a few of the following were less emphasized in this yearâs talks and vendor showcases.

Pure Kubernetes Implementations

While the number of certified Kubernetes distributions have ticked back up to historic highs, vendors are now focusing on the application layer rather than leading with K8s. Deploying Kubernetes to the public and private cloud is now largely standardized. Organizations are therefore now turning their infrastructure attention to edge cases such as maintaining legacy VM workloads, or else are moving up the stack into serverless.

Unchecked Developer Autonomy

Security is no longer optional. The rise of software supply chain attacks and rise in open source and cloud resources have spurred companies to balance developer productivity with the need to secure their data and environments. While developer experience is still paramount, the ecosystem is realizing that âyou build it, you run itâ should not mean âno guardrailsâ. Organizations are moving toward a more centralized experience, implementing managed platforms like backstage. Tools like keycloak and OPA have risen to âmust-havesâ in cluster administration and permission management.

Dialogues at the Booth: Harnessing Customer Voices

Harness has been a long time advocate of the Cloud Native community and ecosystem, and is proud to help sponsor KubeCon. We had a multitude of great conversations across our ArgoCon, BackstageCon, Litmus Chaos, and main-floor KubeCon booths. Helping support the next generation of workloads by reducing friction in software delivery are top of mind for many individuals we got to interact with over the course of the week.Â

Â

Engineering pillars and experiences that previously in a non-cloud-native world such as security, scalability, and robustness would be afterthoughts are front and center during the development cycle now thanks to the ever increasing burden of shifting left.Â The Harness Platform is well poised to help reduce significant toil as more expertise is disseminated across your evolving delivery pipelines. We are excited to see what is in store over the course of the year before KubeCon 2024.Â Â

Next Stop, Salt Lake City For KubeCon NA 2024

As we wrap up our journey at KubeCon 2023, we're taking a moment to reflect on the rich tapestry of ideas and innovations that were shared. This year's event has not only reinforced the pivotal role of cloud native technologies in shaping the future of software but also highlighted the evolving landscape where some trends rise and others give way to more pressing innovations.

At Harness, we've had the privilege of engaging directly with the community that's pushing these boundaries. Our conversations at the booth brought us face to face with the pulse of the industryâwhere the enthusiasm for Kubernetes (K8s) is as robust as ever, and the love for our four-legged friends (K9s) continues to bring smiles and a sense of camaraderie.

â

Looking beyond the current cloud native vistas, we're excited about what's on the horizon. As we set our sights on KubeCon in Salt Lake City as the next flagship conference gathers adopters of cloud native technology, we carry forward the insights and feedback we've garnered here in Chicago. The dialogue doesn't end with the closing of the conference doors; it's just the beginning. We invite you to continue these conversations with us, explore how Harness can streamline your DevOps journey, and share in the excitement for what's to come.

Until we meet again, let's keep pushing the envelope in our respective fields, inspired by the collective wisdom of KubeCon 2023. And remember, whether it's for your infrastructure needs or a harness for your K9, Harness is here to support you. See you in Salt Lake City!

-Nick, Dewan, Ravi

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/harness-product-modules/service-reliability-management>

Source URL: <https://www.harness.io/harness-product-modules/feature-flags>

Source URL: <https://www.harness.io/blog/navigating-success-unveiling-harness-game-changing-user-experience>

Navigate a New Way: Your Dedicated User Experience

At Harness we strive to continuously improve our products to provide you with the best user experience. In our efforts to do so we found that our one-size-fits-all interface no longer meets the needs of our ever growing platform. Some of these growing pains include difficulty managing resources that are shared across modules and changing scopes. It has become clear that our navigation platform needs to mature alongside our platform

Introducing Nav 2.0, a new design that updates the information hierarchy of our platform to provide a curated user experience for multiple personae. Brace yourselves for a brand new look featuring an interface designed to reduce confusion and save you valuable time, allowing you to get ship done.

â

Still Curious? Let's Look at Some Features:

Stay Focused with our Scope Selector: Look to the top left of your screen to find the new scope selector. Quickly navigate to your desired level of specificity and see an overview of that account, organization or project.Â

Don't Be Distracted, Choose a Module: Focus your workspace by choosing an individual module. Doing so removes distraction from your workspace and reduces the navigation options available to help you find what you need, faster.Â

No More Digging! Find Common Resources at the Top: Our brand new productivity view will now allow for immediate access to shared scope resources such as pipelines or executions without having to navigate through a module.

Never Forget Where You Are: Always keep track of your workflow with our new breadcrumb trail on each page that lets seamlessly travel back in your hierarchy.

Access Settings Easier: Configure your platform quickly with a brand new settings page that can be accessed from all three scopes.

Don't Worry, The Wait is Short

Get ready to embrace the enhanced Harness user interface, arriving in the near future. Soon you'll have the opportunity to explore these improvements firsthand. Your feedback is crucial to us, so when you dive into these exciting new features please don't hesitate to share your thoughts and suggestions. To try out the new features find the "My Profile" menu on the bottom left of your screen and give us feedback using the "Help" menu above it. Let's navigate to success together in 2024!

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/harness-product-modules/gitness>

Source URL: <https://www.harness.io/blog/contribute-to-harness-cli>

How to Contribute to the Harness CLI

Last September, we released a command line client for working with Harness. The Harness CLI is open sourced under the MIT license, and it provides an alternative to the Web UI for creating and managing pipelines, environments, and other CI/CD workflows. The purpose of this post is to provide some guidance around contributing to the project, including a retrospective of my own first PR, with lessons around contributing to open source in general. You're highly encouraged to read through the release blog for more on our motivations for developing a command line client.

Learning From My First PR

I recently opened my first pull request for the CLI project: a small tweak around creating secrets in Harness' built-in secrets manager. The team was kind enough to approve and merge the PR last week. While this was my first code contribution to the CLI, I've relied on it heavily as a user in my day-to-day work. Here are a few things I learned in moving from a user to a contributor. These are lessons can probably be applied to almost any open source project.

Don't be afraid to start small

Open source projects often have community expectations that far exceed the number of contributors and maintainers. OpenSSL famously had one full-time maintainer when the library's Heartbleed vulnerability was discovered back in 2014. Many project maintainers at minimum split their responsibilities between their business facing role and open source work. They're excited then when community members offer their help, as they almost always could use it.

Some projects have a CONTRIBUTING.md in the repository root that lays out contribution guidelines. Otherwise, opening a GitHub issue (for larger contributions) or pull request (for smaller patches) is a good bet. A maintainer will then be happy to point you in the right direction.

Feel free to offer code if that's your skillset. However, contributions to docs are just as valuable. If you don't feel comfortable around the codebase, any efforts toward enhancing or clarifying core documentation, tutorials, and diagrams should be well received. Skilled technical writers are sorely needed in open source. My recent contribution to the Harness CLI was a blend of both code and docs. I added an argument to the secrets management command, and also updated the help docs to include the new required parameter.

Understand the project's ecosystem

As I mentioned previously, don't be afraid to contribute even if you're new to the project, with copyediting the docs as a great starting point. For more substantial contributions, you should strive to understand both the codebase and user experience. The Harness CLI, for example, implements command line equivalents for workflows performed in the Harness Web UI. It uses REST API calls for the creation and management of most entities. You'll want familiarity with these concepts, and should also have a Harness account for learning and testing.

Beyond a technical understanding of the software, be sure to also align with the codebase in style and syntax. If variables are camel case, use camel case. If functions are heavily broken up, try to follow that pattern. Perfection isn't expected of course. Just make an effort to speak the language of those who will be reviewing your PR.

In my case, you'll see I wrote my code to match the project's conventions for command flags and variable names.

Leverage AI coding assistants... to a point

I'd be remiss if I didn't bring up generative AI and coding assistants. These tools help developers feel more confident writing code that otherwise might intimidate them. Please do use AI to develop code snippets, mermaid diagrams, improvements to documentation wording, or any other contribution you'd feel would be helpful. Still, don't generate boilerplate, untested code and submit a PR without

understanding the what and why. That would just waste the time of reviewers.

Instead, prompt the UI with problems motivated by your knowledge of the project. Then carefully review the output and modify as needed to match the codebase's style and contributor guidelines. AI and code assistance are great productivity tools; just combine their use with care and intentionality.

Contributing to open source is a fantastic way to stretch your technical skills and network with developers from all over the world. Harness hosts several open source projects, including the Harness CLI as well as the development platform Gitness. We welcomes contributions, so feel free to open an issue or PR. Please also consider joining our community Slack channel to interact with other Harnessians as well as the broader CI/CD community.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/harness-product-modules/sto>

Source URL: <https://www.harness.io/blog/harness-sei-code-quality-enhancement>

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

Introduction to SEI's Transformative Power

In the ever-evolving landscape of software development, the significance of producing high-caliber code is undeniable. This is where Harness Software Engineering Insights (SEI) shines, guiding teams toward elevated software quality, enhanced productivity, and overall excellence. Here, we delve deep into the pivotal role of SEI's Quality Module in aiding teams to gauge, supervise, and uplift their code quality.

Understanding SEI's Quality Module

The Trellis Framework: At the heart of SEI's transformative potential is the industry-validated Trellis Framework. This intricate design

provides a comprehensive analysis of over 20 factors from various Software Development Life Cycle (SDLC) tools, enabling teams to efficiently track and optimize developer productivity.

â

Measuring Product Quality with SEI

Lagging Indicators: Retrospective Excellence

Lagging indicators are retrospective measures, offering insights into past performance. Let's break down these metrics:

â

Defect Escape Rate: This metric, crucial for understanding production misses, measures the percentage of defects that go undetected during production and reach the customer. A higher defect escape rate can signal poor quality control, leading to customer dissatisfaction.

Escapes per Story Point or Ticket: This indicates the number of defects per unit of work delivered. An elevated number here can point to quality lapses in development.

Change Failure Rate: This metric measures the percentage of changes leading to failures, indicating the robustness of the product.

Severity of Escapes: This highlights the seriousness of defects, with higher severity demanding urgent attention.

APM Tools - Uptime: Measuring product availability and performance, a higher uptime percentage is indicative of good product quality.

Customer Feedback: Direct customer feedback, both positive and negative, provides valuable insights into product quality.

â

Leading Indicators: Proactive Quality Assurance

Leading indicators predict current or future performance. We explore these further:

â

SonarQube Issues: This includes code smells, vulnerabilities, and code coverage. Issues flagged here can indicate quality concerns in the codebase.â

Coverage % by Repos: Evaluating code coverage percentage across various repositories.

Automation Test Coverage: A higher percentage here suggests a robust, reliable product.

Coding Hygiene: Measures such as code reviews and comments improve code maintainability and reduce defect risks.

Program Hygiene: This includes acceptance criteria and clear documentation to ensure the product meets requirements.

Development vs Test Time Ratio: A balanced ratio is crucial for product quality.

â

â

Test Metrics

â

Automated Test Cases by Type: Categorizing test cases into functional, regression, performance, or destructive types.

â

â

Test Cases by Scenario: Differentiating between positive or negative scenarios.

Automated Test Cases by Component View: Providing a component-wise breakdown.

â

â

TestRail Test Trend Report - Automation Trend: Showcasing the trend of total, automated, and automatable test cases.

â

â

Structured Engineering Improvements

SEI's architecture integrates with CI/CD tools, offering over 40 third-party integrations. This structured approach aids in goal-setting and decision-making, driving teams towards engineering excellence.

Resource Optimization for Efficiency

Beyond metrics, SEI assists in resource allocation optimization, aligning resources with business objectives for efficient project delivery.

Centralized Visibility through Dashboards

SEIâs dashboards provide a holistic view of the software factory, highlighting key metrics and KPIs for better collaboration and workflow management.

Conclusion

Harness Software Engineering Insights, with its Quality module, stands as a beacon for development teams, combining metrics, insights, and tools for superior code quality. To learn more, schedule a demo with our experts.

â

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/how-the-size-of-a-pull-request-can-impact-developer-experience>

How the size of a Pull Request can impact developer experience?

Code reviews are a critical component of the software development process that helps maintain quality and ensures that new code

integrates seamlessly into an existing codebase.Â

However, the size of pull requests (PRs) can significantly impact the efficiency and effectiveness of these reviews.Â

The size of a PR in many scenarios becomes a major factor that affects the reviewer's experience and eventually the overall health of a software project.

What's the best size for a PR?

A large body of research has shown that code reviews are the most cost-effective way of finding defects in code. Studies show that PRs under 105 lines of code work best.Â

This benchmark is more than just a number. It reflects a deeper understanding of a developer's experience during the review process. But why is the size so crucial, and what happens when PRs become too large or remain within the recommended range?Â

Small vs. Big PRs: What's the Difference?

The experience of reviewing a PR varies significantly depending on its size. Let's compare reviewing a 105-line PR to one that's 400-800 lines.

Small PRs (Under 105 Lines)

Large PRs (400-800 Lines)

In theory, PR size is important. But in real life, it's even more crucial.Â

Large PRs can slow things down and stress the team resulting in delayed releases and making things harder for everyone. It's like when you had a big book to read for school.Â Many would choose Cliff Notes instead because the whole book was too much.Â

Small PRs are like short articles that get full attention, while large PRs are like big books that people just skim through.

The Developer Experience: More Than Just Speed

While delivering work quickly is important, the experience you provide to your developers and engineering team is equally crucial. The size of PRs can significantly impact this experience. In a highly competitive market, fostering a healthy developer environment is key.

Small, regular PRs make reviews better and help the team work well together.

How does Harness SEI help in measuring the PR activity?

In managing the challenges of PR sizes and their impact on code reviews, Harness SEI helps in solving the problem of measuring PR activity using SCM reports.Â

SCM reports provide a detailed analysis of activities within SCM tools, offering insights into:

- Individual developer contributions.
- The volume of rework and other code changes.
- Lead time and activities related to PRs and SCM issues.
- Review collaboration.
- Most active repositories and files.

These reports can be customized based on project, repository, time range, and other data points, depending on your SCM integrations.Â

The SCM Committers Report is particularly useful for evaluating individual developer contributions. It provides a clear picture of each committer's involvement in the Collection, displaying:

- The number of PRs they have worked on.
- The number of commits they've made.
- The number of lines of code they have contributed.

This information is crucial for ensuring accountability and transparency, effective project management and resource allocation, and evaluating the performance of developers or teams.

Engineering teams can leverage Harness Software Engineering Insights to first baseline the people, processes and tooling bottlenecks and then drive a continuous improvement process. To learn more, schedule a demo with our experts.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/harness-product-modules/idp>

Source URL: <https://www.harness.io/harness-product-modules/aida>

Source URL: <https://www.harness.io/blog/january-2024-product-updates>

January 2024 Product Updates

January 2024 Product Updates

At Harness, we have been hard at work so you can delight your customers without facing software delivery toil. Here is what has been changing in the previous month across Harness Products.

Continuous Delivery & GitOps

Full Continuous Delivery & GitOps Release Notes

Continuous Integration

Full Continuous Integration Release Notes

Chaos Engineering

Full Chaos Engineering Release Notes

Security Testing Orchestration

Full Security Testing Orchestration Notes

Internal Developer Portal (IDP)

Full Internal Developer Portal Notes

Software Engineering Insights

Continuous Error Tracking

Full Continuous Error Tracking Release Notes

Infrastructure-as-Code-ManagementÂ (IaCM)

âââ

Early Access

â

Continuous Delivery & GitOps

Internal Developer Portal (IDP)

â

Continue the Journey

- Learn more hands-on with our increasing list of Tutorials.
- Further your knowledge with one of our Certifications.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/getting-started-with-the-react-sdk>

Getting Started with the React SDK

What is going on with all of my amazing friends in the developer community? In this post we are taking a look at getting the Harness React SDK up and running in your project. This article is a low barrier to entry with the core code to get started. In the example below

we are going to take what the React SDK GitHub has to offer and shorten it to get you started with your first feature flag in React. If you want a more in depth breakdown of the SDK, head over to our React SDK GitHub repo and take a look!

Requirements

Before we begin installing and using the React SDK, make sure you have these requirements on your system:

- Installed Node.js v12 or a newer version
- Installed React.js v16.7 or a newer version

I am using Vite in order to set up my projects in React. You can use Create React App if you choose, but Vite has a lot more features and performance improvements. At the time of this writing, Vite ships with React v18.2.

Installing the SDK

We are going to use npm in this example but I am also providing the install command if you are using yarn as well.

To install the React SDK with npm, type the following command in your terminal of choice:

If you are using Yarn, you can run the following:

Once installed you should see the package added to your package.json file. Run `npm install` to install all your dependencies and generate your node_modules folder.

â

SDK Implementation

We are going to directly modify two files in this example. The first file being the `App.jsx` where all our components are going to be imported. The second file is going to be the component file itself, `Logo.jsx`.

Inside our `App.jsx` file we are going to import our Logo component, PrimaryNav component, Scss file, and the FFContextProvider from the react-client package.

The `FFContextProvider` brings in props like `apiKey`, `target`, etc.

Once we import all our components we are going to build out our App component. The App component gets pulled into the `main.jsx` file as you can see below:

We are creating a component called App and returning all the necessary markup that we need to render our UI.

We wrap the whole thing in our `FFContextProvider` so the block in-between can communicate with the SDK. Since the API key is client side, it is ok to have that be visible. Anyone that can open up the console in the browser can see it.

Reactifying the Logo Component

The only thing we are going to import in our Logo component is the `useFeatureFlag` hook from the SDK.

If everything in your Harness UI is set up for your feature flag, the ID you pass in your `useFeatureFlag` hook should match what it says in the Harness UI.

â

After we import `useFeatureFlag` we create a component called `Logo`. Within that component we declare a `const isEnabled` and set that to `useFeatureFlag('awesomelogohadergradient')`.

We create an `if` statement that will handle whether or not our feature is on or off. We pass in `isEnabled` and if that is true, we show the `h1` with the classes `logo-title` and `gradient` to display our awesome, new gradient. If our feature flag is disabled in the Harness UI, the user will see the default `h1`.

â

â

Then we export our `Logo` component to use inside `App.jsx`. That is it! You have successfully implemented the React SDK into your application with a core UI component. I hope you found this article helpful to get you up and running with feature flags and the React SDK.

â

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/webinar-velocity-with-control>

Webinar: Increased Velocity with Better Control

New webinar: increased velocity with better control

Software is vital to just about every aspect of an organization, from finance to customer engagement and business intelligence. As demand for software skyrockets, the innovation cycle shrinks. This is putting more pressure on engineering teams to deliver faster without sacrificing quality.

Make no mistake, software delivery has never been more important. However engineering teams are often plagued with manual toil, scripting, and wasted time in getting code to production, making it harder to innovate at pace.

Harness' own Martin Reynolds, Field CTO, joined *The Register*'s Tim Philips to discuss the challenges that modern software delivery teams are facing, including what gets in the way of transformation and how to make change without slowing down delivery. Tim and Martin also discussed if it's possible to achieve velocity, quality and governance while maintaining control and how cost management and security can come into play.

The current state of the SDLC is broken

Currently, the software delivery lifecycle (SDLC) is a brittle and inefficient process, due to a build-up of combined sets of tools that have been stitched together. On top of this, businesses are expecting their software developers to do even more â such as shifting left on security. Instead, developers are having to spend time on ops tasks and fixing issues, rather than doing what they do best â coding.

Martin notes that it should be easy for developers to deploy their software. But the reality is â for many organizations, it's not â and often teams get âused to the broken-nessâ. They look at the process in place and think it is too hard to fix â and so hot fixes and workarounds have become the norm.

Where are developers spending their time?

If we look at where developers spend their time, for many, it's not building software.

Many developers are stuck waiting for builds to happen, doing manual testing instead of automating, tracking down how deployments

performed, or unraveling code following a security failure. A Harness survey of over 500 organizations shows that, on average, **40% of developer time is spent on non-coding toil** which is approximately two days a week. Martin thinks it is time to question what the best use of engineers' time actually is, because the current state of software development is far from aspirational.

He points out: "No-one comes into software development saying they want to carry out compliance reviews."

Giving the time back

This is the problem Harness is addressing, by moving forwards to something that's more integrated and flows more easily. Harness helps engineering teams with CI and CD, but also has modules to support every area of the SLDC, including security testing using ML and AI. It learns what looks good and tells you when something's broken or even rolls it back for you.

"That's what Harness wants to do, give developers time back," Martin continues.

Speaking from experience

Martin shared his own experience as an end-user of Harness in his role as a DevOps & IaaS director at previous employer, Advanced. He explained, "We wanted to get a grip on our deployment process and were spending a lot of time trying to consolidate tools like Jenkins, GitLab, CircleCI, and home-grown scripting. This setup broke often, and some deployment processes took weekends, which is not a good place to be."

To see what Harness could do, Martin put it to the test. He carried out a deployment with his team in a day and a half, before watching Harness do the same deployment in just 30 minutes during a call. He noted it's amazing to see so many capabilities come straight out of the box, from Kubernetes container support to canary or blue/green deployments.

He also explains how Harness acts as a partner, bringing whole teams together to understand an issue and help customers overcome their challenges.

How do you know you have been successful?

Harness can help to win over organizations and engineering teams showing success and demonstrating how easy it can be. Read more about how Harness has worked with some of the world's most innovative organizations, helping engineering teams to:

• Eliminate 50% of major production incidents at Ancestry

• Accelerate Citi's release times from days just 7 minutes

• Reduce deployment toil by 75% at United Airlines

You can listen to the full recording of Martin's webinar, "Increased Velocity with Better Control", on *The Register* here.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

[Documentation](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[Case studies](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[We want to hear from you](#)

Enjoyed reading this blog post or have questions or feedback?
Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/using-ai-to-improve-feature-flags-onboarding>

Using AI To Improve Feature Flags Onboarding

Getting Started with Harness Feature Flags

The first step in getting started with Harness Feature Flags is to install the SDK in your code. Sounds simple â and it mostly is!

But, we are always pushing to simplify things to help you get started faster, and we recently rolled out a subtle but cool change to help you do this.

The Historical Feature Flags Onboarding Experience

When you sign up for Harness Feature Flags you're taken to the Getting Started flow. This walks you through creating a flag, creating an SDK key and then getting some setup instructions for the languages of your choice. You can also get back to this flow anytime you want via the sidebar, if you want to see the language-specific code samples again.

â

â

Until now, we had to maintain these code samples manually per language. This wasn't too much work, but it limited how specific the code samples could actually be. But, AI has helped us improve this.

Feature Flags Onboarding Reimagined with AI

Utilizing Harness AIDA, we are now able to generate code samples dynamically. This means instead of only having per-language samples, we can pass inputs like *version* and *framework* to get more customized SDK onboarding code samples to use.

So, instead of Node.js, you can get a getting started snippet for a specific version of Node.js + Express.js.

In the future, we want to keep expanding this to provide even more customization and user inputs based on your specific application architecture.

â

A Smooth Feature Flags Onboarding Experience

As we integrate AI, via AIDA, all across the Harness platform we are looking for every opportunity - big and small - to use AI to help make developers' lives better.

Providing more dynamic onboarding examples is a cool example of where AIDA can make a big difference in a small way by taking the time to find every opportunity.â

Interested in learning more about Harness AIDA, generative AI, and the SDLC?

Request a demo and explore the capabilities of Harness AIDA in today's dynamic software delivery landscape. Check out our Harness YouTube channel to see the product in action!

If you are interested in learning more about how we are infusing AI capabilities across the SDLC, check out other blogs in our AI Feature series to learn more!â

- Introducing Dashboard Intelligence Powered by AIDA
- Introducing CD Error Analyzer
- Introducing Harness Support Powered by AIDA
- Introducing Policy as Code Assistant Powered by AIDA

Our Plan to Infuse AI Across the SDLC

Stay informed about our exciting future developments by keeping an eye on our comprehensive product roadmap.

â

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/backstagecon-2023-recap-developer-experience-top-of-mind>

BackstageCon 2023 Recap: Developer Experience Top of Mind

As the creators of systems and platforms, developers can be seen as the most advanced users in terms of the platforms they create and the platforms they interact with; developers should âjust be able to figure things outâ. With this mindset, developer experience can sometimes take a backseat to âis developer experience worth the effortâ. This belief is aging and what is rising is the investment in Developer Experience or DX.â

Internal Developer Portals or IDPs are designed to drive DX. The hallmark Open Source IDP, Backstage, has its community coming together at a co-located event at KubeCon North America 2023 for the second year. Harness is proud to support and sponsor the event and the movement which IDPs represent with mutual collaboration and learning.Â

Happy Internal Customers = Easier to Create Happy External Customers

The implementers of an IDP would be platform engineering or developer experience teams and the main consumers/customers of an IDP would be developers. Not too dissimilar to the implementation and relationship of software delivery pipelines to developers. Any time there is a good amount of learning to take place, dissatisfaction and discouragement can come into play.Â

For example if your organization had to support Spotifyâs Agile Methodology, engineers would regularly change team makeup to deliver features. Because of this, learnability of new or different methodologies team-to-team. Having a concise area of information that is prudent for developers to understand the relationships and items needed to be successful throughout the SDLC from idea to production would be key. From a platform engineering ethos, if you are able to delight your internal customers, they will in turn organically strive to delight your external customers. This was top of mind for many who we had conversations with at BackstageCon.Â Â Â

BackstageCon Learnings

Earlier last week we have done analysis on the official talks last year and this year at BackstageCon. The talks were certainly impressive though the conversations that we had outside the official talks e.g âthe hallway talksâ were indicative that many organizations are still early on the evolving developer experience journey.Â Â

IDP Journeys are Still Early

We had the great opportunities to talk to dozens of folks who attended the BackstageCon. This year, BackstageCon was a co-located event which had a different format at KubeCon. Since all Co-Located events were covered under the same price, users had a day filled with exploring different Co-Located events. This allowed for more cross pollination between different events and learnings from individuals at any level.Â

Below, our very own Himanshu Mishra in his talk about what Backstage offers above and beyond two pillar features that users starting their IDP journey would focus on, Software Catalogs and Software Templates.Â Â

Throughout the day at the Harness Booth, we were able to learn from many individuals who are wanting to further developer experience.Â

From The Harness Booth

We had lots of great conversations with those interested in furthering developer experience. From those learning about Backstage for the first time at the event to those implementing an IDP at scale at their organization. A common theme across individuals at all levels is that developer experience can always be improved.Â
Â

At BackstageCon we were pleased to show some of the latest features of Harness IDP alongside the Harness Platform to solve these very problems.

Harness IDP Furthering Your Backstage/IDP Journey

Harness continues to invest into our IDP platform which is based off of Backstage to allow for you to further your IDP journey. As a SaaS platform, there is no need to worry about the infrastructure needed to run Backstage.Â

Coinciding with BackstageCon this year, we released our Scorecard Feature inside Harness IDP to provide a quantifiable measure of software maturity.Â

Taking a look at Harness IDP would be a prudent move in improving developer experience.Â

Start Your DX Journey with Harness IDP

Developer Experience and effectiveness is a journey. Harness IDP and the Harness Platform overall can help reduce toil. Feel free to explore how the Harness Internal Developer Portal (IDP) can enhance your developer experience and help you with your Software Delivery Life Cycle, sign up for a demo today!

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/introducing-sto-developer-certification>

Start or Expand Your DevSecOps Education Journey - Introducing STO Developer Certification

As systems grow more distributed, complex, and critical to our daily lives the fog of development starts to set in; no one person has complete end to end visibility into the entire workings of a system. To combat the fog of development, dissemination of expertise across your pipelines is crucial. Security in an application and infrastructure context is a good model of this with the DevSecOps movement.Â

Software development teams are becoming more tasked with shifting left requirements to produce hygienic software. In the real world, software ages like milk and not like wine, so what was hygienic today might not be hygienic tomorrow. Your CI/CD pipelines are conduits of change and are excellent spots to disseminate expertise and ensure compliance and standards to security posture.Â

Harnessâs Security Testing Orchestration or STO module is purpose built for your pipelines by orchestrating and prioritizing results from a multitude of scanning tools. Most organizations will have more than one scanning tool because tools can be granular or vertical in focus around a few pillars such as intent, language, and distribution. Because of the complexity of modern systems, everyone involved with the development of these systems should take a stake in helping secure these systems.Â

Security, Everyoneâs Responsibility

At Harness, we view security as an important skill to have, we are offering our STO Developer Certification for free so everyone can up-level their DevSecOps skills. Taking a look at our study guide, will provide a great foundation around application vulnerability management.Â

Because so many components are in modern software today, keeping up with the bill of materials / how components age can be tricky. Harness STO can help you identify and prioritize issues that do need to be addressed. Having Harness STO as part of your pipeline is a prudent capability and being certified in Harness STO is a great skill to have.

Study and Sign Up Today

Getting certified at the developer level on Harness Security Testing Orchestration is a great milestone in your DevSecOps journey. Register for the exam from the Harness Developer Hub once you feel comfortable taking the exam.Â

-Harness Product Education Engineering Team

â

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/retiring-harness-cd-community-edition-in-favor-of-gitness>

Retiring Harness CD Community Edition in favor of Gitness

We launched the Harness CD Community Edition as a free, source-available edition of the Harness CD & GitOps module in January 2022. Our intent was to help new users of Harness CD & GitOps learn and use the product in a lightweight (as measured by low resource utilization) and self-managed (as opposed to our SaaS edition) form factor. We received excellent feedback and were able to deliver many of the goals we started out with. However, the true promise of a simple CD pipeline is now better delivered by Gitness, the open source software development platform that we launched at Unscripted in September 2023. As a result, we are retiring the Harness CD Community Edition in favor of Gitness. This means that the currently available release will become the last release and no new releases will be made in the future. Additionally, the underlying source code repositories that were licensed under PolyForm source available licensing will become private.

We recommend all Harness CD Community Edition users download and use Gitness for their day-to-day CD needs. Given that Gitness also includes a git-compliant source code management system combined with the power of Drone pipelines, we are confident that this will be a significantly positive outcome for our user community.

What is Gitness?

Gitness is a new open source Git solution with the breadth and depth of features to empower development teams to write and deliver code with safety and speed. It's a one-stop-shop where you can host your git repositories and automated CI/CD pipelines seamlessly. Say goodbye to juggling multiple tools, and hello to streamlined collaboration, and a smoother development workflow. With this announcement, Harness is doubling down on making Gitness a reliable platform for developers.

â

For more detail, you can review the following blog posts:

- Gitness: Your Ultimate Open Source Development Platform
- Harness Releases Gitness â Open Source Git Platform

Using Gitness for Basic CD

As the Deploying to Kubernetes with Gitness blog highlights, Gitness pipelines are a great way to get started with your basic CD journey. Every step in the Gitness pipeline runs on its own isolated container instance. And steps can be configured to perform

application builds, create docker images as well as push the images to a Docker registry. Similarly, we can also use these steps to launch Kubernetes deployment commands such as kubectl or helm install. This ability forms the foundation of a basic CD pipeline where changes to code will not only trigger a build that generates a container image as an artifact but also automatically deploy the generated artifact onto the deployment target of your choice (which in the case of the above blog was Kubernetes).

Upgrading from Gitness to Harness CD & GitOps

Harness CD & GitOps is the world's most advanced CD platform. It is purpose-built for the world's most demanding enterprises, enabling support to deploy any app to any infrastructure platform (Kubernetes, Serverless Functions, Traditional VMs), advanced deployment strategies (Canary, Blue-Green) with powerful pipelines to orchestrate critical business requirements including governance, intelligent deployment verification with auto rollback of failed deployments, rich integration ecosystem for developer workflows, and much more. You can try Harness CD & GitOps for free today on the Harness SaaS Platform. However, if you need a self-managed yet lightweight option, then Gitness is the recommended way to get started.

Gitness has the ability to upgrade to Harness SaaS Platform where the Gitness pipelines get automatically converted to Harness CI pipelines. The addition of a new Harness CD & GitOps stage to this pipeline is the next step. This will add the advanced CD & GitOps capabilities to this pipeline where the power of Harness delegates, external secrets management, audit trails and pipeline governance come to light.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/harness-doubles-down-on-gitops>

Harness Doubles Down on GitOps

Harness Doubles Down on GitOps, Embracing Both Argo and Flux

â

We're proud to be at KubeCon today, showcasing our heightened commitment to GitOps by integrating support for Flux alongside existing support for Argo. This strategic enhancement positions Harness as a pioneer in the GitOps sphere, offering unparalleled choice and flexibility to its users. We are also demonstrating support for Argo Rollouts for the first time in public!Â

â

GitOps, a modern approach to software delivery, leverages git as the source of truth for declarative infrastructure and applications. Within this realm, Argo and Flux are recognized as prominent GitOps reconcilers, facilitating the automation and consistency of deployments.

â

GitOps has been a revolutionary stride for developers, yet the necessity for security and compliance checks has often been a stumbling block for many organizations. Our Continuous Delivery and GitOps module effectively bridges this gap, ensuring developers enjoy the GitOps experience they desire, coupled with the indispensable compliance checks.

â

The newly unveiled Flux support empowers users with a unified dashboard to monitor and manage multiple Flux controllers. This is a monumental enhancement, especially considering the absence of a graphical user interface in Flux. Harnessâs dashboard will deliver a seamless and intuitive experience, streamlining GitOps operations and rendering them more user-friendly.

â

Harness's inclusion of both Argo and Flux epitomizes its dedication to availing the finest tools to developers based on their unique needs. The GitOps community has traditionally been polarized between Argo and Flux enthusiasts. However, by supporting both, Harness eradicates the need for organizations to compromise or be tethered to a single tool.

Argo Rollouts

Additionally, Harness is amplifying its robust ArgoCD integration by leveraging Argo's Rollouts capability, a feature added to the open-source project earlier this year, which allows for more granular control over deployments, ensuring smoother and more reliable software delivery.

Harness's continuous delivery pipelines for both Argo and Flux will automate a plethora of GitOps interactions. From initiating a rollback due to a problematic deployment to automatically promoting a new version to the subsequent testing environment upon successful test completion, Harness guarantees streamlined and automated operations.

â

âHarness acknowledges the diverse needs within the GitOps community. By supporting both Argo and Flux, we're not merely offering a tool but a choice. Our objective is to empower developers with flexibility and ensure they have the best tools at their disposal for their unique workflows," said Mark Ramm, Product Management Lead for GitOps at Harness.

This announcement, delivered at KubeCon in Chicago, underscores Harness's dedication to innovation and its unyielding commitment to catering to the evolving needs of the developer community.

â

To learn more about GitOps with Harness, check out our tutorial for Kustomize deployments with ArgoCD; it includes instructions for how to sign-up for a free Harness account.

â

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/argocon-2023-gitops-achieves-escape-velocity>

ArgoCon 2023 Recap: GitOps Achieves Escape Velocity

Introduction

As the curtains closed on ArgoCon North America 2023 this evening (November 6th), the Chicago air buzzed with the collective energy of Kubernetes and Argo aficionados. The event, held in conjunction with KubeCon + CloudNativeCon, brought together the community to deepen their understanding of the Argo Project and its constellation of continuous delivery tools â Argo CD, Workflows, Rollouts, and Events â that are enhancing Kubernetes' deployment and management landscape.

This year at ArgoCon North America 2023, we at Harness took great pride in our role as a platinum sponsor, providing more than just backing â we dove headfirst into a wealth of discussions and workshops, engaging with the community at every turn. Our dedication to the Argo Project and the wider cloud-native ecosystem was on full display as we shared our insights, learned from our peers, and took valuable feedback on our contributions to the community â from the Litmus Chaos initiative to the OpenTofu project.

For those who couldn't join us or wish to relive the experience, this blog will serve as your portal to the event's highlights. We'll walk you through the keynotes, engaging talks, enlightening discussions, and more, capturing the essence of ArgoCon right here in our recap.

About the Argo Project

At ArgoCon North America 2023, the central theme revolved around the Argo Project and its pragmatic approach to enhancing Kubernetes. Argo, with its suite of tools including Argo CD, Workflows, Rollouts, and Events, is fundamentally about addressing the day-to-day challenges encountered by those managing Kubernetes applications.

These tools arenât just technical advancements; they're practical solutions. Argo CD simplifies the deployment process, Workflows handle complex jobs, Rollouts introduce safer updating mechanisms, and Events trigger actions based on specific scenarios. Together, they create a more manageable and resilient Kubernetes ecosystem.

The Argo Project's enhancement of Kubernetes APIs means less complexity and more automation for teams deploying cloud native applications. It translates into faster deployment times, reduced risk of errors, and a more streamlined workflow for operations teams.

With the Argo Project's impact outlined, let's pivot to the highlights and learnings from ArgoCon, where real-world applications and success stories took the spotlight, offering insights into the ways these tools are being leveraged to shape the future of Kubernetes deployments.

Conference Highlights

This year's topics coincided with general industry buzz. Many talks focused on how Argo can manage the complexity of deploying desired state applications at scale without compromising security and governance.

AI

Artificial intelligence has been on top of technology headlines over the past year, and several talks focused on Argo tools applied to AI model training and observability. A key challenge is managing the immense amounts of parallel data processing inherent in AI applications. Teams at organizations like Intuit have begun pairing Argo Rollout capabilities to include anomaly detection as Kubernetes clusters scale, and Argo Workflows are increasingly used to manage the end to end training and evaluation of large language models like Llama.

Securing the Development Lifecycle

As software lifecycles become increasingly intricate, security concerns have intensified, particularly regarding the integrity of the technology stack. This has led CISO teams in both the private and public sectors to focus on mitigating vulnerabilities within the supply chain. In response, Argo developers and users have identified several security trends.

Firstly, the community is recognizing the benefits of the GitOps paradigm, which, by its nature, provides an immutable and declarative setup that acts as a single source of truth, aiding in maintaining a consistent audit trail for compliance purposes. Secondly, Argo is developing specific resources aimed at enhancing the security of multi-tenant applications.

Moreover, the Argo community is engaging in discussions to define what constitutes a well-architected framework, focusing on establishing a single source of truth that can help in the early detection and prevention of deployment misconfigurations, much like how cloud adoption led to the creation of standardized architectures and practices.

Managing Scale

Argo tools are at their core deployment tools, and an immense amount of community focus is being spent on how applications can be scaled to hundreds, thousands, and millions of users and environments. It's clear that open source infrastructure as code (IAC) tools like Terraform aren't going anywhere, but there's been a pivot from pure automation to optimization and managing application complexity (for example, multi-cluster deployments). Major cloud providers like AWS have adopted and shared new patterns for designing and bootstrapping Kubernetes applications that can deploy at massive scale.

The Harness platform is designed to manage scaling infrastructure and business processes in ways with which native Argo currently struggles. This includes integrating GitOps deployments with existing project management and service desk tools. Harness also provides a single orchestration layer to manage horizontal and vertical scaling of Argo applications, along with automating promotion between environments.

Closing Remarks

ArgoCon 2023 has cast a spotlight on AI, security, and the need for scalable technology solutions. Argo tools have proven essential for modern software development and delivery needs. However, as deployment pipelines and strategies grow more sophisticated, the simplicity of ArgoCD might be challenged.

Harness Continuous Delivery not only integrates seamlessly with GitOps engines like Argo CD or Flux but also simplifies the progression of software artifacts through development, QA, UAT, and production stages. Harness provides a framework for multiple approval modes, ensuring that each transition can be thoroughly vetted according to the organization's governance needs. This level of control and oversight is crucial for maintaining quality and security as software moves closer to end-users.

As we reflect on the lessons from ArgoCon 2023, we encourage the tech community to consider how these tools can be applied to create secure, resilient, and scalable systems. To learn more and to join the discussion on leveraging these tools for your infrastructure needs, please visit Harness Continuous Delivery.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

[Documentation](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[Case studies](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[We want to hear from you](#)

Enjoyed reading this blog post or have questions or feedback?
Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/unleashing-the-power-of-generative-ai-transforming-the-inner-and-outer-loop-dynamics-in-development>

Unleashing the Power of Generative AI: Transforming the Inner and Outer Loop Dynamics in Development

Software Development in the modern era is akin to navigating an inner loop-outer loop framework, each with distinct characteristics and challenges.

Inner Loop: This encompasses high-value creative tasks, such as developing secure code, solving problems, designing, reviewing, and coaching.

Outer Loop: In contrast, involves repetitive tasks that divert attention from the inner loop's high-value work. These include building code, deploying artifacts, bookkeeping, administrative work, and unproductive meetings.

The advent of Generative AI has revolutionized the development landscape, initially serving as a co-pilot for developers, enhancing the inner loop. We're witnessing a shift from spending four days to write and one day to test code, to a scenario where writing code takes only a day but the outer loop work (building, testing) consumes four days.

GitHub Copilot, with 1 million paid users and 37,000 organizations, has emerged as a significant player in this transformation. It accelerates coding, with studies indicating a 55% increase in developer productivity. However, the growing volume of code presents a challenge—outer loop activities become the bottleneck.

In a study of 73 enterprises, an average of 10,193 developer hours per year per enterprise was spent on running and testing builds, with an additional 942 hours on manual rollbacks. The current state of outer loop tools and processes is inadequate for the evolving development landscape.

Addressing the Challenge: Next-Gen Intelligent Outer Loop Tooling

The outer loop tooling and processes has to be able to operate with the same intelligence and speed as the AI enabled inner loop.

Faster Build System

A super-fast build system is imperative to keep up with the volume of new code. The system should intelligently test only the code areas that have changed or pose a higher quality risk. For example, it should rigorously test generative code or examine certain error-prone areas of the code. Additionally, the system should leverage AI capabilities to triage build failures.

Hassle Free Deployments with Smart Feature Flags

To reduce developer toil, developers need the ability to leverage templated pipelines for swiftly deploying their code. They should be capable of constructing new pipelines using an AI co-pilot. Once the code is deployed, AI can be employed to verify the deployment and automatically roll back if necessary. Smart feature flags should be utilized to expedite code deployment to production. AI can also be leveraged to automatically clean up stale feature flags, thereby reducing the cognitive load on the developer.

Integrated Security and Governance

Research indicates that Generative code can generate a high volume of insecure code, with studies showing that Copilot generates vulnerable code 40% of the time. Although this number is expected to decrease over time, the need for security testing increases significantly in this new area. Many organizations are currently grappling with fixing security vulnerabilities. With GenAI, the volume of code requiring scanning and fixing only increases. Integrating security into the pipeline, deduplicating vulnerabilities, explaining vulnerabilities, and providing automatic fixes or fixing the code are areas where AI can play a significant role. Left-shifting security so that issues are caught pre-commit is also something GenAI can be particularly helpful with.

Continuous Resilience

As more Generative code is deployed, the criticality of service and system resilience becomes apparent. The exponential pace of innovation demands more Chaos Testing in the pre-production phase. Identifying and triaging reliability issues are areas where AI can play a substantial role in this new world.

Improving Developer Experience

As more developers embrace GenAI, there is a significant opportunity to reduce developer toil. While GenAI usage in inner loop

activities will undoubtedly help developers alleviate some pain, there is still a considerable need to reduce tribal knowledge concerning people, processes, and services. Creating a catalog of metadata around different services, their ownership, and all other related metadata can drastically reduce the time developers spend in unnecessary meetings. Surfacing outer loop activities in the inner loop can assist developers in reducing cognitive load. Providing golden paths for repeated outer loop activities is also of utmost importance.

Data-Driven Cost and Process Optimization

As developer productivity is unlocked and more GenAI code is deployed, increased cloud costs become a major concern for most companies. Tracking cloud costs across different dimensions and identifying opportunities to reduce idle and wasteful spending is crucial. GenAI can analyze data, pinpoint areas of opportunity where spending can be reduced, thereby reducing the cognitive load on the team responsible for tracking and managing costs.

Another area is the increased need for engineering intelligence. There is need to continuously optimize workforce, process and tooling for both inner and outer loop. As more GenAI gets infused into the SDLC, it becomes important to understand what is working and where there are gaps or opportunities. Understanding developer productivity and quantifying developer experience also becomes important. Leveraging frameworks like DORA and SPACE, and surfacing metrics and insights across people, process and tooling becomes an imperative.Â

â

Connecting the Inner Loop and Outer Loop ToolingÂ

However, just optimizing the outer loop or inner loop separately is not enough. We need the inner loop and outer loop toolingÂ to be connected and seamlessly share intelligence. For example we may want the build system to do more testing if the code has been generated by AI. Conversely we may want to learn from past build and developer failures and warn developers when they are writing new code.Â

Conclusion

As GenAI is infused into the inner loop today, there is a need to thoroughly examine the outer loop activities and understand how it can support this new world. A robust next-gen platform that assists developers in leveraging AI capabilities end-to-end is necessary. Having one platform that spans both the inner loop and outer loop becomes imperative to ensure that bottlenecks are minimized, allowing developers to focus on highly creative work without worrying about repetitive outer loop tasks.

For developers seeking to be at the forefront of innovation, now is the time to integrate Harness AIDA into your workflow. If you are interested in learning more about how we are infusing AI capabilities across the SDLC, check out our blogs in our AI Feature series to learn more!

- Introducing Dashboard Intelligence Powered by AIDA
- Introducing CD Error Analyzer
- Introducing Harness Support Powered by AIDA
- Introducing Policy as Code Assistant Powered by AIDA
- Introducing AI Infused Feature Flag Onboarding

â

References

<https://arxiv.org/pdf/2204.04741.pdf>

<https://www.zdnet.com/article/microsoft-has-over-a-million-paying-github-copilot-users-ceo-nadella/>

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/december-2023-product-updates>

December 2023 Product Updates

December 2023 Product Updates

At Harness, we have been hard at work so you can delight your customers without facing software delivery toil. Here is what has been changing in the previous month across Harness Products.

Continuous Delivery & GitOps

Full Continuous Delivery & GitOps Release Notes

Chaos Engineering

Full Chaos Engineering Release Notes

Internal Developer Portal (IDP)

Full Internal Developer Portal Notes

Software Engineering Insights

Continuous Error Tracking

Full Continuous Error Tracking Release Notes

Self-Managed Enterprise Edition

Early Access

â

Continuous Delivery & GitOps

Continuous Integration

Chaos Engineering

Software Engineering Insights (SEI)

Continue the Journey

- Learn more hands-on with our increasing list of Tutorials.
- Further your knowledge with one of our Certifications.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/harness-sei-churn-rate-insights>

Unlocking Efficiency: Exploring Churn Rate with Harness Software Engineering Insights (SEI)

Introduction

Improving the effectiveness and productivity of developers is crucial for the success of an engineering team. To achieve this, the team should adopt a metric-driven approach to identify the root causes that scales down the team's productivity. Out of the many metrics that engineering leaders can use to understand an engineering team's productivity is Churn Rate. In this article, you'll learn how high churn rates can impact the team and how Harness SEI can help lower churn rates in your organization.

What is Churn Rate?

In the context of software development, churn rate refers to the amount of work that is added or removed from a sprint backlog during a sprint. It measures the scope change and provides insights into the volatility of the sprint backlog.

- A high churn rate indicates a lot of changes in the sprint backlog, which may result in delays in completing the sprint, and may also indicate that the requirements are not well-defined.Â
- A low churn rate suggests that the sprint backlog is stable and the requirements are well-defined.

Why Churn Rate Matters?

Unveiling Sprint Volatility:

Churn Rate acts as a window into the dynamic nature of your sprint backlog. By understanding the scope changes during a sprint, teams can identify and address volatility, fostering a more stable and predictable development environment.

Optimizing Workflows:

SEI's Churn Rate empowers teams to optimize workflows by pinpointing areas of change mid-sprint. This insight allows for targeted

strategies, ensuring that the development team can adapt swiftly and stay on course despite evolving project requirements.

Enhancing Collaboration:

With Churn Rate, collaboration between product managers and engineers reaches new heights. The metric provides a common ground for discussions, enabling both teams to make informed decisions and align efforts seamlessly.

The Formula Unveiled:

Churn Rate = (Points added mid-sprint + Points removed mid-sprint + Positive difference of changes in planned issues) / Points committed at the start of the sprint

- **Points added mid-sprint:** The sum of story points for items added during the sprint.
- **Points removed mid-sprint:** Identifies the reduction in story points due to the removal of items during the sprint.
- **Positive difference in planned issues:** Reflects the positive changes in story points for planned issues during the sprint.

â

â

Incorporating Churn Rate into SEI

SEI seamlessly integrates Churn Rate into its comprehensive suite of metrics, leveraging the power of the Trellis Framework. As part of the 40+ third-party integrations, Churn Rate ensures that your software factory's performance is not just measured but optimized, setting the stage for unparalleled efficiency.

Unlocking Potential with SEI's Churn Rate

In conclusion, Churn Rate emerges as a catalyst for transformative change in software development. SEI's commitment to providing actionable insights and workflow automation reaches new heights with this innovative metric. By understanding and harnessing Churn Rate, software delivery teams can not only increase productivity but also build a foundation for sustained excellence.

In a world where adaptability is key, SEI's Churn Rate becomes the compass that guides your software engineering ship through the ever-changing seas of development, ensuring you not only navigate challenges but thrive in the face of change. Experience the power of Churn Rate with Software Engineering Insights â where productivity meets precision.Â

Engineering teams can leverage Harness Software Engineering Insights to first baseline the people, processes and tooling bottlenecks and then drive a continuous improvement process. To learn more, schedule a demo with our experts.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?
Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/announcing-the-harness-srm-backstage-plugin>

Announcing the Harness SRM Backstage Plugin

We are excited to introduce the latest addition to our suite of Open Source Backstage Plugins - the Harness SRM Backstage Plugin. This plugin is designed to seamlessly integrate with your Backstage Instance as well as with Harness IDP to help with development team's workflow, ensuring that Service Level Objectives (SLOs) and error budgets are not just a metric but a part of your daily development practice.

Why Focus on SLOs?

Good Site Reliability Engineering (SRE) practices hinge on the continuous monitoring and adherence to SLOs. These objectives are pivotal in maintaining the reliability and performance of services. However, in the fast-paced world of software development, it's often challenging for developers to continually engage with separate observability tools. This is where our plugin bridges the gap.

Streamlining Observability

The Harness SRM Backstage Plugin is more than just a tool; it's a solution to a common oversight in the development process. By aggregating SLO data from Harness Service Reliability Management Module, this plugin brings critical insights directly to the developers' dashboard. This integration means that your team no longer needs to switch contexts or platforms to monitor their SLOs.

High Availability of Adequate Information

With this plugin, information about SLOs is not tucked away in a separate tool but is readily available in the Developer Portal. This accessibility ensures that your team is always aware of the current status of your services, leading to quicker responses and resolution if SLOs are at risk of being breached.

Empowering Developers

By making SLO data readily available within the familiar environment of the Developer Portal, the Harness SRM Backstage Plugin empowers developers to take proactive steps in maintaining and improving service reliability. This approach not only enhances individual productivity but also fosters a culture of accountability and ownership within the team.

Conclusion

In summary, the Harness SRM Backstage Plugin will help with how development teams interact with and respond to SLOs. By integrating critical data into the daily workflow, it ensures that adhering to good SRE practices is not an additional task but a seamless part of the development process.

Signup today for SRM Module and start using the Plugin to experience the difference this plugin makes in your team's efficiency and the reliability of your services.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/harness-product-modules/cloud-cost-management>

Source URL: <https://www.harness.io/blog/bitbucket-servers-sunset>

Bitbucket Server's sunset on February 15th, 2024: A Evolving to streamlined DevSecOps from a disjointed past

Tick-tock, Bitbucket Server users. As Atlassian shifts its focus to cloud solutions, discontinuing Bitbucket Server, it's crunch time to find a reliable alternative. Continuing with Bitbucket Server beyond this point poses an increased risk, associated with a lack of active support, unresolved bugs, and unaddressed security vulnerabilities. So, what does the future look like for BitBucket ServerÂ users?

Bitbucket Server users are left with multiple options:

- Keep using Bitbucket Server, enduring increased risk due to lack of support, bugs, and security fixes.
- Migrate to Bitbucket Data Center or Cloud, missing key capabilities needed for reliable software delivery, such as Feature Flags.
- Switch to a different vendor. In that case, we suggest you consider Harness, our AI-powered Software Delivery Platform

But why a software delivery platform, you ask?Â

Code is king in the realm of software delivery. As we navigate the ever-shifting digital landscape, a robust, cohesive platform for managing, reviewing, building, testing, and deploying code is critical. Clinging to a patchwork of standalone tools not only fragments your experience but also devours precious time on context-switching instead of coding, chipping away at productivity. Imagine the complexity of managing CI, CD, and feature flags separately, with the added burden of mastering these tools, users and permissions management, handling multiple procurement processes, and building and maintaining system integrations. That's a lot of redundant work.

Switching to a consolidated DevSecOps platform changes the game. It streamlines operations, slashes maintenance hours, and boosts

security, allowing developers to laser-focus on crafting code rather than wrestling with a jumbled toolchain.

Why Harness is your ultimate devSecOps platform

In the competitive DevOps landscape, platform engineers are on the lookout for tools that can scale with their needs. At Harness, we're committed to delivering an excellent developer experience, facilitating the sharing of best practices, and ensuring effective governance without being burdensome.

Our approach is validated; Harness was ranked first for the Platform Engineering use case in the Gartner 2023 Critical Capabilities for DevOps Platforms report (more on this in our blog why platform engineers trust harness).

In the words of Ratna Devarapalli, Director of IT - Architecture, Platform Engineering & DevOps at United Airlines:

To read more about how Harness helped United Airlines Accelerates Deployments by 75% With Harness visit [United Airlines Case Study](#).

So what is it that makes Harness shine? Let's dive in.â

Next generation CI/CDâ

Harness revolutionizes DevOps workflows, offering unparalleled speed and intelligence in software delivery:

Fastest CIâ

Harness CI is up to 4x faster than competing CI tools. This unparalleled speed amplifies developer efficiency as well as translates to significant infrastructure cost savings, due to reduced build time and competitive cloud pricing. Key features contributing to this speed include:

- **Test Intelligence:** Harness CI's ML-based Test Intelligence feature intelligently identifies and runs only the tests impacted by recent code changes, reducing test cycle times by as much as 80%.
- **Cache Intelligence:** With the push of a button, Cache Intelligence takes charge of caching common dependencies and Docker layers, speeding up build times and simplifying the caching process.
- **Isolated cloud build machines:** with Harness's no-share infrastructure Harness ensures performance without the noisy neighbors to slow you down.

Worldâs most advanced CDâ

- **Smart deployment strategies:** Harness's CD capabilities stand out with out-of-the-box deployment strategies like Blue/Green and Canary, so you don't need to waste time scripting complex logic. Paired with AI-assisted release verification, and automatic rollbacks, deployments are quick and fail-proof.
- **Seamless multi-cloud deployments with proactive infrastructure management:** Harness enables effortless deployments across major common cloud providers including AWS, GCP, and Azure, as well as seamless integration with Terraform and other similar tools for cost-effective, on-demand environment provisioning, enhancing deployment flexibility and reducing infrastructure cost.
- **GitOps excellence:** Harness champions comprehensive GitOps support with native integrations for Argo and Flux, and a centralized control plane for visibility across clusters

Smart Feature Flags

Harness simplifies progressive delivery with Smart Feature Flags, enabling uninterrupted deployments and selective feature releases. This functionality is enhanced by automatic detection and optional automatic removal of stale flags, helping to prevent feature flag debt from accumulating.

Developer first experience

A recent LinkedIn survey we conducted sheds light on a pressing challenge in the software development industry: a staggering 80% of developers reported spending less than half their working hours on actual coding. This eye-opening figure signals a clear call to action for reshaping our development ecosystems. At Harness, we're tackling this challenge head-on. Harness is not just about improving efficiency; it's about enhancing the joy of creation, ensuring that developers can immerse themselves in their craft with minimal distractions. With Harness, the goal is clear: more coding, less overhead.

Here are a few examples of how Harness streamlines workflows to put coding time back into the hands of developers.

- **Streamlined pipeline design** - Harness users can easily author and quickly understand pipeline with our intuitive visual drag-and-drop editor, and an 'as-code' editor is also available with auto complete and schema validation, making editing pipeline in yaml a breeze. Users can easily save their pipelines, stages and steps as templates for future use, cutting down on repetitive tasks.
- **Guiding development with 'golden path' templates** - Harness encourages best practices from the start, using shared templates, easily discoverable from within pipelines, to promote collaboration and standardize excellence across teams.
- **AI-driven diagnostics** - Harness empowers developers to troubleshoot and resolve build and deployment failures swiftly and effortlessly, using AI-powered root cause analysis and remediation suggestions.â
- **Bridging development and security:** Security is crucial, but so is agility. With Harness, developers can easily request security

exemptions, increasing collaboration between developers and the security team.

Security & Governance: The Harness Assurance

Harness is redefining governance and compliance, transforming them into seamless aspects of the CI/CD process, so you can shift-left security without shifting-left the workload.

- **Proactive security scans from the get-go:** with Harness you can shift-left security with security scanning seamless integration into your CI/CD pipelines. Our Security Testing Orchestration helps your developers prioritize and squash vulnerabilities swiftly, ensuring that security is a constant, not an afterthought.
- **Empowered yet governed:** Harness embodies OPA-based Policies, providing a framework for consistent governance while maintaining the flexibility for teams to tailor their pipelines. This approach ensures that security checks, like mandatory scans, are embedded in your software delivery process.
- **Uncompromising governance and compliance:** Harness offers an array of features to keep your pipelines not only efficient but also compliant and secure. This includes easy-to-use Templates, OPA-based policies, robust Secrets Management, and granular Role-Based Access Control (RBAC), all supplemented with Approval workflows, Notifications, and comprehensive Audit Trails. It's the governance you need, tailor-made for your organization's unique demands.
- **SLSA L2 Compliance Without Complications:** Leverage our Software Supply Chain Assurance to achieve SLSA L2 Compliance with ease. Harness automatically crafts SBOMs and attestations, safeguarding the integrity of your artifacts every step of the way.

Crystal-Clear Visibility with Harness

Our DevSecOps platform offers deep visibility across your development pipeline, with detailed insights ranging from DORA metrics to test performance and security assessments. This ensures compliance with regulatory standards and a clear understanding of your processes. Customizable dashboards give you the power to focus on what matters most to you. This level of clarity allows you to identify and resolve bottlenecks quickly, fostering rapid and high-quality software delivery.

Harness Code Repository is here

Harness has already distinguished itself in the realm of DevSecOps, offering stellar CI/CD. The natural progression of this journey was introducing of Source Code Management (SCM).

Revealed at our recent Unscripted event, Harness Code Repository is adding enterprise-grade code management into our platform. Built on the open-source Gitness, this premium module is crafted to simplify and enhance the workflows of development teams.

Harness Code Repository is redefining code collaboration and governance. It offers seamless collaborative reviews and advanced governance features, including branch protection and policy enforcement through OPA, to ensure code complies with predefined organizational standards and best practices. We're also breaking new ground with AI-driven features, beginning with Semantic Search AI. This cutting-edge functionality transforms codebase navigation, allowing developers to use natural language for their queries. It streamlines the onboarding process and simplifies troubleshooting, marking a significant leap in developer efficiency.

Harness Code is set to redefine the integration of code management within the CI/CD pipeline, offering a streamlined experience that bridges coding with deployment. This recent addition promises to enrich developer workflows, bolster productivity, and diminish operational complexity, reinforcing Harness's commitment to an efficient, developer-focused ecosystem.

Conclusion

As we approach the final days of Atlassian Server, it's more than a farewell; it's an invitation to evolve. Harness stands ready to guide you through this transition with a platform that's not just a step up from Bitbucket Server but a leap into the future of DevSecOps. Join us on this journey and redefine what's possible in your software delivery process.

Ready for the next step? sign up to get started, or request a demo to learn more.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/securing-images-cosign-opa>

Securing CI/CD Images with Cosign and OPA

With the growing adoption of containers in modern development, ensuring the integrity of container images has become central to application deployment strategies. Rapid deployment offers agility, but it also presents security challenges. Container image signing addresses these challenges, allowing engineering teams to verify that the deployed images are authentic and unchanged.

However, an authentic image is just one piece of the puzzle. Making sure these images meet organizational standards is important, and policy engine tools are crucial for that. By establishing and enforcing clear guidelines, these tools pave the way for a secure and streamlined deployment workflow.

In essence, container image signing involves adding a digital stamp to an image, affirming its authenticity. This digital assurance guarantees that the image is unchanged from creation to deployment. In this blog, I'll explain how to sign container images for Kubernetes using Cosign and the Open Policy Agent. I will also share a tutorial that demonstrates these concepts.

If you prefer a video tutorial, please watch here:

Choosing the Right Tools for Image Signing and Verification

Choosing the right tools for container image signing and verification is important in CI/CD pipeline security. Let's walk through the available options to find the one that best meets your needs for securing container images.

Image Signing and Verification Tools:

Notary v1, previously known as Docker Content Trust, uses The Update Framework (TUF) but has some issues with signature portability and storage. Though it established a solid foundation in image signing, it lacks some of the enhanced security features found in more recent tools.

Grafeas: While Grafeas offers a comprehensive solution for the software development lifecycle, it is not designed for public or open-source software (OSS) image verification. It's better suited for first-party integration, particularly with Google Kubernetes Engine (GKE).

Notary v2: The evolution to Notary v2 brought improvements in signature portability and integration with third-party key management solutions. However, it does not provide a certificate authority, leaving public key discovery for open-source image verification as an unresolved issue.

The Update Framework (TUF): TUF is a framework, not a tool, designed to enhance the security of software update systems. It focuses on resilience against key compromises and attacks, employing verifiable records to verify the authenticity of update files. TUF's flexibility and integration ease make it a foundational element in securing software updates, though it's not a direct image signing tool like the others.

Cosign: In this context, Cosign from the Sigstore project offers a compelling solution. Its simplicity, registry compatibility, and effective link between images and their signatures provide a user-friendly and versatile approach. The integration of Fulcio for certificate management and Rekor for secure logging enhances Cosign's appeal, making it particularly suitable for modern development environments that prioritize security and agility.

Cosign's strength lies in its verification process, which is vital in the CI/CD pipeline. When integrated with policy engines like the Open Policy Agent (OPA), Cosign ensures not only the authenticity of container images but also their compliance with organizational standards. Additionally, Cosign provides enhanced support for SPIFFE, GitHub Actions, or service account identities, and it includes warnings for signing OCI images by tag, highlighting the risks associated with tag mutability.

Policy Enforcement Options:

Open Policy Agent (OPA): OPA's strength lies in its ability to define fine-grained policies as code, offering granular control over policy enforcement. Integrated seamlessly with Kubernetes, it enforces policies in real-time, preventing unauthorized image deployments.

Kubernetes Admission Controllers: These controllers are a native part of the Kubernetes ecosystem, making them a straightforward choice for Kubernetes users. They excel at validating and enforcing policies for various Kubernetes resources, including pods, making them scalable for large deployments.

When choosing tools for container image signing and policy enforcement, it's important to balance functionality, scalability, and community support. Among the options discussed, Cosign and OPA (Open Policy Agent) stand out for their effectiveness in securing containerized applications.

Cosign simplifies the process of image signing and verification. It offers a user-friendly approach that caters to developers and security teams alike. With a growing user community and easy integration, it's a practical choice for securing container images.

OPA, on the other hand, excels in policy enforcement within Kubernetes environments. It enables you to define and enforce policies as code, granting you fine-grained control over image deployments.

In the upcoming section, let's delve into architectural diagrams and explore how the combination of Cosign and OPA offers a practical approach to sign and verify container images while enforcing them using a general policy engine.

Architectural Diagrams: Securing Container Images and Deployment

Traditional cryptography uses public-private key pairs for signing and verifying images. However, modern practices, like cosign, prefer keyless signing. In this approach, an OIDC (OpenID Connect) provider is utilized, making the process more streamlined and secure, as it removes the complexities of key management.

The process starts with an architect selecting a trusted public base image as the foundation for their application. Let's dive into two distinct but interconnected flows that demonstrate how container image signing with Cosign and policy enforcement using Open Policy Agent (OPA) play a crucial role in ensuring a secure deployment pipeline.

The process starts with an architect who selects a trusted Public Base Image as the foundation for their application. To ensure the image's integrity, the architect runs a vulnerability scanning process. This step identifies and addresses any security vulnerabilities within the base image.

Next, any unnecessary or unused components are removed from the image, minimizing potential attack vectors. The architect adds the necessary libraries, dependencies, and software packages required for the application to run efficiently. Then, the image undergoes rigorous security testing to ensure that it meets security standards and aligns with organizational policies.

The final step involves image signing with Cosign, a tool specifically designed for image signing. Cosign adds a digital stamp to the image, affirming its authenticity. This signature plays a crucial role in verifying the image's integrity during deployment. The signed image is then pushed to an artifact registry as a new, secure base image.

In the deployment phase, a developer selects a base image for their application, either an **Unsigned Public Base Image** or a **Signed Public Base Image**. If the developer chooses an Unsigned Public Base Image, the deployment process checks the image's signature and evaluates it against predefined policies using Open Policy Agent (OPA). Due to the lack of a valid signature, the image is **Denied** For Deployment.

On the other hand, if the developer opts for a Signed Public Base Image, the same verification and policy enforcement process is applied

using OPA. This time, since the image has a valid signature, it is **Allowed** For Deployment.

Hands-On: Deploying Secure Containers with Cosign and OPA

Having explored the strengths of Cosign and OPA for container image signing, you're now ready to apply these tools in a practical scenario. To get a real-world feel for how these technologies can enhance the security of your Kubernetes deployments, follow this hands-on tutorial. It will guide you through the process of securely deploying container images using the combined capabilities of Cosign and OPA in your Kubernetes cluster. The Harness Software Supply Chain Assurance (SSCA) module addresses the challenges of securing your software supply chain, including image signing and verification. To explore how SSCA can enhance your pipeline security, check it out.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/november-2023-product-updates>

November 2023 Product Updates

November 2023 Product Updates

â

At Harness, we have been hard at work so you can delight your customers without facing software delivery toil. Here is what has been changing in the previous month across Harness Products. Â

Continuous Delivery & GitOps

- PIPELINE_FAILED
- STAGE_FAILED
- STEP_FAILED

The following pipeline events now include the name and pipeline tag:

- PIPELINE_SUCCESS
- STAGE_SUCCESS

â

Full Continuous Delivery & GitOps Release Notes

Continuous Integration

Full Continuous Integration Release Notes

Cloud Cost Management

Service Reliability Management

Full Service Reliability Management Release Notes

Chaos Engineering

Internal Developer Portal (IDP)

IDP has now graduated from Beta into Public Preview. During Unscripted in September, we made a series of announcements. Here are some quick links for your recap.

- Launch Demo in Keynote by Jyoti Bansal and Eric Minick.
- Platform Engineering Demo by Alex Valentine.
- Announcement Blog Post by Himanshu Mishra.

Full Internal Developer Portal Notes

Software Engineering Insights

Continuous Error Tracking

Full Continuous Error Tracking Release Notes

Platform

Here are some important improvements to our platform that should enhance your user experience:

Self-Managed Enterprise Edition

Early Access

Continuous Integration

Chaos Engineering

Software Engineering Insights (SEI): â

Self-Managed Enterprise Edition

Full Early Access Release Notes

Continue the Journey

- Learn more hands-on with our increasing list of Tutorials.
- Further your knowledge with one of our Certifications.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/simplifying-policy-creation-and-management-with-harness-aida>

Simplifying Policy Creation and Management with Harness AIDAâ€¢

Coming Soon â AIDAâ€¢ Policy as Code Assistant

Harness continues to innovate in the realm of Continuous Delivery (CD) by introducing AIDA with Policies capabilities. This blog will dive into why we are developing this new feature and how it will transform your OPA experience.

â

The essence of this innovation is to simplify policy creation, management, and comprehension, which are crucial for maintaining governance and compliance in modern software delivery pipelines.

Understanding Open Policy Agent (OPA) â Simplifying Governance in Action:

For those new to the concept, Open Policy Agent (OPA) is a powerful, general-purpose policy engine that unifies policy enforcement across the stack. To better illustrate its significance, consider a common use case: preventing a software deployment that lacks an essential approval step. OPA allows you to define and enforce policies like this, ensuring that every deployment adheres to your organization's compliance and governance standards. This capability is crucial in modern software delivery to ensure control and compliance.

By integrating OPA into your workflow with Harness, AIDA can assist in writing these governance checks, thus streamlining your delivery process while ensuring adherence to crucial policy requirements.

â

Here are some of the key capabilities and benefits of Harness AIDA with Policies capabilities:

Write OPA Policies with ease:

Learning a new programming language to write policies can be daunting and time-consuming. This is where REGO (the language that powers Open Policy Agent and Harness Policies) has been a game changer - it's a powerful language tailored for evaluating JSON

objects to make decisions based on specified criteria. However, with Harness AIDA, the game is changed yet again. Users are no longer required to dive deep into the syntax of REGO to start drafting policies. AIDA facilitates a seamless experience in getting a working policy up and running in a matter of minutes. This ease of adoption is revolutionary as it lowers the entry barrier for users new to policy creation and management.

Refinement and Build Context-Specific Policies:

A significant advantage of using AIDA is the ability to refine policies to address specific use cases rooted in the Harness platform context. This contextual adaptation makes modifying your Harness Policy a breeze. Over time, as your requirements evolve, your policies can morph into more complex structures through natural language inputs. The beauty of it is, you don't have to be a REGO guru to draft or modify complex policies, making policy management a less daunting task.

â

Policy Summarization and Learning:

Harness believes in fostering a culture of continuous learning and collaboration. With AIDA, new users can effortlessly learn about Harness Policies and obtain summarized details concerning a particular policy. This feature enables users to understand policies drafted by their peers without the necessity of reaching out for explanations. AIDA's ability to comprehend and summarize policy behavior significantly cuts down the learning curve and accelerates team onboarding.

â

â

Enhanced Collaboration:

â

The summarization feature not only accelerates learning but also enhances collaboration among team members. When policies are easily understandable, it fosters a conducive environment for discussions, feedback, and continuous improvement. The seamless collaboration ensured by AIDA can be a significant driver for promoting a culture of transparency and shared understanding within your organization.

â

Harness's AIDA with Policies capabilities is undeniably a giant stride towards simplifying policy management while ensuring robust governance in your software delivery pipeline. By alleviating the need to become a REGO expert and enabling easy understanding and collaboration around policies, AIDA is set to be a significant asset for developers and teams aiming for streamlined, compliant, and efficient software delivery processes.

â

Want to level up your pipeline governance and continuous delivery practices? Â

â

Experience the ease of policy management with Harness AIDA and elevate your Continuous Delivery journey to new heights. Explore more about Harness AIDA with Policies capabilities at harness.io Â and sign up for a sign up for an AIDA demo today!Â

And if you've enjoyed this blog, be sure to explore the other features in our AI Feature Focus series to learn even more about what makes AIDA truly exceptional.

- Introducing Dashboard Intelligence Powered by AIDA
- Introducing CD Error Analyzer
- Introducing Harness Support Powered by AIDA

â

Our Plan to Infuse AI Across the SDLC

Stay informed about our exciting future developments by keeping an eye on our comprehensive product roadmap.

â

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/adventures-at-magnoliajs>

Adventures at MagnoliaJS

Last week I had the great pleasure to speak at a very intimate developer community event in Jackson, Mississippi. The event I spoke at is called MagnoliaJS. MagnoliaJS is ran by two great community ambassadors, Kayla & Richard Sween, as well as a great group of volunteers. This year the event was Halloween themed. There were decorations outside the venue, as well as spooky coffee cups, various Halloween candy scattered about, and even speakers and attendees in costumes.

This year's event I did not plan accordingly lol. You would think that the horror fan would have had his stuff togetherâ|nope. There was a Spirit Halloween nearby but I felt that next year I could plan better and just talk up my idea to the community. Be on the lookout for what Kayla and I have cooking up.

âDay One

Day one of the event was everything you could ask for. The day kicked off with opening remarks by the one and only, Kenneth LaFrance. When I say the one and only, I mean he is Kenough! IYKYK lol. Following Kenâs opening remarks, Taylor Desseyn from Gundot.io opened up with content creation

The day was filled with great speakers from Alex Riviere with their talk on design systems, to Rizel Scarlett with an AMAZING talk on breaking down barriers through the use of emerging tech. Rizel is a Staff Developer Advocate at TBD. Make sure you attend a session of

theirs if you ever get a chance.

Throughout the day we had talks from Karl Groves, speaking on accessibility, to Pato Vargas displaying immense knowledge on simplifying your codebase with React monorepos. If you get a chance to watch Karl deliver his accessibility talks, and Pato literally talk about anything, don't sleep on it. Trust me.

Taylor closed out the event with another great talk. Taylor is the former Managing Director of Global Solutions at Vaco. Now Taylor is at Gun.io as a Talent Advocate. To sum up his talk, the abstract below explains it all.

Day Two

Day two kicked off like day one with opening remarks from Ken LaFrance. Ken gave yours truly an incredible introduction as I kicked off the conference. I delivered my talk on the relationship between ROI (return on investment) and a11y (accessibility). Of course I opened up with who I was, my role at Harness, and what we do at Harness. I can safely say that 99% of my audience now knows who Harness is and what we are all about. It was eye opening.

Mo Daniel gave their VERY FIRST talk ever. Mo is one of the nicest people I have met in my life. Their talk really hit home when it came to learning. They ended their talk with a positive affirmation and we all stood and clapped for an amazing performance.

I joined in on a talk where I did a voice over as a sea captain along with Abbey Perini and Nerando Johnson. That was a blast to do. My friend Mark Noonan, Senior Software Engineer at Cypress, delivered a talk on front end testing. And we all did voices as characters throughout the talk. I even dropped a Jaws reference and luckily most of the audience got it.

There were a handful of great talks over the two days of the event. I wish I could cover them all here. Todd Libby closed it out with his accessibility talk on Deceptive Patterns & FAST. And in his Halloween, East Coast fashion, he came out of the entrance where the kitchen is, dressed in a Lobster costume and uttered the words, "man is it hot in there!"

If you get a chance to attend or speak at MagnoliaJS, definitely do it. Great people, great talks, great community.

â

Social image and cover image from the MagnoliaJS Twitter. All images below are from myself or community members.

â

â

Â

â

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/trunk-based-development-with-feature-flags>

Trunk Based Development with Feature Flags

Hey all you wonderful developers out there! In this post I am going to take you through creating a feature branch and using the Harness React SDK to implement a flag in your project. This post is an introduction to getting a feature setup using Vite, React, and Sass to change the look of a logo in a navbar.

Project Setup

The world of web and software development changes every day. It is hard to keep up with, we all know that. The days of using Create React App seem to be over. In its place has stepped Vite. You can head over to the Vite site and see what it is all about, but if I had to put a description on it, it would be; CRA with superpowers. Extremely fast installs, and load times make it a great tool to start your project with. You can also choose what language you want to use when prompted in the terminal.

We are going to install Vite as the base setup for our project. Fire up your terminal and install Vite using this command:

If you are using Yarn, use this command:

You can manually install Vite with the framework of your choice or you can select the prompts in the terminal.

Great, now that we have that setup, go back to your terminal, change directories into my-project and fire up your editor of choice. If VS Code is in your PATH, you can type code . and that should open it. Open your terminal back up and run:

This will install all your dependencies and fire up your server which should be 'localhost:5173'. The rest of this article is going to assume you have installed the Harness React SDK. If not, there will be a follow up post to this on installing and implementing that specific SDK. Stay tuned for that!

Component Wrapping

â

In this example, I created a very basic navbar using semantic markup and jsx. There is a logo that is left aligned, and a navigation that is right aligned using flexbox. Please keep in mind that this navigation I built is a desktop view only and is not using any type of routing. The âlinksâ are list items within an unordered list. It is up to you the developer how you approach the routing for your component/application. YMMV.

The feature we want to ship changes the look of the logo. In the example we are only going to be wrapping the logo component in a flag, and making sure to put that in a feature branch so we can immediately keep it disabled/hidden, while merging it back into the main branch. We can always verify that the feature works by switching to our feature branch and toggling the feature on/off in the Harness app.

â Â Â Â Â Â YourLogoÂ Â Â Â

YourLogo

We are going to use CSS to control the feature. I am using Sass for this, you can use vanilla CSS. Create a feature branch according to your branch naming convention. I like to stick with lastname/whatever-the-feature-is.

Once your branch is created, add the CSS to change the look of the component and then add the class to the component itself.

Markup

YourLogo

You can validate this works on save if you flip the switch in the Harness UI and enable the flag. This post is going to assume you have the flag setup in the Harness UI already and can navigate the interface. If the feature is doing what you expect when you toggle the flag on, you have successfully created a feature branch with a feature flag. Now you can merge this into main and keep it disabled as to not release it until you are ready. You also have the option of continuing to work on said feature if you choose, just open another branch.

â

LGTM!

Everything works as expected, tests pass, the PR has been reviewed and approved, now letâs get this merged. Make sure you stick to your merging process. I personally like to squash and merge my branch but your mileage may vary. Now that we have our code with the feature in main and it is still hidden behind a flag, no one can see it until you flip the switch or someone in charge of the release flips the switch.

The great thing about using feature flags is that you can designate a target(s) in order to test the feature. This is especially useful when having someone QA the feature. Another great thing about using feature flags is separating deploy from release. The code can be deployed 100% to production and until the flag is toggled, it will stay hidden. Once you and your team are ready to release it to the world, flip the switch and voila, your feature is live!

â

â

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/harness-chaos-engineering-landscape>

Harness Chaos Engineering Faults Landscape

Harness Chaos Engineering provides a library of chaos faults using which chaos experiments are constructed and run. It is simple and intuitive to construct the chaos experiments using the given set of chaos faults. Before we delve into the details of the faults, let's look at the anatomy of a chaos experiment and the role of the chaos faults in the chaos experiments.

Quick Review of a Chaos Fault and a Chaos Experiment

A chaos fault is an actual fault or some distress injected around a system resource like CPU, Memory, Network, Time, IO system, Nodes, etc. A chaos experiment is an attempt to measure the resilience of a system when one or more chaos faults are run on it. In Harness Chaos Engineering, an experiment not only runs chaos faults, but it measures the resilience of the system in the context of the faults that were run.Â

When a chaos experiment is completed running, it provides a âResilience Score,â which indicates the resilience of the target system against the faults that are injected.Â The Resilience Score is the % of successful steady state measurements measured during the chaos experiment execution.

Resilience Score of a Chaos Experiment

A developer who is designing and implementing the chaos experiment controls the meaning of a Resilience Score. The higher the number of steady state checks or probes passed during the experiment execution, the more it contributes to the resilience score. The steady state measurements in Harness CE are done through the Resilience Probes. Many resilience probes can be attached to a fault. The more probes you add to the faults inside the experiment, the more realistic the resilience score of the experiment will become.Â

The Resilience score of a chaos experiment = The percentage of successful resilience probes in the experiment.Â

â

Chaos Fault Landscape in Harness Chaos Engineering

Faults for Kubernetes Resources

Runs via the Kubernetes Chaos Infrastructure or Agent

Supported Faults:

All Pod related faults, Node faults, http faults, IO/database chaos, network faults and load chaos. These faults are certified for the cloud Kubernetes services like EKS, AKE and GKE as well as for the on-prem versions like RedHat OpenShift, SuSE Rancher and VMware Tanzu.Â

Faults for VMWare Resources

Runs via the Kubernetes Chaos Infrastructure or Agent

Supported Faults:

Chaos faults are either injected through the VCenter APIs or through the VMware Tools directly on the operating system running inside the VM.Â Some faults such as VMÂ power off, VMÂ disk detach and VM host reboot are performed at the VCenter Level. Most of the common faults related to CPU/Memory/IO/Disk stress, http, DNS and Network faults are performed through the operating system through VMware tools. All the common faults are supported for the VMs running on Linux or Windows.

Faults for Linux Resources

Runs via the dedicated LinuxChaos Infrastructure or Agent

Supported Faults:

All faults related to resource stress, network and process. DNS error and spoof, Time Chaos and Disk are also supported. With ssh fault, network switches can also be targeted.

â

Faults for Windows Resources

Runs via the Kubernetes Chaos Infrastructure or Agent

Supported Faults:

All faults related to resource stress, network and process. Time Chaos and Disk fill are also supported. These are supported for Windows instances that are running on Azure, VMware and AWS.

â

Faults for AWS Resources

Runs via the Kubernetes Chaos Infrastructure or Agent

Supported Faults:

Deep coverage in the chaos faults for EBS, EC2, ECS and Lambda. AZ down faults for NLB, ALB and CLB. Some faults for RDS. All Kubernetes faults are supported for EKS on AWS.Â

Faults for GCP Resources

Runs via the Kubernetes Chaos Infrastructure or Agent

Supported Faults:

Faults for GCP VM disk and instance. All Kubernetes faults for GKE.Â

Faults for Azure Resources

Runs via the Kubernetes Chaos Infrastructure or Agent

Supported Faults:

Faults for Azure VM disk,Â instance and web app. All Kubernetes faults for AKS.Â

Faults for Cloud FoundryResources

Runs via the dedicated LinuxChaos Infrastructure or Agent

Supported Faults:

Support is extended for Pivotal Cloud Foundry as well as any other Cloud Foundry versions. Faults for Cloud Found App like Delete App, remove routes to app, stop app, unbind service from app etc are supported.

Faults for Spring Boot Resources

Runs via the Kubernetes Chaos Infrastructure or Agent

Supported Faults:

Chaos faults for Spring Boot Apps. App Kill, CPU Stress, Memory Stress, Latency, Exceptions and any chaos monkey fault as wrapper.Â

Conclusion

Harness Chaos Engineering supports a wide variety of chaos faults that spans across Operating systems, Cloud platforms and Kubernetes. These faults enable end users to verify the resilience of the code being deployed to the target systems or of the systems serving business critical applications.Â Check out all the Harness Chaos Faults on the Harness Developer Hub.

â

â

âSign up FREE to experience the ease of resilience verification using chaos experiments. Harness provides a free plan that enables you to run a few chaos experiments free of charge for unlimited time.

â

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/gitness-bitbucket-server>

Dreading the upcoming Bitbucket Server support deadline? Try Gitness.

Bitbucket Server will no longer be supported by Atlassian as of February 15th, 2024. After this date, product support and bug fixes will not be available. If you are currently self-hosting Bitbucket Server and investigating alternatives, add Gitness to your list.

What is Gitness?

Announced at Harness's Unscripted 2023 conference, Gitness is an open source development platform packed with the power of source code hosting and automated software delivery pipelines. Gitness is distributed as a self-contained, lightweight Docker container, you can have it up and running in less than 30 seconds.

This brief introduction video gives an overview of Gitness features:

Import Your Projects

Bitbucket Server projects can be imported directly into Gitness.

If you would rather not import entire projects from Bitbucket Server, Gitness also supports importing individual repositories.

Once you have created your projects, Gitness provides robust user access controls. Developers can collaborate and review each other's changes through pull request workflows.

Beyond Source Code

In addition to managing your source code repositories, Gitness includes native software build and test pipelines. In fact, Gitness is the next step in the evolution of Drone, a mature container-native continuous integration tool with over 100 million pulls on Docker hub.

Gitness supports multiple pipelines per repository, see sample pipelines for many popular languages and tools.

If you need some help creating your pipeline, Gitness can automatically generate pipeline YAML based on the code in your repository.

Pipelines can be triggered from events in your repositories, and you can create conditions to control which steps run for certain events (push to a branch, opened pull request, etc). Expressions can be used to provide dynamic values in your pipeline at runtime.

Conclusion

Gitness can help make your organization's transition from Bitbucket Server smooth and painless.

To learn about all of Gitness's features, see the official Gitness documentation. For questions and support, join the Gitness community slack channel.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/why-make-dashboards-when-aida-can-do-it-for-you>

Why Make Dashboards When AIDAâ€¢ Can Do It For You?

Harness Custom Dashboards â Then and Now

How We Used to Do Dashboards

Harness Custom Dashboards has long been an industry differentiator. We provide unmatched transparency and granularity into your SDLC data. While a wealth of data is undoubtedly exposed, there exists a learning curve for users that lack business intelligence experience. For executives who want answers fast, a simple data question quickly transforms into a daunting quest into the unknown seas of dimensions, measures, table calculations, etc., just to figure out what our total AWS costs were month-over-month, year-to-date.Â

Thatâs where AIDAâ€¢ comes in.

As we announced during unscripted 2023, AIDA will be integrated into our Custom Dashboards service. We are calling this Dashboard Intelligence by AIDA. AIDA allows any Harness user to quickly create dashboards using only a chat promptâone widget at a time.

How AIDA Generates Dashboards for You

For this post, we will walk through a CI/CD dashboard example. In this CI/CD dashboard, we have some beautiful widgets centered around CD, but there are no CI widgets. Letâs use AIDA to bring in some complimentary CI data.

When one edits/creates a new dashboard, an option to build with AIDA will appear in the top-right of the screen:

For this CI section of the dashboard, we have received specific requirements: Our stakeholders want to see the success rate for our CI builds, total builds by trigger type and a spreadsheet providing supporting details for our total builds. For someone new to the BI experience, this could be daunting. With AIDA, we can just ask for thisâin English. Letâs start by getting the success rate for our CI builds:

In a matter of seconds, we have a widget at the foot of the current dashboard showcasing our CI build success rate:

While this is good, it isnât as interesting as a bar chart. Letâs use AIDA to build a bar chart showing our total builds by trigger type:

As expected, we have a beautiful bar chart at the foot of our dashboardâcreated in seconds:

While bar charts, scatterplots and line graphs can be exciting, they donât always provide a lot of underlying detail. Letâs use AIDA to surface more granular detail under our CI builds:

AIDA, as expected, generated a spreadsheet giving additional detail underlying our total builds:

With just three prompts, we were able to create three informative widgets. Thanks, AIDA.

Say Goodbye to Manual Dashboard Creation

While there will still be a place for hands-on development for BI visualizations, most of the dashboarding tool will be eliminated using AIDA as a co-pilot in your dashboard creation journey. No longer does one have to endure a learning curve to learn a new business intelligence software. Now any stakeholder can simply ask AIDAâ€ what is going on with their data, and get answersâ€fast.

Interested in harnessing the power of AI across the SDLC?

Sign up for an AIDA demo today and explore the capabilities of Harness AIDA in a dynamic software delivery landscape.Â

If you are interested in learning more about how we are infusing AI capabilities across the SDLC, check out other blogs in our AI Feature Focus series to learn more!Â

- Introducing Harness Support Powered By AIDA
- Introducing AIDA Error Analyzer for Continuous Deployment

Our Plan to Infuse AI Across the SDLC

Stay informed about our exciting future developments by keeping an eye on our comprehensive product roadmap.

â

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/cd-and-the-modern-software-shop>

CD and the Modern Software Shop

âShould we have systems of this size and complexity? Is it the manufacturerâs fault for producing them or the userâs for demanding them? One shouldnât ask for large systems and then complain about their largeness.ââ

Ascher Opler, NATO Software Engineering Conference, 1968

â

My first job after college was as a field tech for a local IT provider. I reviewed firewall logs, and repaired printers, and deployed servers. I reassured worried users that their computer is likely safe despite scary popups.

I wasn't at an innovative SaaS startup, rather it was more like Office Space. At the time I wasn't familiar with modern software toolchains like containers, developer portals, CI/CD, and end-to-end monitoring. I certainly didn't understand the benefits of continuous delivery.

But what the job did provide me was time with users in the "real" world. That is, employees and business owners who want software to just work so they can do their jobs and go home. We're talking trainers strapping on heart rate monitors, factory employees clocking in and out, and support reps updating a ticket status.

The typical user is generally not the type to spend 30 hours trying to port their favorite emacs package to Rust. Rather, it's a specialist owner doing focused work. Their paycheck depends on the network staying up and their spreadsheet app not crashing. The following conclusion, then, reflects what I experienced at that MSP and in each job since.

People want their software tools to just work, with minimal friction and cognitive overhead.

This is utterly non-controversial for consumer software. Companies design smartphones for to be immediately useful upon powering them on. Personal devices and apps are walled gardens of pleasure and convenience, making choices and even "thinking" for their end users.

Now think of the enterprise; does the previous sentence still apply? Consumers desire guidance to a world of pure imagination. But organizations typically want complete control over their processes, tools, and data.

The result is layers of runbooks and delivery processes, where friction and cognitive overhead are kind of the point. It proves the organization is precise and cautious!

A great contradiction in the software trade has been the creation of delightful software using incredibly *undelightful* development practices and deployment tools. How did we get here? Are engineers as happy with the tools they use to deliver software as users are with the software itself? And have we made any progress toward this ideal?

I've noticed that the organizations that do manage to approach this utopian ideal engage in the following:

Everything is automated.

Everything is version controlled.

Everything is immutable.

Everything is observable.

To better understand how the above have become best practices, let's discuss how we arrived here.

Lessons from the physical world

The current software release process derives from the principles used in manufacturing *physical* products. This approach originated from Frederick Taylor's concepts on work management during the early 1900s. Taylor believed that business is like solving an engineering problem. He thought that the management should have control over everything needed to produce the best results at the lowest cost.

Taylorism heavily standardized and specialized production methods. On the other hand, firms implementing Taylorism became paternalistic. As supply chains got more complicated, having a small group of managers making decisions became a problem. The scientific bureaucracy slowed down innovation.

What's the analog to developing and deploying software? Probably the 1968 NATO Software Engineering Conference. The meeting aimed to predict the difficulties in treating computer science as a mature engineering field. They coined the phrase "the software crisis," which might be summarized as *implementing large systems is hard, and changing them without breaking things is even harder*.

The meeting attendees advocated deployment method we'd consider quaint today. Computer experts would completely design the software specification before writing any code. They'd then hand the specification to a programmer, who'd then code without any further decision making. It was scientific management through and through.

We now know that's not a recipe for success. Deployment environments aren't fully closed systems. For any reasonably complex system, the deployment action will impact the software performance itself.

Yet firms continued to embrace Taylorism and created extensive decision trees and playbooks to manage the division of labor. We're talking literal books of every possible decision a system administrator should make for any eventuality.

Back in the physical world, lean manufacturing took off in Japan in the 1950s. Companies like Toyota implemented a pull system that was laser focused on minimizing waste and excess inventory. In the 1980s, the book *The Goal* popularized these methods, and western companies used them to compete with affordable Asian vehicles.

Still, software remained in Waterfall-land. Sequential development took precedence over real-time user demands.

The infrastructure needed for iteration and modern version control just wasn't in place yet (CVS wasn't first released until 1986). Feedback loops were long, and tools comprised proprietary stacks of compilers, linkers, and databases. Development teams worked on centralized mainframes or minicomputers, with manual builds, testing, and deployment.

Early and often

Things finally changed in the early 1990s. NFSNET upgrades allowed TCP/IP and opened the doors to the World Wide Web.

Communities and software tools began to flourish and make use of this new communication backbone. In 1995, JUnit came out as a unit test framework to help automate testing. Apache HTTP server helped dot-com companies start using practices similar to today's continuous integration (CI) and CD.

Key to this innovation was a Cambrian explosion of open source software. It was all developed in view of the public, accompanied by a culture of rapid experimentation. Eric S. Raymond's *The Cathedral and the Bazaar* presented a mantra of **Release early, release often. And listen to your customers.**

As companies started adding open source to own tech stacks, the culture followed. Extreme programming and the Agile Manifesto brought in new project management styles. These styles emphasized user feedback loops instead of top-down directed flows like Waterfall. "Hacker culture" had now become a business model, and one that worked.

Moving up the stack

After the dot-com crash, Web 2.0 changed how developers built and released software. The internet became a network of platforms for users to generate and post content. Software was now a service in and of itself, rather than simply a supporting communication medium. Downtime was no longer acceptable and operation teams achieved their moment in the sun.

Marc Andreessen proclaimed that software was now eating the world, and would affect all areas of industry, digital and physical. Books like *The Phoenix Project* served as sequels to *The Goal*, but applied to software delivery and operations.

Cloud computing arrived when Amazon started opening its internal tools to external customers (thereby launching Amazon Web Services). Organization realized they could innovate faster by focusing on their core competencies while outsourcing lower layers of the stack. All companies could now be software companies, and tools became more abstract as "plug and play" models became common.

Teams began **shifting-left** to identify constraints as early as possible in the development process. In practice this means treating everything as software-defined.

Think of how infrastructure has undergone significant changes in the past thirty years. It has evolved from mainframes to servers, to VMs, to containers, to serverless setups and now AI-generated configs. Now think of the tools that have sprung up to support and maintain these abstractions. They include configuration management, continuous integration, infrastructure as code, and large language models.

Every problem is now a software problem, and modern continuous delivery requires thinking as such. This brings us back to the four principles identified earlier.

Everything is Automated

Scientific management, lean manufacturing, and agile all put a heavy premium on documentation to standardize release cycles. Before computers, this was just a means for later workers to have a standard process to follow. Now, computers can implement runbooks and deployment playbooks as coded specifications.

In other words, program or automate it if possible. Runbooks become configuration management and infrastructure as code. Deployment playbooks become imperative or declarative pipelines. Best practices involve using a range of tools, both from vendors and open source, along with their reference architectures.

Because we're thinking of operations as software development, is also means!

Everything is version controlled

Change management is a mature set of approaches with roots in Taylorism and lean manufacturing. ITIL standards brought it to technology in the 1980s. Today, once we represent deployment procedures as software, we can **store everything in version control**.

We document our versioned changes via commits in a code repo. Branches, code review, and pull requests represent our change management procedure.

We can go a step further and implement GitOps. This enforces that our version controlled codebase is our *declared source of truth*. Our production deployment *must* look like the HEAD of our main branch, for example. Our chain of commits representing our version history then moves us from mutable, difficult to measure deployment environments, to!

Everything is Immutable

Immutable means it can't be changed. Immutable deployments mean we can't change a deployment after it happens; we must do a new

one. If that sounds uncomfortable, think about how manufacturing companies fix product defects. Make small changes in the process that improves the *next* product off the line.

Another way to think of this is the *pets vs cattle* analogy. We think of deployments and infrastructure as livestock - interchangeable and subject to iterative improvements.

Remember version control is our source of truth. A change means a new commit on a branch in the repo. Kicking off a new deployment pipeline means parsing the full commit. The new deployment is then the delta between the new commit and the previous one.

We're swapping out cattle, not trying to apply ad hoc changes outside our automated source of truth.

Immutable therefore implies that our changes are more traceable. And because we fix our environments in place after deployment, we can move towardâ†

Everything is Observable

If we represent our deployment process as code, we can only document (and thereby automate) what we have known. Observability provides continuous feedback loops and data to drive our automation choices. Tools like Prometheus provide a core metrics exporter we can filter into dashboards or organize into DORA metrics.

Full observability isn't a panacea. Reams of data are only useful if they are interpretable and actionable. DataOps integrates the interpreting and decision making around the software-driven practices we've already described. In CD, metrics like *deployment frequency* and *mean time to resolution* directly connect to business performance, if not survival.

In Summary

CD is still a young discipline. The industry has learned from the challenges faced by producing physical goods. It has therefore developed a set of evolving best practices summarized as: automate everything you can around your deployments to immutable and observable environments, with underlying configurations maintained in a version controlled source of truth.

âHarness is among the platforms that seek to untangle the complexity and provide a deployment solution that just works, while encapsulating the principles articulated in this article. We encourage you to sign up for free and start deploying today. You can also check out the Harness Community projects on GitHub to see practical examples of modern CD pipelines built around the principles discussed here.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

[Documentation](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[Case studies](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[We want to hear from you](#)

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

[Sign up for our monthly newsletter](#)

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/drone-3-0-redefining-next-10-years-of-simplicity-innovation-in-open-source-ci-cd>

Drone 3.0 - redefining next 10 years of simplicity & innovation in open source CI/CD

Over the past decade Drone has disrupted the market with its simplicity and ease of use, growing to 10,000+ self-hosted installations. With 30k+ github stars and a robust, active open source community, Drone is a leader in container-driven Continuous Integration.

Teams choose Drone because it is simple, self-service, and you can be up and running in minutes.

Drone over the past decade has flourished through its vibrant open source community with over 100m pulls on DockerHub, 100,000+ active users, and 250+ contributors.

We want to express our heartfelt gratitude to each and every one of you for being a part of the incredible Drone community. Your unwavering support, dedication, and contributions have made this open-source project the success it is today.

With this change we are adding native Source Control management capabilities, which includes support for essential features like code hosting, pull requests, code reviews and more. This is a major evolution of Drone, from Continuous Integration to a full fledged Developer Platform. All under the Apache 2 license.

Drone Community

For the existing Drone community and customer base, nothing will change. You can continue to use Drone and don't have to migrate to Gitness.Â

Drone will also continue to integrate with GitHub, GitLab, Bitbucket, Bitbucket Server, Gitea, Gogs, and Azure Devops.

Drone will always remain open-source, and Harness will invest significantly over the coming years to its community, platform, and mission.Â

Roadmap

The power of open source development lies in the collaboration and collective effort of passionate individuals like you. Your commitment to improving Drone, sharing your expertise, and helping fellow community members has driven innovation, efficiency, and quality in our project.

You can follow the roadmap here <https://ideas.harness.io/gitness>. Please submit your ideas and feedback to help us prioritize the roadmap.

Conclusion

As we move forward, we look forward to even greater achievements and advancements with your continued support. The future holds exciting opportunities for our project, and with your dedication, we are confident that Drone will continue to be a leading force in the open-source world.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer

happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/road-to-backstagecon-2023-a-sneak-peek-into-an-exciting-lineup-a-recap-of-2022>

Road to BackstageCon 2023: A Sneak Peek into an Exciting Lineup & A Recap of 2022!

BackstageCon 2023 is right around the corner! Hosted as a CNCF Co-located Event in Chicago, this year's BackstageCon promises a plethora of insightful sessions, keynotes, and lightning talks. It will be taking place on November 6 at McCormick Place West in Chicago, Illinois.

Our very own Himanshu Mishra, Product Manager for Harness IDP module will be speaking about Backstage core and focus on 16 powerful features that makes Backstage suitable for the self-service Internal Developer Portals in his talk **What Does Backstage Really Offer à Looking Beyond Catalog and Templates and Into the Core Platform**. The talk will emphasize Backstage's superiority over homegrown portals, highlighting its flexibility and control, making it an optimal choice for businesses. Himanshu was previously part of the Backstage core team at Spotify. Previous talks like **Backstage: Shaping the Future Of Developer Experience and The Evolution of Backstage Backends**, from 2022 could be a good prelude to this talk.

Here is a sneak peek on some other exciting talks to look out for -

Roadieâ David Tuite's talk on "**How to Adopt Backstage**" continues 2022's themes focusing on **developer happiness**, emphasizing the real-world benefits of Backstage. Ryan Emerle from Comcast will discuss **how their DevHub, powered by Backstage**, addresses challenges and promotes extensions. Meanwhile, Wolfgang Gottesheim and Andi Grabner from Dynatrace will share their journey of **enhancing developer efficiency** for over 1000 engineers using Backstage, highlighting the rollout challenges and success measurements, which are on the same line of the last year's talk on **DAZNâs journey of using backstage**.

GenAI in Backstage: Ben Wilcock from VMware will spotlight the **integration of AI models** like ChatGPT in Backstage, showcasing how AI can elevate developer productivity by automating tasks and generating code. In a related **lightning talk**, Nate Axcell from TELUS will share their transformative journey with Backstage. From a collaboration tool for 2,500 employees to an internal search for 100,000 in just a year, TELUS harnessed AI capabilities, resulting in a 566% surge in Backstage adoption in Q2. This rapid adoption translated to significant productivity gains, saving each employee over 284 days of effort in two months.

At the upcoming event, there will be a deep diveÂ into Backstage adoption stories. A **panel**, featuring experts from Spotify, U.S. Bank, Expedia Group, Twilio Segment, and Lunar, will discuss their Backstage journey, touching on technical and cultural shifts. Additionally, folks from B3, a top global stock exchange, will share their unique experience. They've been able to use **Backstage to boost developer satisfaction in a regulated environment**. Join us to learn from these real-world implementations and insights.

Keynotes: A series of sponsored keynotes featuring industry stalwarts like Chris Westerhold from Thoughtworks, Balaji Sivasubramanian from Red Hat, Meg Watson from Spotify, and Scott Sisil from VMware.

Lightning Talks: Quick, insightful sessions on a range of topics, ensuring there's something for everyone.

And that's just a sneak peek! The **full schedule** has even more exciting content, discussions, and networking opportunities. This being the second edition of Backstagecon.

But that's not all! With Harness being a proud sponsor, we're excited to connect with fellow Backstage enthusiasts and share our expertise and insights with the community. If you're keen to explore how the **Harness Internal Developer Portal (IDP)** can enhance your developer experience and help you with your Software Delivery Life Cycle, check it out here.

â

See you in Chicago! ðð

â

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/get-actionable-insights-with-aida-cd-error-analyzer>

Get Actionable Insights with AIDAâ€‘, CD Error Analyzerâ

Continuous Delivery Today

In the DevOps landscape today, Harness has grown to be a leader in innovation. Continuous delivery has become a backbone in efficient and reliable software deployments, streamlining software development processes.

Using Harnessâs platform, letâs take a look at how the CDâ module works. In a dream world, every time I deploy my code to a pipeline, it would run successfully. But unfortunately, in the real world, we know that this is not always the case. Seeing the red alert that my deployment has failed leads to developer toil. But with Harnessâs new innovative AI toolkit, this stress can be eliminated.

In this blog, we will go over the AIDA CD Error Analyzer feature and how it can be used to perform root cause analysis and remediate issues.

Let's start by taking a look at some example deployments in my project. I am seeing a few failed pipelines. How can we quickly remediate that?

How does AIDA Change the Landscape?

AIDA CD Error Analyzer is an innovative AI DevOps tool engineered to provide comprehensive root cause analysis (RCA) and remediation steps for failed pipelines. RCA is important in the world of continuous delivery because it helps identify underlying causes of issues that arise during development and deployment, leading to pinpointing the error and resolution. Ultimately, RCA contributes to the delivery of high-quality software, and AIDA will help us achieve that goal.

Let's take a look at one of these failed pipelines as an example of how AIDA benefits software delivery teams. Before clicking on the purple AIDA button, let's dive into the typical workflow of resolving failed pipelines.

Manual RCA Causes Unnecessary Toil

To resolve this failure, engineers would have to open the console view to discover the error and have to sift through logs to find the root cause.

Let's take a look at what that would look like in our Harness platform.

A Better World with AIDA as Your DevOps Assistant

Now, let's ask AIDA to help us by pressing the purple button. In a matter of seconds, AIDA analyzes these logs for you to come to the relevant error message and suggested remediation step. This eliminates toil, improves productivity, and helps you continuously improve faster.

In this case, I had an image pull error. AIDA provided the root cause of the error and suggested remediation steps for me to quickly fix it.

How does AIDA CD Error Analyzer Work?

AIDA is built as an enterprise-ready privacy-first solution and is powered by innovative GenAI technology and advanced algorithms.

For AIDA CD Error Analyzer, Harness uses the execution logs to render the AI remediation responses. For more detailed information, refer to our comprehensive privacy guide.

Uncovering Actionable Insights Using AIDA

Using AIDA, teams will be empowered to write meaningful code and reduce developer toil. They will experience the following:

1. Increased Productivity:

- With AIDA CD Error Analyzer, teams can streamline their CD processes by identifying and remediating issues faster than ever before. This leads to increased efficiency in the software delivery pipeline, reducing waste and toil and improving resource utilization.

2. Rapid Issue Resolution:

- AIDA CD Error Analyzer promotes faster issue resolution by pinpointing the root causes of pipeline failures. Software teams can take immediate corrective actions with the suggested remediation step, leading to quicker turnaround times for issue resolution and software deployment.

3. Improved Pipeline Reliability:

- AIDA CD Error Analyzer helps identify and address issues early in the Continuous Delivery pipeline. By catching problems at their root, it ensures that the pipeline remains reliable, reducing the likelihood of bottlenecks or disruptions.

4. Continuous Learning:

- AIDA CD Error Analyzer facilitates continuous learning by capturing and documenting the root causes of issues and their resolutions. For new developers, this information is crucial to learn more about DevOps best practices and RCA. Additionally, for all members, this knowledge can be used to prevent similar issues in the future and to educate team members on best practices.

Start your elevated CD journey today.

AIDA CD Error Analyzer is a valuable AI DevOps tool that not only identifies issues in pipeline failures, but also contributes to eliminating toil and empowering developers. It is an essential feature to add to your developer toolbox to help you achieve your goals of

delivering high-quality software efficiently, reliably, and in the most productive way possible.Â

Interested in harnessing the power of AI across the SDLC?

With AIDA CD Error Analyzer being a GA feature, a member of our Harness team can guide you through enabling it today.

If you are interested in learning more, sign up for an AIDA demo today and explore the capabilities of Harness AIDA in a dynamic software delivery landscape.Â

If you are interested in learning more about how we are infusing AI capabilities across the SDLC, check out other blogs in our AI Feature Focus series to learn more!Â

- Introducing Support Powered by AIDA

Our Plan to Infuse AI Across the SDLC

Stay informed about our exciting future developments by keeping an eye on our comprehensive product roadmap.

â

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/hate-reading-docs-me-too-thats-why-we-have-suppprt-powered-by-aida>

Hate reading docs? Me too. That's why we have Support powered by AIDAâ¢

Navigating Customer Support in the Digital Age

In today's fast-paced world, quick and efficient access to information is more crucial than ever. Technical documentation can feel like navigating a labyrinth of vast knowledge, making it challenging to find answers promptly.

At Harness, we understand that navigating through our extensive documentation, blogs, pricing plans, and other public resources to find answers can sometimes be a challenge, leading our customers to file support tickets and wait for a response from our devoted support team. With this challenge in mind, we wanted to elevate your customer experience. That's why we're excited to introduce Harness Support powered by AIDA, your dedicated assistant ready to answer your questions about Harness and its products.

Introducing Harness Support Powered By AIDA

Harness Support powered by AIDA is designed to streamline your access to information by harnessing the power of AI. It works seamlessly inside the product, offering a user-friendly interface for asking questions. When you ask a question, Support powered by AIDA quickly scans our vast knowledge base to provide relevant, accurate, real-time answers.

After logging into your account, click the Support button in your screen's bottom right corner. From there, feel free to ask your question relating to Harness's platform, products, and documentation. You will receive a response within a few seconds with direct links to relevant documentation for easy reference. You can also provide feedback on the response so engineering can ensure the answers are aligned with your expectations.

Support Powered By AIDA In Action

First, let's navigate to the Support icon.

Now, let's ask a question about Harness!

â

As you can see, Support powered by AIDA is a game-changer in the customer support experience as it streamlines the process of asking questions and finding information. With the power of AIDA, Support swiftly addresses your questions about the Harness platform and provides solutions. It's as simple as that.Â Â Â

Now, How does Support powered by AIDA Work?

As a technical overview, Support powered by AIDA is the embodiment of advanced algorithms and innovative GenAI techniques to bring you answers to your questions.Â

â**Your Question to AIDA:** The moment you type out your query, GenAI springs into action, ensuring you get a prompt response tailored to your needs. You can ask anything related to the Harness platform, products, and documentation.

â**From Words to Numbers:** Your questions aren't just words on a screen. With the help of advanced algorithms, they're transformed into 'embeddings'â making the solutions for software troubleshooting more efficient.

â**The Power of the Vector Database:** This is where the Harness AI tools truly shine. AIDAÂ scans its vast database for the best, most accurate answer to your query.

â**Crafting Context:** A good answer isn't just about accuracyâ it's about relevance. AIDA uses its AI integration in modern customer support capabilities to craft a correct and contextually relevant response to your needs.

â**Engaging the Best in AI:** Behind the scenes, AIDA uses the most advanced AI models to ensure the user experience is second to none.

To read more about how Support powered by AIDA works behind the scenes, check out this technical post on our Â Engineering Medium.Â

The Future of Customer Support with AIDA

Our Support powered by AIDA represents a significant leap forward in customer support and access to Harness resources. With the power of GenAI, we've harnessed the ability to provide instant, accurate answers to your questions about Harness.

Harness Support powered by AIDA streamlines your experience, ensuring you have the information you need at your fingertips, making your journey with Harness smoother and more efficient. We're committed to providing you with the best support possible, and this new feature is a testament to that commitment. Whether you're a developer seeking technical documentation or a user with product inquiries, Support empowers you with the knowledge you seek 24/7 without waiting for support tickets or email responses. Harness the power of Support to supercharge your interaction with Harness products.

Get Started On Your Elevated Support Experience Today

Are you intrigued by the transformative power of GenAI in DevOps? Request a demo to explore the capabilities of Harness AIDA in a dynamic software delivery landscape.

As we conclude this AI Feature Focus blog, remember that we've only just begun to unveil the incredible capabilities of our AI. Be sure to check out the rest of our feature series to discover what truly sets our AI apart.

For a deeper look into Harness AIDA, visit our Harness YouTube channel to witness it in action. Â

Our Plan to Infuse AI Across the SDLC

Stay informed about our exciting future developments by keeping an eye on our comprehensive product roadmap.

â

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/leveraging-feature-flags-for-zero-downtime-database-migrations>

Leveraging Feature Flags for Zero-Downtime Database Migrations

Database migrations can be a complex and risky process, especially when done in production. Any downtime or data loss can have a significant impact on your users and business.

Feature flags can help you mitigate these risks and achieve zero downtime database migrations. By decoupling your database changes from your code changes, feature flag management tools allow you to deploy and test your new database schema and code in a controlled and safe manner.

Just a few months ago, we leveraged our own feature flags for upgrading customers from our First Gen CD product to the Next Gen CD version in order to ensure a smooth transition. Recognizing that some customers wanted management approval before revealing the Next Gen user experience, we employed feature flags to conceal its entry point. Additionally, to achieve parity during the initial upgrades, we enabled certain features specifically for select accounts, without affecting the larger user base. These features addressed either behavioral changes or functionalities present in First Gen but absent in Next Gen. Furthermore, Harness' feature flags played a pivotal role in the migration of Harness users. Through them, we could adjust properties in the Harness MongoDB, porting users over from First Gen to Next Gen. In essence, these feature flags ensured a controlled and secure upgrade for our customers.

How to use feature flags for zero-downtime database migrations

Here is a step-by-step guide on how to use feature flags for zero-downtime database migrations:

Benefits of using Harness Feature Flags for zero-downtime database migrations

There are several benefits to using Harness feature flags for zero-downtime database migrations:

- **Reduce downtime and risk.** By decoupling your database changes from your code changes and deploying them in a controlled manner, you can reduce downtime and risk.
- **Increase agility.** Feature flags allow you to deploy and test new database changes without having to deploy new code. This can help you increase your agility and responsiveness to market changes.
- **Improve reliability.** By enabling you to test new database changes in production with real data, feature flags can help you improve the reliability of your database migrations.
- **Leveraging a kill switch.** If you get a huge volume of data written to your database it has the potential to impact other customers. Feature flags allow you to opt to discard non-critical data per customer account. Although you don't want to find yourself in that situation, it's always good to be able to prevent one bad account from breaking others.

Examples of how Harness feature flags can be used for zero-downtime database migrations

Here are some examples of how Harness feature flags can be used for zero-downtime database migrations:

- **Migrate to a new database schema.** You can use feature flags to gradually migrate your data from an old database schema to a new one. This allows you to test the new schema in production with real data before fully cutting over.
- **Add new database columns.** You can use feature flags to gradually enable new database columns. This allows you to test the new columns in production with real data before making them available to all users.
- **Migrate to a new database platform.** You can use feature flags to gradually migrate your data from one database management system to another. This allows you to test the new platform in production with real data before fully cutting over.

Beyond database migrations

Here are some other the key benefits of using Harness Feature Flags:

- **Automated rollout pipelines:** Harness Feature Flags allows you to create automated rollout pipelines that can be used to deploy new features to production in a controlled and predictable manner.
- **Governance:** Harness Feature Flags has the most in-depth governance tool with Harness Policy as Code, powered by Open Policy Agent (OPA), providing a number of capabilities such as role-based access control, approval workflows, and audit logging.
- **Automated lifecycle management :** Harness Feature Flags helps you manage your flag tech-debt by detecting potentially stale or deprecated flags and automatically removing them if needed
- **Integrated with CI/CD:** Since Harness covers a range of software delivery products, you can easily integrate your feature flags into CI/CD as a unified pipeline that helps maintain the flags as part of the software development lifecycle.

Conclusion

Feature flags are a powerful tool that can help you achieve zero-downtime database migrations. By decoupling your database changes from your code changes and deploying them in a controlled manner, Harness feature flags can help you reduce downtime, increase agility, and improve the reliability of your database migrations.

Want to migrate your own database with zero-downtime? Visit our product page or sign up to start using feature flags.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

[Documentation](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/deploying-to-k8s-gitness>

Deploying to Kubernetes with Gitness

Gitness is an open-source Git solution designed with developers in mind. At its core, it offers a reliable space to host git repositories. However, its capabilities extend far beyond that. If you've been keeping up with recent news and announcements, you've likely come across its features for code hosting, collaboration, and continuous integration. In this blog, I'll demonstrate how you can leverage this platform to deploy to Kubernetes, thereby completing the full DevOps cycle of build and deploy.

If videos are more your style, we've got you covered:

Before you begin

You will be running Gitness as a containerized application, and during this process, you'll also deploy a sample application to a Kubernetes cluster. To get started, ensure you have the following:

- Docker engine running.
- A Kubernetes cluster (You can use k3d or a managed Kubernetes service).
- kubectl installed and user is authenticated.
- Your own Kubernetes manifest (this is optional).

A note: Ensure your user or service account on the Kubernetes cluster has the ability to create and list deployments, service accounts, roles, rolebindings, and more. It's typically advised to use a service account and its associated token instead of a user account token within a build/deploy system. Here's a manifest that includes definitions for a service account, role, role binding, and secret. By using this, you'll be able to leverage the **gitness-sa** service account to initiate deployments from Gitness.

Save the provided content as **gitness-sa-manifest.yaml**.

Let's use a dedicated namespace called **gitness**:

Once done, execute the following command to create the necessary resources:

Make a note of the **gitness-sa-token**. We'll use it later:

The **echo** command trims any newline character from within the token to avoid formatting errors when copying over.

Setup and configure Gitness

Setting up Gitness is straightforward and it's suitable for lightweight machines – yes, even on a \$4 DigitalOcean VM. If you already have the Docker engine up and running, you can get started with Gitness with the following command:

Visit `localhost:3000` in your browser and sign up using a User ID, Email, and Password. Once you've successfully registered and are presented with the initial Gitness interface, you can choose to create a new project or import an existing one. A project in Gitness is similar to a group in GitLab or an organization in GitHub.

Let's initiate a project named **gitness-cd**. After it's set up, we will proceed to add two secrets that Gitness requires to orchestrate a deployment on your Kubernetes cluster. From the left navigation pane, select **Secrets** followed by **+ New Secret**. You'll then create the following two secrets: **k8s-server** and **gitness-sa-token**.

For the k8s-server secret, use the output from the subsequent instructions as its value. To retrieve the URL of the Kubernetes API server:

We already made a note of **gitness-sa-token** in the previous section so use that value to create **gitness-sa-token** secret.

A Word on Gitness's Current Stage

Gitness is still in the early days and is going through rapid development. One of the known issues is that when using a script within a run step, the Gitness secrets are printed in the pipeline execution log in plain text. The Gitness community continues to report similar issues and this feedback will guide this platform's evolution. With your help, we aspire to set new industry standards for open-source code hosting & pipeline engine.

You might observe numerous shell script commands embedded within the CD pipeline and wonder, "Aren't we transitioning away from custom shell scripts?" Indeed, as Gitness matures, expect plugins and integrations that will streamline Kubernetes cluster configurations. This will eventually enable application deployments without reliance on shell scripting. For the time being, bear with the temporary solutions. The essence of this blog is to showcase the promising horizon awaiting Gitness.

Using Gitness for source code management

You can import an existing GitHub or GitLab repository but for this blog, let's create a new repository in Gitness. From the left navigation bar, select **Repository** and click on **New Repository**. Give this repository a name, for example, **demo-repo** and click **Create Repository**. You can keep all other options as default.

Once the repository is created, you'll see an option to add a new file. You can generate git credentials, clone the repository, and work from a terminal. Since we'll be adding one file, let's use the Gitness UI to add that file. Click on **+New File**, give the file a name **nginx-deployment.yaml**, and paste the following code:

Alright, let's just commit those changes straight to the main branch. By the way, if you've got a Kubernetes app and some manifests lying around, feel free to use yours. Otherwise, our sample **nginx-deployment.yaml** has got you covered.

Create a deployment pipeline in Gitness

The pipeline concept in Gitness is a child concept under the Repository entity. You can create one or more pipelines for a given repository. For this blog, assume that you have a continuous integration pipeline that can build your code, run tests, and push an image to an image registry. In this section, we'll create a deployment pipeline that can deploy that image to your Kubernetes cluster. Let's use **nginx:latest** as an example but you can swap this with your choice of a container image.

Getting started is straightforward:

Replace the sample pipeline with the following pipeline definition:

Click **Save and Run** from the top-right corner and watch your deployment pipeline logs to ensure that Gitness deployed an nginx application on your Kubernetes cluster. A successful pipeline execution will look something like this:

Unfolding the **cdstep** will reveal the smooth sailing of the **nginx-deployment** pods. You're probably brimming with questions about this pipeline definition, right? No worries - we're diving deep into those in the next section.

Breaking down the pipeline definition

Let's delve a bit deeper into the Gitness pipeline structure. A pipeline in Gitness can have one or multiple stages. In our example, we have two defined stages: **cistage** and **cdstage**.

Both of these stages utilize the **alpine/k8s:1.26.9** container image, which includes the kubectl tool. The **cistage** is more of a placeholder right now, simply executing an echo command.

The **cdstage**, however, does the heavy lifting. It initiates the deployment through a series of shell commands. First, it establishes a connection to the Kubernetes cluster, leveraging two secrets provided by Gitness during runtime. Next, it triggers a deployment using the **nginx-deployment** file from our Gitness **demo-repo**. To round things off, it checks to confirm the successful launch of the pods.

It's worth noting that these commands are executed within the Gitness run step, and they operate from the root directory of your git repository.

Challenges with deployment from a Continuous Integration (CI) tool

While Continuous Integration (CI) tools are indispensable for maintaining code quality, they aren't always suitable for deployment tasks. Here's why:

If you're looking for an advanced continuous delivery platform, give Harness Continuous Delivery (CD) a try! From deploying multiple services across multiple environments to comprehensive GitOps support, Harness CD can deploy any app, anywhere without any scripting involved.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/arm-harness-ci-cloud>

Lower Infrastructure Costs with Arm; Get there Faster with Harness CI Cloud

The ARMv8 (a.k.a. ARM64) CPU architecture, announced 12 years ago, was the first Arm architecture to support a 64-bit instruction set. Apple famously introduced its ARM64-based M1 chip in November of 2020. In June of this year, Apple completed its transition away from x86_64-based CPUs by launching a new Mac Pro model powered by the M2 revision of the chip. In the PC market, Arm notebooks are expected to increase to 27% market share by 2027. In the cloud computing space, all major service providers offer Arm solutions.

We have entered a world where Arm can't be ignored. If your company isn't adapting your applications and running infrastructure on Arm, chances are your competition is.

Cost and Energy Savings with Arm

The Arm architecture offers significant potential cost and energy savings. A recent study showed that moving to Arm-based cloud infrastructure can achieve up to 40% better performance per dollar over comparable x86_64-based infrastructure.

Companies like Squeaky reduced their cloud spending by 35% by switching to Arm. When analyzing CPU, memory, and latency metrics after the migration, Squeaky reports no performance degradation on the new hardware.

Database company Aerospike saw annual costs decrease by an estimated 27% after switching to Graviton2 processors on AWS EC2. They also cut carbon emissions by an estimated 49% while achieving aggressive targets for transaction throughput and data access latency.

Docker and Arm

The 2022 StackOverflow Developer Survey found that Docker is becoming a fundamental tool for Professional Developers, increasing from 55% to 69%. According to the CNCF 2022 annual survey, 44% of respondents use containers for nearly all applications and business segments, and another 35% say they use containers for at least a few production applications.

This means that **the first challenge many companies face when adopting Arm is how to build their application containers for this different architecture**. While Docker supports building images for other CPU architectures under emulation, this can significantly impact performance.

Native Builds vs. Emulation

To visualize the difference in performance, we built base Docker images for Python, Ruby, HAProxy and Redis using both emulation and Harness CI's native Linux ARM64 hardware.

This graph shows build time in minutes for Harness's native Arm hardware compared to emulation (Tested with minimum 4 vCPUs and 16GB memory):

These results show **builds completed five times faster on average with native Arm hardware**. This is valuable time your developers could be using for *development*, rather than waiting for pipelines.

While Arm clearly has incredible momentum, some companies in the continuous integration space have been slow to provide hosted resources. **Here at Harness, Arm infrastructure has been available in Harness CI Cloud since the beginning.**

Get Started with Harness CI Cloud

Feeling inspired to experiment with ARM64 in your organization? Sign up for Harness today and enjoy 2,000 free monthly Harness Cloud build credits for your pipelines.

After creating an account, you'll be guided through connecting to your Git repository and building your first pipeline.

To run builds on our ARM64 Linux runners, select **Cloud**, and then select **ARM64** on your pipeline's **Infrastructure** tab.

Next, add a Docker build step in your pipeline's **Execution** tab.

See our **Build and Push to Docker** documentation to learn how to customize this step for your application.

If you'd like to learn more, schedule a demo to see how Harness can revolutionize your software development process.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

[Documentation](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[Case studies](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer

happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?
Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/harness-unveils-four-game-changing-modules>

Harness Unveils Four Game-Changing Modules

Harness Unveils Four Game-Changing Modules

We're beyond excited to share some groundbreaking news with you. At the recent {unscripted} conference, we unveiled not one, not two, but four innovative modules on the Harness platform.

Our CEO and co-founder, Jyoti Bansal, perfectly encapsulated our enthusiasm: "The four new modules we are launching today represent a significant leap forward on our mission to enhance efficiency, foster collaboration, and fortify security throughout the software delivery lifecycle.â

Each module is designed to redefine software delivery and elevate the developer experience. Let's dive in!

1. Harness Releases Code Repository

Harness Code Repository is a premium module based on open-source Gitness (launched today) and tailored to meet the demands of enterprise teams and organizations. Gitness is a developer-friendly, open-source Git platform that addresses common obstacles in traditional software development workflows. Read more in the Gitness release blog [here](#).

Code Repository brings inÂ Collaborative code reviews, advanced governance with branch protection and policy enforcement powered by OPA to ensure code complies with predefined organizational standards and best practices, reducing manual intervention and human errorÂ and seamless integrations tailored for the development process on the Harness platform.

It's launching in Beta soon. Please sign up using this form!

2. Harness Internal Developer Portal (IDP): Empowering Developers Like Never Before

Developer portals are no longer a luxury; they're a necessity. As development organizations grow and information becomes scattered across many small teams and services, we need to help developers understand it all.

Enter the Harness IDP. It accelerates onboarding, consolidates information across the toolchain, and brings developer documentation to a central, searchable home. With features like the Software Catalog (built on the Backstage platform), Scorecards, and Self-Service Automation, developers spend less time looking for help and more time innovating.Â

Discover more about Harness IDP.

3. Harness Infrastructure as Code Management (IaCM): Automation and Security at Its Best

Infrastructure as Code Management (IaCM) is revolutionizing how companies manage cloud resources. Our module takes IaC to the next level by adding pipeline automation, governance, and security. From an advanced IaC automation pipeline to OPA-based infrastructure policies, Harness IaCM ensures efficient, secure, and compliant infrastructure management.

Dive into Harness IaCM.

4. Harness Software Supply Chain Assurance (SSCA): Secure, Compliant, and Efficient

Modern applications are being built with an exponentially increasing number of open-source components, introducing new vulnerabilities that put software consumers at risk. To help organizations extend their DevSecOps practices to enhance open-source governance and ensure artifact integrity, our Software Supply Chain Assurance (SSCA) module offers unparalleled visibility and control over open-source components. From generating SBOMs to ensuring software integrity, this module is a comprehensive solution for modern software supply chain challenges.Â

Learn more [here](#).

Exciting updates to existing modules

- **GitOps with Flux:** At {unscripted}, the Harness team demonstrated upcoming support for Flux. Harness brings a centralized

management layer, graphical dashboarding, and strong pipeline management to Flux's command-line-driven GitOps. As the GitOps community debates Argo vs Flux, our answer is, "Why not work with both?"

- **Getting started help:** We also announced the public preview of a new Harness CLI and a new Getting Started UX for this module.
- **Automated Feature Flag Lifecycle Management:** Coordinating feature flags in the code and the management tooling can be tricky. On the floor, we are showing new capabilities for archiving old flags and quickly restoring them if prematurely archived. We will also show our beta feature on automatically removing stale flags from the code base instead of manually removing them, giving time back to the development team. [Learn more here.](#)

Oh, and we have another exciting announcement before we sign off! We've initiated two industry-wide communities:

- Engineering Excellence Collective, a consortium of 300 senior engineering leaders dedicated to addressing industry challenges. [Read more about it here.](#)
- FinOps Excellence, a community-driven initiative to bring together leaders working in cloud cost management globally. [Read more about it here.](#)

Harness is committed to transforming software delivery. With our latest modules, we're one step closer to that vision. Stay tuned for more updates, and as always, happy coding!

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/delegates-and-agents-onramp-to-scale-with-harness>

Delegates and Agents - Onramp to Scale with Harness

By leveraging a Harness Delegate or GitOps Agent, you are able to model commands you would have to maintain locally for remote execution. There are a multitude of benefits to this approach which pays dividends on day two operations. Starting with a local approach to get started, can build to a remote approach.

â

Video

â

Local Machine

Core to automating commands that run on your machine is to run those commands off your machine e.g on a remote machine. Harness has a concept of Delegates that will run your commands on your behalf. When building, publishing, and deploying an application, several commands need to run. With these commands, the dependencies such as CLIs and authentication needs to be installed and authenticated against. Below, is a sample series of commands.

â

Leverage Remote, Introducing the Harness Delegate

Getting these commands off of your local machine, you could mimic 1:1 the commands you have run locally into something like a

bastion host so the remote bastion host will communicate to the external services. Building your own remote instance does have complexities that need to be managed such as access control, uptime, and maintenance such as library upgrades.

â

Harness delivers this concept for you instead and instead of mimicking your commands in lengthy shell scripts, you are modeling your commands with Harness. By using a Delegate, you are also able to position the Delegate to run on your infrastructure and ability to be separated as necessary. For security/sensitivity concerns, having work done on your infrastructure allows you to build a more secure perimeter.

â

The Harness Delegate is a containerized worker node that has the ability to be run in Docker and Kubernetes anywhere on your infrastructure. Harness just needs the ability to communicate to the Delegate so segregation is straightforward.

â

Platform Operations Challenges Solved with Harness Delegates

The power of the Harness Delegate shows itself in several scenarios. You are free to install as many Delegates as you need. The Delegates can also be scoped to restrict usage.

Direct Access to Your Kubernetes Cluster

Both models are supported by the Harness Delegate. The Delegate can have direct access which requires no configuration on your part to the Kubernetes cluster it is deployed into. If you are getting started with Harness, most users would install a Delegate into where they want their workload to be deployed, in the below example, K3d.

â

This example can be extrapolated by deploying Harness Kubernetes Delegates across multiple clusters with direct access to those clusters across multiple cloud providers. Once Harness has connectivity to those clusters, creating a multi-cloud, multi-cluster scenario is easy, just install another Delegate and use Harness as the deployment control plane.

Having multiple delegates across multiple cloud providers is not a requirement. A single Harness Delegate can handle multiple authentications and be positioned in appropriate infrastructure to run tasks.

Separation of Duties

Harness delegates can be scoped to run certain tasks or be limited to certain scopes inside the Harness Platform. In a more complicated example, Team A can deploy to GCP and Azure for disaster recovery and Team B can only deploy to AWS. This can actually be accomplished with one Delegate but for scaling/separation of duties, can wire two different Delegates to take on different workloads for each team.

Additional Delegate Workloads

The Harness Delegate is a main point of communication between Harness and your services/integrations that need to be run. For example if using Continuous Verification, the Harness Delegate is facilitating communication between your observability solutions and the Harness Machine Learning to validate your deployment. With a Delegate, you do not need to install tooling/client CLIs on the Delegate for supported workloads.

Installing a Delegate

There are multiple ways to install a delegate. Take a look at our Delegate Installation Tutorial to match your preferred installation method to the infrastructure you want to deploy a Delegate to. For very Kubernetes centric workloads and embracing GitOps methodologies, Harness also has a GitOps Agents designed to scale tasks similar to the Harness Delegate.

GitOps, a new set of Capabilities and Challenges

With Kubernetes becoming mainstream, the GitOps paradigm is becoming more popular. Leveraging the declarative nature of Kubernetes with source control, GitOps allows you to have your source of truth of Kubernetes-centric infrastructure in source control.

Local Machine

Getting started with ArgoCD from your local machine requires several commands to get started. With these commands, the dependencies such as CLIs and authentication needs to be installed and authenticated against to perform these tasks. Below, is a sample series of commands.

â

If you had only one cluster to deploy to, e.g the same cluster that is running ArgoCD in and only had one person administer ArgoCD, this model makes sense. Though a fallacy of distributed computing is that there is only one administrator. Management complexity starts to be introduced with multiple clusters for deployments and as the number of administrators/users start to scale. Some of the complexity is covered by an Argo ApplicationSet, but clusters can change. This is where Harness's GitOps Agent comes in.

Leverage Remote, Introducing the Harness GitOps Agent

Expanding on the above example, if you needed to add an additional Kubernetes cluster there are a few approaches. The first would be to install and manage multiple ArgoCD instances and have a manifest [or ability to template] to apply per cluster. Second, Leveraging Argo's ApplicationSets, can leverage an array of cluster objects via a Cluster Generator. Third, would allow Harness's GitOps Agents to handle this complexity for you.

If you open your local KubeConfig file, there is a combination of a certificate [e.g ca.crt], a role/token, and an address needed to communicate with the Kubernetes Admin API Endpoint. That is how connections are made between clusters. That is the information that is needed to wire communication with another cluster. Leveraging an ApplicationSet, does come with the point of view from the Argo project that ApplicationSets are a privileged entity and have security implications.

To facilitate abstraction and the GitOps work that is needed to be performed, Harness has created a GitOps Agent that will facilitate and execute commands on your behalf. You do not need to have an existing ArgoCD installation to leverage the Harness GitOps Agent. In the below example, you only need one manifest that will be synchronized across multiple clusters, which can even be across multiple providers. The wirings that are needed to communicate back to Harness if installing a GitOps Agent from your Harness Account comes pre-wired.

Also with Harness, you can bring your existing ArgoCD instances to be managed with the Harness GitOps Agent(s).

Art of the Possible with Harness GitOps Agent

Similarly in a more complicated example, Team A can deploy to GCP and Azure for disaster recovery and Team B can only deploy to AWS. This can actually be accomplished with one GitOps Agent but for scaling/separation of duties, can wire two different GitOps Agents to take on different workloads for each team. Only Team A has an ArgoCD instance and Team B does not have to manage their own ArgoCD instance.

Installing a GitOps Agent

Installing a Harness GitOps Agent is straightforward. Can follow the documentation and pick the installation type you would prefer. Can decide if you are bringing your own ArgoCD instance or letting Harness takeover complete management of the GitOps infrastructure. If using AWS, you can have the GitOps Agent assume an IAM Role to access underlying AWS Infrastructure via IAM.

Agents and Delegates, Your Onramp to Scale

In technology, complexity never goes away. The adage that complexity is like an abacus, just moving left to right holds true. Installing a Harness Delegate or GitOps Agent is something else to install but by taking the initial effort to install an entity to do work on your behalf, underlying distributed system complexity burden can be less as you and your organization scale. If you have not signed up for the Harness Platform, sign up today!

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/elevating-software-reliability-with-smart-feature-flags>

Elevating Reliability in Software: Harnessing Smart Feature Flags for Velocity and Performance

As digital transformation sweeps across industries, companies are increasingly realizing the critical role of software reliability in meeting their business objectives. Reliable software ensures a smooth and consistent user experience and is the backbone for organizational growth and customer trust. It's about speedy software delivery and providing operational resilience post-deployment.

What exactly are feature flags? Traditionally, they are a framework developers can use to turn specific code paths on or off. You can think of feature flags as extending Continuous Delivery into Continuous Deployment - a way to put changes into production and turn them on in a controlled way later (or hide and remove them in the same way).

At Harness, our feature flags are smart. We give developers more than an on/off toggle of their code path but a means to drive their business and customer outcomes with high velocity and improved performance.

Harness Smart Feature Flags empowers developers to:

- Use pipelines to release with confidence
- Easily manage flags with a Git experience
- Employ automated lifecycle management and cleanup
- Enforce governance for velocity without risk
- Access 24/7 service monitoring for all flag changes

The Core of Reliability

Reliability reflects the ability of a software service to meet or exceed user expectations. This includes availability, latency, correctness, and other aspects influencing user experience quality. The stakes are high - poor reliability can negate the most effective software delivery, impacting organizational outcomes. Integrating reliability considerations early into the Software Development Life Cycle (SDLC) is no longer optional.

Take, for example, our customer case study with Entur. Entur used Harness Smart Feature Flags to gain control and increase deployment frequency as their development teams doubled. Entur increased how often they deploy changes from twice a week to 14 times a day.Â

Additionally, they have reduced the failed change rate from 20% to 5%. Moreover, the time it takes to undo changes has been significantly decreased from 2 hours to just 5 minutes. Harness helped the business go faster and be more reliable for customers without choosing one over the other.

The SRE Approach and Its Impact

Site Reliability Engineering (SRE) offers a user-oriented measurement, shared responsibility, and blameless learning, emphasizing a collaborative culture. SRE is not merely about implementing a set of practices; it's about nurturing a culture that promotes continuous learning and adaptation to manage complex, dynamic technology environments.

While marked by setbacks and lessons, the journey toward enhanced reliability ultimately leads to a resilient system. Investment in SRE improves reliability but requires patience and persistence. The key lies in gradual, measured implementation, avoiding the pitfalls of organization-wide overreach.

A Real-world Application: Feature Flags at Play

Imagine a globally recognized e-commerce platform planning to launch a new recommendation algorithm designed to enhance user shopping experiences. After internal testing, everything seems ready. However, the risk associated with rolling it out all at once could be catastrophic if any unforeseen issues ariseâleading to potential lost sales and customer trust.

By leveraging smart feature flags, this platform implements a phased rollout. Releasing the new feature to a few users first helps spot problems with performance or user experience immediately. As the release proves stable and effective, the percentage of users experiencing the new feature gradually increases, ensuring reliability and a positive user experience.

This approach safeguards the user experience and empowers the development team with data-driven insights to refine and improve the feature further. It's a win-win and made possible through the strategic use of feature flags.

Empowering Developers with Reliability

Developers wield immense power in today's fast-paced software landscape, where innovation is the heartbeat and user satisfaction is the goal. But alongside innovation, reliability is paramount. By empowering developers with the tools and practices to ensure reliability, we're enhancing user experiences and reinforcing the foundation of business growth.

Reliability isn't an afterthought—it's the cornerstone. Equipping developers with the knowledge and strategies to infuse reliability from the earliest stages of the Software Development Life Cycle (SDLC) transforms development into a reliability-driven journey. This isn't a shift in isolation; it's a united approach that aligns development, operations, and user satisfaction.

Enter Smart Feature Flags

Reliability is not just a production event; it must be baked into the entire SDLC. Here's where smart feature flags come into the picture. Harness leverages smart feature flags to move fast and ensure exceptional reliability.

Smart feature flags go beyond the traditional "on" or "off" control of features. They enable real-time monitoring of software changes, thus maintaining a constant vigil on reliability metrics. By correlating reliability data (SLIs, SLOs, and burn rates) with feature deployment, developers gain confidence in rolling out features without adversely impacting user experience.

Navigating Complexities with Smart Feature Flags and Adaptive Deployment

In today's fast-paced software delivery environment, multiple teams work simultaneously, creating an intricate web of changes. By providing 24/7 reliability monitoring into a feature flag, we can monitor dependencies and internal consumers of a product, ensuring the change doesn't trigger any collateral damage downstream.

In environments where software alterations happen in real-time, anticipating every potential issue can be challenging. By coupling adaptive deployment with smart feature flags, developers have the agility to adjust feature releases based on real-time feedback dynamically.

If a particular feature shows signs of unreliable software delivery or introduces performance bottlenecks, the feature's rollout can be paused, modified, or even rolled back seamlessly. This not only preserves the integrity of the live environment but also guarantees that the end-users remain shielded from any disruptions or inconsistencies, further enhancing the overall user experience.

The Evolution Towards Autonomous Ecosystems

As we move towards a future powered by autonomous ecosystems, smart feature flags adapt and evolve, scaling and protecting user experience based on changing environments. These adaptive flags help control an ecosystem of customer experience, factoring in reliability, cost, and value.

Reliability is more than a mere system attribute; it's an ethos that guides the entire software development and delivery process. By making reliability an integral part of the SDLC, developers can focus on innovation and speedy delivery, knowing that smart feature flags have their back, ensuring that the customer experience remains top-notch even amid the continuous churn of updates and changes.

With smart feature flags, we're not just aiming for high velocity; we're setting the stage for reliability at speed. So, as your organization steps into the future of software delivery, remember that reliability and speed are not at odds - with the right tools and approaches, they can be two sides of the same coin.

Welcome to a new era of reliability in software - powered by Smart Feature Flags. Sign up for one of our monthly webinars and workshops and create a free forever feature flag account today. Happy coding!

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/ci-aida-developer-velocity>

Increase Developer Velocity with AIDA® and Harness CI

Developer velocity is about removing barriers and friction points by adopting the best tools and practices to enable the full potential of your development talent. McKinsey's 2020 Developer Velocity report determined the four conditions with the greatest impact on business performance are tools, culture, product management, and talent management. AI assistance in development workflows was an emerging category at the time, and the report recognized that top-quartile companies were "increasingly exploring the use of AI in developer tools."

AI Throughout the SDLC

Harness AI Development Assistant (AIDA®) employs AI to identify and address issues in every stage of the software development life cycle. While most AI developer tools primarily focus on code generation, AIDA is woven into the fabric of every stage of the software delivery life cycle in Harness.

AIDA and Harness CI

In Harness CI pipelines, AIDA can identify and address issues early in the development process. Build failures slow down developers, causing them to babysit pipelines. **AIDA helps developers quickly troubleshoot and resolve build failures by explaining errors in human readable format and suggesting effective fixes.**

When a Harness CI pipeline fails, AIDA analyzes the logs, correlates the error messages with known issues, and suggests possible remediation steps. This allows developers to quickly identify and rectify problems, increasing productivity and reducing toil.

Get Started with Harness CI Cloud and AIDA

AIDA for CI is available today in all Harness accounts. Sign up for Harness today and enjoy 2,000 free monthly Harness Cloud build credits for your pipelines.

To get started with AIDA for Harness CI, you must agree to the AIDA EULA.

Then, you'll get error analysis and remediation from AIDA when a step fails. When viewing the failed step's logs, select the Harness AIDA dialog to review error analysis and troubleshooting suggestions.

If you'd like to learn more, schedule a demo to see how Harness can revolutionize your software development process.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/migrate-from-drone>

Migration Made Easy - Automatically Migrate From Drone To Harness CI

Changing CI providers is often a challenging and tedious task. To make it simpler, we've created the Harness Continuous Integration (CI) Migration Utility. It's a free, open-source tool that makes it easier for users to move their CI pipelines from various CI providers such as GitLab, CircleCI, GitHub Actions, and Bitbucket to Harness.

For those using Drone, our migration utility does more than just convert their pipelines. It also imports everything from your Drone server to the Harness platform, automating the vast majority of the migration work, saving a good amount of time and effort.â

Watch It In Action

In this short demo, we'll demonstrate how to utilize the Harness CI Migration Utility to swiftly export Drone configuration and pipelines, and import them into Harness. This process automatically sets up in Harness the respective projects , pipelines, webhooks and triggers, enabling users to start building with Harness CI right away.â

Why Harness CI is Your Ideal Choice?

Harness CI, powered by Harness Modern Software Delivery Platform, provides a comprehensive solution designed to accelerate builds and streamline the development workflow, freeing up developers to spend more time to focus on what they do best - coding.

Harness can effortlessly scale to meet the needs of the largest organizations, while empowering developers with the autonomy they need to ship high quality software quickly, efficiently, reliably, and securely.

Below you can find a list of the key capabilities and benefits you'll experience when using Harness CI:

Speed and Cost Efficiency with Harness CI

- **Cache Intelligence:** Reduce build time with 1-click automatic caching of dependencies and Docker layers.
- **Test Intelligence:** Smartly run only the tests related to your code changes, reducing test cycle time by up to 80%.
- **Superior cloud build machines** that are optimized for speed with no noisy neighbors to slow you down.â

Ensuring Security & Compliance with Harness CI

- **Secure cloud build machines:** get isolated build machines with with Harness no-share infrastructure for maximum security and keep your code safe

- **Extensibility Through Plugins - Safely** - Tap into the collective wisdom of community plugins by leveraging Drone Plugins, Github Actions and Bitrise Steps actively in your CI Pipelines. With OPA-based policies in place, you can establish guardrails while still allowing developers the freedom to use their favorite integrations.
- **Top-Notch Governance and Compliance** - Harness provides robust, enterprise-grade governance for your CI/CD processes. With OPA-based policies, you can centrally enforce quality and security standards across all pipelines. Additionally, platform features such as Templates, Secrets Management, granular RBAC, Approvals, Notifications, and Audit Trails ensure a CI/CD process that's secure, compliant, and efficient - perfectly tailored to your organization's needs. Read more about Harness Platform
- **Shift-Left Security** - Seamlessly integrate security scanners and orchestrate tests anywhere across your build pipelines. Enable developers to rapidly remediate vulnerabilities through intelligent prioritization and deduplication with Security Testing Orchestration .
- **Achieve SLSA L2 Compliance** by leveraging Harness Software Supply Chain Assurance (SSCA). Harness will automatically generate SBOM (software bill of materials) and SBOM attestations to ensure artifacts integrity. View SLSA L2 requirements.

Boosting Productivity with Harness CI

- **Simplify troubleshooting** with Harness AIDA®, Harness AI Development Assistant, which offers a range of capabilities to enhance the developer experience. This includes swift analysis of build and deployment failures along with actionable remediation suggestions, thereby boosting efficiency and reducing the time required to resolve such issues.
- **Increase Collaboration and Standardization** with granular centralized templates. Capture best practices and onboard team members and projects easily.
- **Declutter your DevOps Toolbox** consolidating your entire software development lifecycle (SDLC) under the Harness umbrella to streamline your Build, Deployment, Release and Security processes.
- **Comprehensive Dashboards and Reporting** - Harness offers detailed insights such as DORA metrics across builds, security issues, test outcomes, deployments, and more, so you can easily spot obstacles for faster software delivery. Plus, you can create customized dashboards tailored to your unique requirements.

Join the Open-Source Journey

In the spirit of collaborative innovation, we're making the Harness CI Migration Utility available as an open-source Git project. We invite developers worldwide to contribute to its development and enhancement, furthering our mission to streamline DevOps processes and eliminate developer toil.

Get Started with Harness CI

Sign up for Harness CI today, and enjoy 2,000 free monthly credits for builds and tests on Harness Cloud. Request a demo to understand how our CI/CD pipeline can revolutionize your software development process.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

[Documentation](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[Case studies](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[We want to hear from you](#)

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/harness-announces-engineering-leadership-community-to-codify-engineering-excellence-standards>

Harness Announces Engineering Leadership Community to Codify Engineering Excellence Standards

Pioneering initiative brings together 300 senior engineering leaders to codify best practices and elevate engineering excellence

San Francisco, Sept 21, 2023 - Harness, the Modern Software Delivery Platform® company, is proud to announce the establishment of the industry-wide Engineering Excellence Collective™, a groundbreaking engineering leadership community. The Collective comprises 300 esteemed senior engineering leaders, CTOs, and MDs from leading global organizations, including Broadcom, CrowdStrike, Encora, NetApp, Palo Alto Networks, Pipe, Wells Fargo, Oracle, Xactly, OutSystems, Pure Storage Inc. and many more. The Collective represents an unprecedented collaborative effort to drive innovation and advance industry-wide initiatives, the first of which is a comprehensive Engineering Excellence Maturity Model. To learn more about the Collective and information about joining please visit engineeringx.org.[®]

“I am thrilled to participate in this amazing initiative to define and codify best practices. This is a big gap for engineering leaders. This model serves as a guidepost for understanding how to move the needle and challenge ourselves to get better. It really scratches my itch to understand how other leaders are solving similar challenges and how we can learn from the collective wisdom of engineering leaders,” said Karan Gupta, VP of Engineering at Palo Alto Networks.

“It was a pleasure collaborating with some of the top minds in the field to put together this much needed compendium of best practices,” said Preeti Iyer, Distinguished Engineer, SVP Enterprise Architecture at Wells Fargo.

The Engineering Excellence Collective’s aim is to foster an environment of knowledge exchange and best practice sharing among top engineering minds to stimulate industry progress. At the heart of this initiative is the group’s first major initiative, the Engineering Excellence Maturity Model, a cutting-edge framework that outlines 11 crucial pillars essential for achieving engineering excellence in software development. These include:

- Planning and Requirements Process
- Discoverability and Documentation
- Developer Environment Experience
- Development Process
- Build Process
- Quality and Resilience Testing
- Deployment Process
- Integrated Security and Governance
- Metrics and Insights
- Incident Management
- Learning and Development

This comprehensive model provides a prescriptive guide to elevate engineering practices and drive excellence and developer experience within organizations. Each pillar encompasses a set of capabilities necessary to reach the full potential of engineering achievement. Unlike existing frameworks, the Engineering Excellence Collective doesn’t replace established models like DORA or SPACE. Instead, it supplements them by offering specific guidance on capabilities and processes needed to enhance DORA metrics.

“Harness has used the groundbreaking Engineering Excellence Maturity Model to create an innovative survey assessment tool. The Harness Engineering Excellence Survey Assessment, publicly launched today, empowers engineering teams to gauge their excellence across each facet and obtain an overall Engineering Excellence Score. Teams can also compare their scores against industry benchmarks, receive tailored recommendations, and prioritize improvement areas. For more information on the Survey Assessment please visit <https://assessment.harness.io/assessment/invite>

“Engineering leadership can be a lonely job. We are changing that by creating a community for engineering leaders to learn from each other and codify best practices. The initial initiatives we’ve worked on – the assessment and the maturity model – give engineering leaders a unique perspective on the strengths and weaknesses in their engineering practices, and put together a holistic plan of improvement prioritized by areas that will be the most impactful,” said Nishant Doshi, GM of Software Engineering Insights at Harness. “This initiative is a collaborative effort to bridge gaps in codifying engineering best practices and provide peer-based learning opportunities that foster progress, growth, and innovation.”

Harness is committed to promoting industry-wide innovation and excellence. The initiative demonstrates Harness’s dedication to providing the tools and resources necessary for engineering leaders to navigate the complex landscape of engineering excellence.

For more information about the Engineering Excellence Collective and the Engineering Excellence Maturity Model, visit [www.engineeringx.org](https://engineeringx.org). We welcome new members who are passionate about driving engineering excellence in their organizations.

Harness is the leading end-to-end platform for complete software delivery. It provides a simple, safe, and secure way for engineering and DevOps teams to release applications into production. Harness uses machine learning to detect the quality of deployments and automatically roll back failed ones, saving time and reducing the need for custom scripting and manual oversight, giving engineers their nights and weekends back. Harness is based in San Francisco. Please visit www.harness.io to learn more

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/gitness-your-ultimate-open-source-development-platform>

Gitness: Your Ultimate Open Source Development Platform

We are thrilled with the release of Gitness, our new open source development platform packed with the power of code hosting and automated DevOps pipelines. This groundbreaking release of an open source Git repo, empowers developers with a powerful solution for writing and delivering code with confidence.Â Â

Gitness is a new open source Git solution with the breadth and depth of features to empower development teams to write and deliver code with safety and speed. It's a one-stop-shop where you can host your git repositories and automated devOps pipelines seamlessly. Say goodbye to juggling multiple tools, and hello to streamlined collaboration, and a smoother development workflow.

Why Gitness? Let's Break It Down:

Streamlined Collaboration and Speed: Gitness facilitates collaborative code reviews through automated status checks and mandatory reviewers, fostering teamwork and maintaining code quality.

Supercharged with AI: Gitness integrates AI-powered semantic search, helping developers quickly navigate and understand the codebase, thus expediting development and debugging.

Automated Pipelines: Create pipelines with ease, leveraging the hundreds of plugin steps integrating with your tools and technologies.Â Pipelines are triggered automatically for code changes, and viewable directly in the pull request.

Security: Robust branch protection rules and user access management ensure secure code management, reducing risks associated with unchecked changes.

Get started with Gitness in minutes

Gitness is ultra lightweight. With a simple Docker command, Gitness is up and running in 30 seconds and runs on a \$4 Digital Ocean server!Â Thereâs no learning curve, just start creating the Git repositories you know and love, in the intuitive UI packed with the features you need.

Automated import makes getting started with Gitness a breeze. You can effortlessly import your existing GitLab Groups or GitHub Orgs into Gitness. The import will not only migrate all of the repositories, but even detect any GitHub Actions or GitLab pipelines in those repos and convert them to Gitness YAML. With a single click your repos and pipelines are ready to go, and you can start developing and building your code in Gitness.Â

Get started developing in Gitness today!Â Check out gitness.comÂ

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/internal-developer-portal-public-preview>

Step into Tomorrow: Harness Internal Developer Portal is Ready for the Public

Three months ago, we announced the Beta release of our Internal Developer Portal (IDP) module to elevate the Developer Experience and streamline Platform Engineering practices. Today, we are thrilled to announce the Public Preview release! We couldn't have reached this milestone without the support and invaluable feedback from our early adopters. Thank you for helping us in building the best and most beloved Software Delivery Platform.

At Harness, we understand that success is a collaborative effort. During our Beta release, we joined forces with dozens of our customers, most of whom were dedicated Platform Engineers and DevOps enthusiasts. Together, we aimed to tackle common challenges and make the developer's journey smoother. Some of them built developer Self-Service Flows to improve **new service onboarding time by 100x** - from weeks to minutes. While many used the Software Catalog to **organize their internal software ecosystem** and reduce cognitive load for developers. Let's dive deeper into these use-cases of Harness IDP, what is new and what is about to come.

Unlocking Developer Self-Service

Expert research analysts like Gartner predict that in a couple of years, 75% of organizations will have a self-service developer portal. We are already witnessing this transformation among our customers. Self-service means that developers spend more time building features for their customers, rather than being blocked on other people and manual processes. It is critical in go-to-market time for newer projects in many companies.

â

Harness IDP empowers organizations to achieve developer self-service in a number of ways -

- **Powerful Pipelines:** Our pipelines act as a robust low-code/no-code workflow orchestrator, facilitating end-to-end service onboarding which includes repository and toolchain setup, approvals, CI/CD configuration, integrations with third party tools like Jira and ServiceNow, and more.
- **Guard Rails for Security and Governance:** While developer's experience is important, we respect the need of compliance, governance and security. Harness IDP ensures that the necessary guardrails are in place with granular Role-Based Access Control (RBAC), Open Policy Agent (OPA), approvals and audit trails.
- **Intuitive Self-Service Flows:** IDP enhances the user experience by providing developers with a collection of self-service flows. They can effortlessly discover what they can create, provision or systems they can get access to - dramatically reducing the time it takes to resolve their platform needs.

Getting Organized with Software Catalog

In today's fast growing microservices-based world, a Software Catalog is foundational. Catalog is a core feature of Harness IDP. And that is why instead of reinventing the wheel and the standard, we adopted the most popular and highly scalable catalog out there. Our IDP is powered by the popular CNCF incubating project Backstage.io. The YAML specification for defining software components is extremely flexible and widely adopted in the industry. Catalog also brings the UI to create a single pane of glass for every software component. Here's what our customers love about the software catalog -

â

- **Establish service ownership:** Nothing is more frustrating than needing help with a service or API and not knowing who to go to. The catalog makes ownership clear, saving time and angst.
- **Single pane of glass:** Catalog presents a single view for each software component with necessary information like builds, deployments, documentation, environments, dependencies, and more. This approach significantly reduces cognitive load, making it easier for developers to focus on what they do best.

Introducing Scorecards - Measure Service Maturity

â

Platform Engineers work very hard to establish best practices within their org, but lack tools of enforcement. They fail to measure if their internal services are keeping up with them. Whether these are DevOps standards or just development best practices, it's important for developers to be aware of them and how to take actions. Here are couple of use-cases that Harness IDP Scorecards helps you solve -

â

- **Define and enforce standards** - If you have got a static page called *DevOps best practices*, it's time to convert them into dynamic checks and measure against your software components to get a score.
- **Drive migration and adoption** - Want all services to upgrade to the LTS version of Java? Want to replace your testing toolkit with something else? Put it in a scorecard and track the progress across all your services.

â

Our journey so far

â

Throughout our journey from Beta to Public Preview, we have prioritized rapid customer feedback and implementation, addressing requests within days at times. We are committed to evolving our IDP with a rapid pace. Here are some of the features that we have added since Beta -

â

- Introduced Scorecards, a solid use-case of Internal Developer Portals
- Expanded our library of supported backstage.io plugins
- Out of the box integration with our CI, CD and Feature Flags modules
- Faster and assisted onboarding for existing Harness customers
- Proxy support for secure connections to private infrastructure using delegates
- Support for GitHub and Google OAuth based plugins
- Improved self-service flows with user-aware input fields
- Unified Search experience for developers by extending IDP search with Confluence
- Harness API support to facilitate in-house automation using the catalog

â

This doesn't include dozens of bug fixes and system improvements! All our technical documentation can be found on developer.harness.io including tutorials and release notes.

â

What's next?

â

Our journey with IDP is far from over and we are innovating at a rapid scale. Here is a small list of what is coming next in Harness IDP -

â

- **Custom Plugins:** You will be able to write your own custom plugins using the backstage.io plugin framework to make Developer Portal your own!
- **Auto-discovery of Services and Dependencies:** Using Harness Network Maps, IDP will soon offer auto-discovery of software components and dependencies without having to manually define them in the YAML.
- **Improved self-service flows:** We have heard a lot about how complex it can be to set up these flows. We will be taking this

experience to the next level with out of the box steps, ease of setup and a much more interactive user interface for developers.

- **Adoption Insights and Dashboards:** Integrating with Harness Custom Dashboards, Platform Engineers will be able to gain deeper insights about the adoption within their organization to measure success and value gained through Harness.
- **Custom data sources:** It's inevitable for platforms like IDP to be fully extensible. You will be able to bring your own data sources using APIs or files in your git repositories to create your own Scorecard checks based on them.
- **Deeper integration with Harness platform:** Our enterprise customers love the features provided by our underlying platform. Whether it is the granular Role Based Access Control, Audit Trails or Open Policy Agent - we are integrating IDP to seamlessly work with our platform.
- **On-prem support:** Harness IDP is SaaS as of now. But several of our customers like to manage Harness on their own premises. Harness IDP will be available to try on our self-managed platform.

Our roadmap is based on all the feedback we have received so far but is not set in stone. YOU can influence the roadmap today, by signing up to try Harness IDP.

Try Harness IDP today

We invite you to explore our Internal Developer Portal (IDP) and experience the future of Platform Engineering. Check out more details about the product on [harness.io](#) and request for a demo. If you have any questions or want to share your thoughts, feel free to directly email us at idp-interest@harness.io.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/harness-releases-gitness-open-source-git-platform>

Harness Releases Gitnessâ Open Source Git Platform

Release of first significant open source developer platform in nearly a decade offers a unified solution for source code management and CI pipelines, improving collaboration, enhancing security, and automating tasks in software development workflows

San Francisco, Sept 21, 2023 - At the {unscripted} conference, Harness, the Modern Software Delivery Platform® company, announced the availability of Gitness™, a fully open source Git platform that brings a new era of collaboration, speed, security, and intelligence to software development. Gitness is freely available today at gitness.com.

Two key obstacles that developers and teams grapple with in traditional software development are the lack of quality open source platforms and the complexity of managing multiple tools. Most existing solutions come with a hefty price tag, lack essential features, or have significant feature bloat, making the product difficult to both use and maintain. At the same time, managing and integrating disparate tools leads to time-consuming administrative tasks and potential integration failures, causing bottlenecks, errors, and security risks in the development process and contributes to developer toil.

Gitness addresses these challenges by providing a comprehensive, new open source Git platform where code hosting and automated pipelines are seamlessly integrated, eliminating the need to juggle multiple tools. Setting up and getting started with Gitness takes minutes, simplifying collaboration, ensuring security, and optimizing development workflows, unifying source code management and devops pipelines in one platform. To get started with Gitness, use Docker run to pull the image and start the Gitness container, sign in, and import their existing Git repoâall in a couple of minutes. Gitness packs the power of code repo hosting, collaboration via pull requests (PR), PR checks with Drone-based CI pipelines, and many more capabilities.

Gitness includes:

- **Streamlined Collaboration and Speed:** Gitness enforces collaborative code reviews through automated status checks and mandatory reviewers. This approach accelerates teamwork and ensures code quality.
- **Enhanced Security:** With branch protection rules and robust user access management, Gitness promotes secure code management, minimizing risks associated with unchecked changes.
- **Efficient Code Management:** Gitness integrates AI-powered semantic search, enabling developers to quickly navigate and understand the codebase, expediting development and debugging.
- **Automated Pipelines:** Every code change triggers automated CI pipelines, optimizing performance through caching, reducing manual intervention, and ensuring consistent build and testing processes.

One of Gitness's standout features is its easy migration process. Developers can effortlessly migrate their existing repositories and pipelines, including those from GitHub and GitLab, to Gitness. This automated migration ensures a smooth transition, enabling developers to start leveraging Gitness's benefits in a matter of minutes.

Harness also announced Harness Code Repository, a premium module based on open source Gitness tailored to meet the demands of enterprise teams and organizations. Harness Code Repository provides enhanced features and capabilities, like governance, with OPA policies and fine-grained RBAC, and seamless integrations tailored for the development process on the Harness platform. Harness Code Repository will be available in beta next month.Â

"Gitness marks a significant milestone for Harness and the software development community and represents our commitment to driving innovation and empowering developers worldwide. As the first significant release of an open source Git platform in nearly a decade, Gitness is equipped to provide *all* developers with the tools they need to streamline their workflows, collaborate effectively, and ensure code quality," said Jyoti Bansal, CEO and cofounder at Harness.

About Harness

Harness is the leading end-to-end platform for complete software delivery. It provides a simple, safe, and secure way for engineering and DevOps teams to release applications into production. Harness uses machine learning to detect the quality of deployments and automatically roll back failed ones, saving time and reducing the need for custom scripting and manual oversight, giving engineers their weekends back. Harness is based in San Francisco. Please visit www.harness.io to learn more.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

We want to hear from you

Enjoyed reading this blog post or have questions or feedback? Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/accelerating-harness-cd-gitops-learning-for-new-users>

Accelerating Harness CD & GitOps Learning for New Users

To address the need of accelerated learning for new users especially when using the Harness CD & GitOps Free plan, we are pleased to announce the public preview of a new "Get Started" UI that takes the user through a guided learning journey. This UI is available for

all users (including those in the Team & Enterprise Plans) starting today and uses sample apps that are publicly available at the harnesscd-example-apps repo. This repo is forked from the popular argocd-example-apps repo to remove the burden of learning even a new sample app. Additionally, it automates the necessary setup of Harness resources using the newly announced, open source Harness CLI. Refer to this blog post to learn more about the Harness CLI.

Harness CD & GitOps is easily the most advanced solution on the market today for automating complex application deployments at scale. It supports the automated deployment of multiple types of applications (such as containerized, serverless or traditional/monolithic and more) onto multiple types of infrastructure platforms (such as Kubernetes/GitOps, AWS ECS/Lambda/EC2, Google Cloud Functions/VMs, private data centers and more). And it does so while using golden pipeline templates that codify the governance standards of an organization.

However, this breadth and depth of functionality can come across as intimidating to a new user who wants to learn the basics of Continuous Delivery and Continuous Deployment using a structured, guided path. Note the focus on learning the âContinuousâ part here since a manual deployment from your local machine onto your target infrastructure while being the simplest does not provide the ability to automate deployments to test and production environments as soon as new artifacts or container images are available as outputs of your CI pipeline.

Key Features

- Coverage for an **extensive breadth of 16 deployment types** with more on the way.
- Easy setup through **pre-populated commands specific to your Harness account** (such as login using API token) and GitHub username/PAT.
- Use of the Harness CLI to **simplify setup of all Harness resources** such as pipeline, service, environment, connectors and more.
- Step-by-step tutorials on Harness Developer Hub for the same scenarios with instructions on how to use the sample app as a starting point for bringing your own app to Harness.

Take it for a Spin - Video Tutorial

Thanks to Nicholas Lotz, our new Developer Advocate, you can get started here by simply watching a video.

Take it for a Spin - Step-by-Step Tutorial

For this blog post, we will see how to do a canary deployment of a Kubernetes Helm Chart using the new UI.

Prerequisites

Setup your Kubernetes cluster

You should have access to a Kubernetes cluster. For simplicity, we will use minikube in this post.

Create free Harness account

You will need a Harness SaaS account to complete this tutorial. It is free to sign up if you don't have one already. After signing up, you will get access to the free tiers of many Harness modules including the Harness CD & GitOps module used in this tutorial. Specifically for Harness CD & GitOps module you will land on the âGet Startedâ page below.

When you click on the above blue button, you will see the various deployment options available.

Select Helm Chart deployment using Harness Kubernetes Helm

As described in the Native Helm vs Kubernetes Helm comparison doc, Harness supports 2 different approaches for deploying Helm Charts. The Native Helm approach is for users who want Harness to simply delegate the deployment to the helm cli running on the Harness Delegate. In this approach, only Rolling deployments are supported and features such as versioning and rollback are limited to those of helm's native features. On the other hand, the Kubernetes Helm approach allows a user to leverage Harness to its maximum potential by enabling advanced rollout strategies such as canary and blue/green as well as Harness features such as versioning and rollback. The only downside is that you may have to edit your Helm Chart to confirm to the Harness Kubernetes Helm requirements highlighted here.

Since we want to do a canary deployment of a Helm Chart, we will pick the Kubernetes Helm option as shown below.

Install the Harness Delegate on your Kubernetes Deployment Target

You will need access to a Kubernetes cluster that will be the deployment target for the Helm Chart. In the CD Pipeline approach, you will create a Pipeline that will automate the deployment on your behalf and push the new application version into your target Kubernetes cluster.

You need to install a Harness delegate on your Kubernetes cluster as a prerequisite for the CD Pipeline. This delegate will then receive various tasks (such as secret decryption, connecting to private systems and more) from your CD pipeline and automatically run them on your behalf. The end result of the delegate is that you neither have to open any outbound ports from your cluster to Harness SaaS nor you

have to provide Harness SaaS with any special permissions to do deployments on your behalf. The delegate takes care of both the authentication and authorization aspect between Harness SaaS and your cluster.

Note that the delegate itself can be deployed as a Kubernetes service (with its own Helm Chart or Manifest) or a Docker container that has network connectivity and permissions to deploy into your Kubernetes cluster. The in-cluster Kubernetes delegate option is the simplest option so let's pick that as shown below. Make sure to note the delegate name you give here since that will be used in the next step.

â

Download Harness CLI and set up GitHub access

We are now ready to set up some prerequisites for creating the Harness resources. First is to download the Harness CLI. Thereafter fork the harnesscd-example-apps repo into your own GitHub account so that you can make modifications as necessary on the path to bringing your own app after you complete this tutorial. You will also create a new Harness API key and GitHub Personal Access Token (PAT). The API key will be used by the CLI to authenticate/authorize with your Harness account while the PAT will be used by Harness to access the sample code for the Helm Chart.

Create the required Harness entities/resources

At this step, all you have to do is simply copy and run the commands on your local machine. The Harness CLI will create a secret, 2 connectors, a service, an environment and an infrastructure on your behalf in your Harness account.

â

Create the canary deployment pipeline

We want to deploy our guestbook Helm Chart using a canary rollout strategy so that's what we will pick below. And then we will copy and run the command on our local machine.

â

Verify the setup

Now we will verify if the pipeline creation was successful.

â

If there was any issue that led to the pipeline not being created then you will see the following.

â

â

If the pipeline creation was indeed successful, then you will see the following. Click Next to move forward.

â

Run the pipeline

Click on the "View and run your pipeline" button to run the pipeline.Â

â

You will see Harness CD & GitOps module in action where the guestbook helm chart is now deployed onto your Kubernetes cluster using a canary rollout strategy.

The Road to GA

Today's launch is an important first step towards helping new Harness users understand the power of Harness CD & GitOps. In the coming weeks, we will be adding GitOps deployments into the UI so that the same learning journey can be realized even in the GitOps context. Additionally, we will be bringing in advanced Harness pipeline features like triggers, variables, templates, Continuous Verification and Harness Terraform Provider as follow-up to the first deployment. Our end goal is to help new Harness users grow into advanced Harness users in the shortest time possible so that they are able to bring the joy of continuous delivery (think hourly or daily deployments) to their own apps without the need for any manual intervention and with all necessary quality, security and governance gates fully configured into the pipeline.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/mono-repos-harness-idp>

Got Monorepos Instead of Microservices? This is How Harness IDP Has Got You Covered

Software development is ever-changing, and so are the tools and methodologies developers use. While microservices are the recent buzz and discussions on mono vs polyrepos are ongoing, monorepos have been a staple for a long time. These monolithic repositories, which house code for multiple projects in a single location, have their merits. They simplify dependency management, make code reviews efficient, and offer a unified versioning system.

However, the rise of Cloud Native tools and technologies has shifted the focus towards microservices, influencing tools like Backstage. Backstage, an open-source developer portal, has become a favorite for many organizations, especially those leaning towards microservices. But what about those still using or transitioning from monorepos?

â

Monorepos, while beneficial in many ways, come with their set of problems for backstage/Internal Developer Portal use case:

Plugins: Tools like Backstage have plugins, such as GitHub insights, that are designed with the assumption that a single repo corresponds to a single software component. In a monorepo setup, this can lead to inaccurate results, as multiple components might reside in the same repository.

Here's where the Harness Internal Developer Portal comes in. While it's true that our primary design has been microservice-first, and monorepos might seem like an afterthought, the IDP is equipped to handle both. The Harness Developer Portal is versatile enough to cater to both microservices and monorepos, ensuring no organization is left behind in their development journey.

How IDP takes special care of monorepos

This Harness Internal Developer Portal aims to enable organizations to handle monorepos as efficiently as microservices, especially given the challenges they present. Here's How:

Conclusion

In conclusion, monorepos offer a flexible and efficient approach to software development. With the ability to place YAML definitions

anywhere, keep documentation organized, streamline service onboarding, and have precise path-aware checks, tools like Harness Internal Developer Portal make managing monorepos simpler and more effective. Whether you're a large enterprise or a growing startup, embracing these advantages can significantly enhance your development workflow.

Try out Harness Internal Developer Portal (IDP), we've got a wealth of resources to help you dive deeper. Please reach out to us for a demo. For a comprehensive understanding, our release blogs detail the latest features and updates. And if you're more of a visual learner, our video tutorials, and documentation offer step-by-step guides to get you started. Dive in and discover the transformative power of Harness IDP!

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/how-quizlet-automates-pull-requests-for-cloud-cost-recommendations>

How Quizlet Automates Pull Requests for Cloud Cost Recommendations

This is a guest post by Harness customer Brian Walsh, Sr. Platform Engineer, Quizlet

Managing cloud costs is always a challenge, especially when we need engineers to take time away from other work to look at and take action on cost savings recommendations. Historically, this was also true for my company, Quizlet, which is a global learning platform that provides engaging AI-powered study tools to help people practice and master whatever they are learning.Â

Quizlet implemented Harness Cloud Cost Management some time ago for its cost savings automation features, which went well, but we had a number of recommendations for additional cost savings that weren't being acted upon. As a Senior Platform Engineer in the Platform team, I was tasked to find a solution to the problem.

Knowing our organization, our team knew that to empower our engineers to take action on cloud costs, we needed to make things as easy as possible for them, with as few clicks as possible for the owners of the cloud services. This led us to create an automated workflow to create pull requests for the Harness recommendations, focusing on our microservices as a starting point.

Going Where the Engineers Are

Before I go into detail about the process and how it works, I think it's important to point out that just creating the automated pull requests in our GitHub repository wasn't enough to empower the engineers to take action. They don't live in the repository on a daily

basis, so they weren't seeing the PRs or reviewing them. We had 0 engagement at the start from any of the 10+ engineering teams that used cloud infrastructure for their applications.

The first step I took to increase engagement was to integrate with Slack so that all new PRs were sent to a dedicated Slack channel that the microservices owners were part of. This raised the visibility of the new PRs immediately, as well as enabling the service owners and the Platform team a place to discuss the impact and implementation of the PRs as needed.Â

The second step was to present this new workflow to the engineering management team to get their support for the engineers to take action on these PRs as they came in. Between these two actions, we went from 0 engagement, to **75-80% engagement** on the automated PRs! These PRs on average **saved us 40%** of our previous spend on Kubernetes workloads they were applied to.Â

How the Automation Works

The automation runs once a week, for the development environment services on the first day, and for the production environment services the day after. It's a Google Cloud Function launched from the Google Cloud Scheduler, architected to integrate with the Harness API (to pull the recommendations), our microservices infrastructure GitHub repository to retrieve and edit the configuration files, and Slack for posting the recommendation PRs to a shared microservices Slack channel.

This is what the architecture looks like:

We've configured the automation to pull recommendations from the Harness API that are over a certain \$ amount, configured by the admin. This helps to keep the engineers focused on recommendations that have a more material impact on costs, so that we strike a balance on cost savings and engineering productivity.Â Â

Because we have a well defined repository structure for the microservices, it has enabled us to more easily automate searching the repo for the necessary configuration files to retrieve and edit. The Harness recommendation includes the name of the Kubernetes workload, allowing us to navigate directly to the correct directory for the service. Over time, we've accumulated the use of both Helm and Kustomize for configuring our microservices, so we added the automation needed to be able to differentiate between the two in order to find the correct config file path and the config values necessary to modify. This did add a small bit of complexity that wouldn't be necessary if we were using a single Kubernetes configuration tool.Â

Once the file is retrieved, the recommended changes are written into the YAML configuration file, then the file is pushed to GitHub, and a PR created. The weekly PRs are automatically assigned to the service owners via the infrastructure repository's CODEOWNERS file, so it's important to keep this file up to date as ownership changes over time. Once done, notifications are sent to Slack for all PRs created during the scheduled job that week.

Reviewing Pull Requests

With the Slack integration, it is very easy to engage with the service owners and answer any questions they may have about the PRs or the potential impacts to prod if implemented. We've automated the creation of an easy-to-digest PR description for these recommendations, again to make things as easy as possible for the service owners. Once the PR is reviewed, it gets approved (or rejected) by the service owner, and merged when ready.

There are a lot of PR and recommendation questions that the engineers ask frequently, so we created an FAQ to answer some of the most common questions, such as:

- **How do I know the recommendation won't break my application in prod?** Nothing is ever guaranteed, but the Quizlet Platform Team has configured the Harness recommendations to, on average, best match the application needs for our organization (i.e. performance-optimized vs savings-optimized). We've verified those settings against a variety of applications, though do recommend using your service owner knowledge to tweak and combine the recommendations with appropriate horizontal pod autoscaling (HPA) settings to yield optimal results.
- **What if I want the recommendation to be slightly less aggressive?** You can always modify the PR with less aggressive CPU and memory settings, and apply those. We include a link to the recommendation in Harness in the PR, which allows our engineers to tune the buffer meter on the right column of the recommendation page to generate proportional cpu/mem recommendation numbers going forward.
- **How do I know the PR will be assigned to the right person?** It's the service owners' job to make sure that the CODEOWNERS file properly points to the correct group. If that is done, then the PR will be automatically assigned to the correct group.Â

If You're Curious to Learn More

We couldn't make this automation and new reporting processes work without having the right tools in place, and Harness Cloud Cost Management has really enabled us to do more with our cloud. You can read more about Harness' Automated Cost Savings features, or talk to Harness sales to get more information on how it's working for us.Â

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/gitops-at-scale-terraform>

Need for Automation - GitOps at Scale

Building a skyscraper is a lot like building software. Initially, you might experiment with materials on a different site or check how beams handle stress. Once confident, you lay the foundation and then start building upwards. Similarly, in software development, you kick things off with a proof of concept (POC) on a small scale. Once you're convinced of the tooling's capacity, you scale up to production. GitOps is no different.

Beginning typically with trials and using Git as the backbone for declarative infrastructure and applications, the challenge arises as you scale: how do you efficiently manage everything? And just as you'd need machinery to help construct a skyscraper, in the world of GitOps, tools like Terraform become invaluable, especially when setting up or "bootstrapping" these automation tools. In this blog, we'll navigate through the early, middle, and advanced stages (day 0, day 1, and day 2) of GitOps and explore how Terraform can simplify the scaling process for day 2.â

Day 0 GitOps Challenges: Laying the Groundwork

Day 0 in GitOps can be likened to the preparatory phase of constructing a skyscraper where materials and designs are rigorously tested. At this initial stage, you're not delving into the complexities of scaling. Instead, you're laying the groundwork, familiarizing yourself with the fundamental principles of GitOps.

A practical starting point is to set up a basic GitOps workflow to deploy a Kubernetes manifest from a single git repository to a single cluster. At this juncture, the majority of tasks can be managed directly through the CLI or UI. The environment is simple enough that manually invoking commands or using native interfaces is sufficient. While tools like Terraform offer automation benefits, they're not essential at this stage. The primary objective of Day 0 is exploration: understanding the tools, assessing their fit, and choosing the best ones for your specific needs. The experience and knowledge garnered here form the bedrock for the challenges and complexities of the stages ahead.

Day 1 GitOps Challenges: Setting the Foundation

Day 1 in GitOps is like getting the ground ready for building a skyscraper. After your first dive into GitOps, the real work begins.

A big task is figuring out how to keep application code and configuration separate. When your app has services across multiple Git repositories, how do you make sure everything deploys smoothly and works together?

Next, as you start automating your CI pipeline, there's a tricky part: if you push manifest changes to a Git repository, you might accidentally start an endless cycle of build jobs and commits. You need to set up your pipelines right to avoid this mess.

Also, as you add more to your app, developers will need to deploy different services, like deployments and stateful sets. Each of these might have its own settings. While tools like Kustomize can help, if you're not careful, you can end up with "overlay folders hell". It becomes hard to know what's where and manage everything.

Day 1 is all about sorting out these early challenges, setting things up so you're ready for bigger tasks and scaling in the next stages.

Day 2 GitOps Challenges: Automation Complexities

As we progress to Day 2 of our GitOps journey, we encounter several challenges:

1. **Bootstrapping:** Tools like ArgoCD are great for automating deployments, but how do we set up ArgoCD itself? This is the classic bootstrapping problem in the GitOps context.

2. **Tool Reliability:** Automation is essential for system consistency. But this is true ONLY if the automation tool is reliable and consistent.

3. **Understanding the End Goal:** With declarative tools like Kubernetes and Terraform, we define the desired outcome. However, we need to clearly understand and define that end state first.

Terraform offers solutions to these Day 2 challenges:

- Tackling the Bootstrapping Dilemma: One of Terraform's core strengths is its ability to set up and provision tools. So, when it comes to initializing GitOps tools like ArgoCD, Terraform can declaratively define the infrastructure, software installations, and configurations. This helps sidestep the "chicken or the egg" problem because Terraform is the tool you use to deploy your deployment tools.

- Ensuring Reliability and Consistency: Terraform is designed with idempotency in mind, meaning you can run the same configuration multiple times and get the same result. This ensures that your infrastructure is reliable and consistent. If something drifts from the desired configuration, Terraform will recognize it and can revert it back, ensuring that the tool itself becomes a reliable part of the automation process.

- Setting Clear End Goals with a Declarative Approach: With Terraform, you define your infrastructure as code in a declarative manner. You specify what you want the end state to look like, and Terraform figures out how to achieve it. This directly answers the challenge of needing to know your end state: you define it in your Terraform configuration, and the tool takes care of making it happen.

For a closer look at how Terraform can boost GitOps automation, the following webinar provides more insights:

Build software like skyscrapers

Just as constructing a skyscraper involves meticulous planning, testing, and execution in phases, the journey of GitOps follows a similar trajectory. We start with small-scale experiments, lay a robust foundation, and then scale up, tackling complex automation challenges along the way. Each stage has its hurdles, but as we've seen, with the right tools and strategies, these challenges can be addressed effectively. Terraform, in particular, stands out as a battle-tested tool in this journey, providing solutions to many of the complexities of Day 2.

If you're keen on elevating your GitOps game, especially in terms of automation, I strongly recommend giving the Harness Terraform Provider a spin. Let's build software like skyscrapers - strong, scalable, and awe-inspiring!

â
â
â
â
â
â
â
â
â

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/the-impact-of-github-copilot-on-developer-productivity-a-case-study>

The Impact of Github Copilot on Developer Productivity: A Case Study

Developer productivity is a critical factor in the success of any software development project. The continuous evolution of software development practices has led to the emergence of innovative tools aimed at streamlining the coding process. GitHub Copilot, introduced by GitHub in collaboration with OpenAI, is one such tool that utilizes advanced AI models to assist developers in generating code snippets, suggesting contextually relevant code, and providing coding insights. To scale developer efficiency, one of our customers adopted GitHub Copilot, leading to increased collaboration and shortened development cycles, as demonstrated by Harness SEI's comprehensive analysis.

The Github Copilot Impact

- Found a significant 10.6% increase in PRs with Copilot integration.
- Demonstrated a 3.5 hours reduction in cycle time, enhancing development efficiency.

Challenges Before Adoption of Github Copilot

Before implementing GitHub Copilot, developer teams grappled with challenges primarily centered around pull requests (PRs) activity and cycle time in their software development processes. The existing workflow exhibited limited PR activity, leading to isolated development efforts and sluggish code review cycles. This hindered collaboration among developers and extended the time taken to integrate changes. Additionally, the cycle time from task initiation to deployment was longer than desired, resulting in delayed feature releases and impacting the product's ability to swiftly respond to market demands.

Manual code reviews were time-consuming and inconsistent, exacerbating the efficiency challenges.

These issues collectively created bottlenecks in collaboration, resource allocation, and timely delivery of software solutions.

Solution

In this study we tried to investigate the impact of GitHub Copilot on developer productivity, with a focus on the number of pull requests (PRs) and cycle time, within the context of a comparative analysis conducted using Harness SEI. The study was guided by the expertise of the Harness Software Engineering Insights SEI and involved a sample of 50 developers from a customer. The study took place over multiple months. In the first 2 months, the developers worked without GitHub Copilot's assistance. In the last few months, they used GitHub Copilot as an integrated tool in their coding workflow. Throughout the study, various performance metrics were collected and analyzed to gauge the impact of Copilot.

Result

The study measured the impact of GitHub Copilot on two important metrics:

- Average number of pull requests (PRs)
- Cycle time

Average Number of Pull Requests (PRs)

The average number of PRs is a critical indicator of development activity and collaboration. The analysis revealed a significant increase of 10.6% in the average number of PRs during the month when developers utilized GitHub Copilot compared to the month when Copilot was disabled. This increase suggests that GitHub Copilot can help to improve collaboration, as developers using Copilot can potentially iterate more rapidly, leading to increased code review and integration.

Cycle Time

Cycle time is defined as the time taken to complete a development cycle from the initiation of a task to its deployment. It is a fundamental measure of development efficiency. The study demonstrated a reduction in cycle time by an average of 3.5 hours during the month when developers leveraged GitHub Copilot, representing a 2.4% improvement compared to the month when Copilot was not used. This reduction suggests that GitHub Copilot's assistance in generating code snippets and offering coding suggestions contributes to quicker task completion and ultimately shorter development cycles.

Conclusion

GitHub Copilot has demonstrated the product's potential to transform software development. The increase in pull requests (PRs) and the reduction in cycle time are two key metrics that demonstrate the positive impact of GitHub Copilot on developer productivity.

Key Takeaways

- GitHub Copilot has the potential to significantly improve developer productivity.
- The tool's ability to generate code snippets and suggest contextually relevant code can help developers to save time and focus on more creative and strategic tasks.
- GitHub Copilot's ability to provide coding insights can help developers to improve the quality and efficiency of their code.
- The study findings underscore the importance of AI-powered tools in the future of software development.

Harness SEI was used to facilitate this study. To summarize, the study proves the capability of GitHub Copilot to significantly improve developer productivity. However, there is still more to uncover. We are conducting more experiments and a more thorough analysis of the experiment data we already collected, looking into heterogeneous effects, or potential effects on the quality of code. We plan to share our findings in further case studies.

To understand developer productivity and unlock such actionable metrics and insights, please schedule a demo of the Harness Software Engineering Insights module here <https://www.harness.io/demo/software-engineering-insights>.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?
Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/harness-product-modules/code-repository>

Source URL: <https://www.harness.io/blog/release-management-with-feature-flags>

Release Management with Feature Flags

Release management is the process of planning, coordinating, and deploying software changes to production. It's a critical part of the software development lifecycle and good risk management, ensuring that changes are made safely and reliably, with minimal impact on users. Feature flags can be a powerful tool for release management, helping you reduce the risk of deploying buggy or incomplete features to production, and rolling out new features gradually to minimize disruption to users. Let's take a look at how we can achieve those stress-free releases using feature flags.

Canary Deployments

Canary deployments is a type of progressive delivery where you gradually roll out a new feature to a small subset of users at first, and then gradually increase the percentage of users who have access to the feature over time. This allows you to monitor the performance and reliability of the new feature in a controlled environment before rolling it out to all of your users. If ever an issue arises, you can halt the rollout, allowing you to contain the effect it has on users.

Enforced Governance

Governance is a key part of release management as it controls who can manage the feature flags. Governance rules such as role-based access control, approval workflows, and audit logging allow you to ensure that the flags are being used responsibly and prevent unwanted or unexpected features from being released to your users.

Automated rollout pipelines

Having an automated rollout pipeline can be used to deploy new features to production in a controlled and predictable manner. This can help you reduce the risk of human error and ensure that your releases are consistent and repeatable. As part of good release management, you and your team should decide on the standardized process that any release will follow. This will be a mixture of team specific rules along with a mixture of the canary deployment and governance rules discussed above. Once that's all decided, setting up an automated pipeline will allow teams to release any feature without the stress of unpredictability.

Feature Flags as Kill Switches

If you discover a problem with a new feature while releasing it to production, you can use a feature flag as a kill switch to quickly and easily toggle off the feature. This way, instead of having to run a whole rollback that involves multiple teams and on-calls to figure out what is happening, you can easily switch the feature off and figure out what went wrong at your development team's own pace. Most importantly, using feature flags here will ensure that your users are affected minimally. It is also good to have an effective incident management plan for issues like these that may arise, which you can read more about here.

Using Harness Feature Flags for Release Management

Harness Feature Flags is a comprehensive feature flag solution that provides all of the features you need to manage your releases effectively. It is easy to use, scalable, and secure.

Here are some of the key benefits of using Harness Feature Flags for release management:

- **Automated rollout pipelines:** Harness Feature Flags allows you to create automated rollout pipelines that can be used to deploy new features to production in a controlled and predictable manner.
- **Governance rules:** Harness Feature Flags has the most in-depth governance tool with Harness Policy as Code, powered by Open Policy Agent (OPA), providing a number of capabilities such as role-based access control, approval workflows, and audit logging.
- **Individual and Group Targets:** Harness Feature Flags allows you to test new features with select audiences, and gather valuable user feedback, by releasing a feature only to individual or group targets.
- **Ease of use:** Harness Feature Flags is a user-friendly solution that is easy to get started with. It provides a clear and concise UI allowing you to easily implement workflows like progressive delivery.
- **Integrated with CI/CD:** Since Harness covers a range of software delivery products, you can easily integrate your feature flags into CI/CD as a unified pipeline that helps maintain the flags as part of the software development lifecycle.

Conclusion

By following the tips above, you can improve your release management process and release new features to your users more safely and confidently. Harness Feature Flags is the perfect tool for release management as it provides all of the features you need to manage your releases effectively.

Sign up for a free trial of Harness Feature Flags today and start experiencing the benefits of reliable release management.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/my-first-experience-using-aida>

AI-Assisted DevOps: My First AIDAâ€ Experience

DevOps Today

Today's tech scene is ever-changing, and in this dynamic landscape, DevOps has become a crucial link between development and operations. It's all about making the process smoother and faster for getting code into production. As businesses and technologies continue to evolve, there are new opportunities for refining and enhancing the efficiency of DevOps processes.

Let's discuss another exciting thing: the integration of Artificial Intelligence (AI) into DevOps. This is a new frontier that's ready to shake things up in how we approach and handle development and operations.

Now, How Can Artificial Intelligence Help?

The integration of AI into DevOps encompasses various aspects with distinct objectives. In my research, I've observed that many AI developer tools primarily concentrate on enhancing code generation and improving developer productivity.

Harness adds some new innovative capabilities to the landscape. Harness's AI Development Assistant, known as AIDAâ€, was designed to address developers' pain points, envisioning a world where developer toil is eliminated. It's a privacy-first, enterprise-ready solution that harnesses AI throughout the entire software development life cycle (SDLC).

I was curious about how AI could help me with my own DevOps endeavors, so I was excited to test out AIDAâ¢.

First Impressions: Unboxing the AI Assistant

Harness plans to have AIDAâ¢ infused into every stage of the software delivery life cycle. Currently, Harness is offering 3 major capabilities for me to explore in my own environment. Let's talk about them.

Continuous Integration:Â Build and Deployment Troubleshooting

The Continuous Integration Module now offers an AIDA capability. Here, AIDA can troubleshoot build and deployment failures.

From my experience as a software engineer, I found that AIDA's integration into the Continuous Integration Module streamlines the troubleshooting of build and deployment failures. AIDA's log analysis significantly alleviates the challenge of pinpointing deployment failure causes. Instead of sifting through lines and lines of log files, AIDA pinpointed the root cause, relevant error message, and even offered a remediation step to implementation.

This feature is designed with a keen understanding of software developer grievances and has a clear goal of minimizing developer toil. Furthermore, it simplifies the debugging process, making it more accessible even to developers with limited experience in dealing with pipeline failures. In fact, one could consider AIDA as a valuable learning tool.

Imagine you're a developer eager to initiate a CI build, only to encounter a sudden failure. Now, picture being able to swiftly pinpoint the error within seconds. Pretty cool, right?

â

Security Testing Orchestration:Â Explain and Fix Vulnerabilities

Next, I experimented with the second feature integrated with the Security Testing Orchestration Module. This capability goes above and beyond to identify and help resolve security vulnerabilities in real time.

What really impressed me during my experimentation with AIDA was how swiftly it scanned my code and pinpointed a security flaw. But it didn't stop there â AIDA went on to explain the nature of the risk, providing me with valuable insights into the intricacies of effective security testing orchestration.

With this important information in mind, I was able to further understand the nuances of proper security testing orchestration. It offered step-by-step remediation guidance tailored to my specific programming language, making the process of addressing the error a breeze.Â

Cloud Cost Management:Â Auto-generate Cloud Asset Policies

The last feature I tested in my environment was for the Cloud Cost Management Module. As someone still learning about the complexities of cloud assets, I was excited to try this feature. It auto-generates cloud asset policies with natural language and even explains what they mean.

Thus, I was able to read AIDAâs comprehensive description of cloud governance rules and learn from them. This feature really democratized policy expertise and makes cloud governance more accessible to learn about in practice.

Overall, I liked how there was no learning curve barrier to user experience of these features. They were intuitive to use and helpful. More than just being helpful, these tools assisted me in learning more about the SDLC. These features are just the beginnings of AIDAâs impact on the software delivery lifecycle. Harness will roll out more capabilities this year that will eliminate developer toil across the SDLC.

Data Privacy and Security Considerations

As the generative AI landscape continues to evolve, itâs essential to understand data privacy and security. AIDAâ¢ is made with a privacy-first approach and ensures the safety of user data. The AIDA terms and privacy policy were clear to understand and transparent.

Future of AI in DevOps: What Lies Ahead?

At Harness, we're actively shaping a future where AI seamlessly integrates into every aspect of the software delivery process. Our aim is to create an environment that's not just efficient but also incredibly intuitive. This vision of an AI-infused future marks a pivotal moment for DevOps, one that promises to revolutionize how Harness approaches development and operations, making it more cohesive and impactful than ever before.

Moreover, Harness's commitment to this future extends to the exciting array of new features we're introducing across the entire SDLC. These innovative features have been meticulously crafted with the goal of alleviating toil and empowering developers to focus on what truly matters: creating exceptional software. By streamlining processes and enhancing collaboration, our advancements aim to elevate the entire DevOps landscape, making it more dynamic and responsive to the needs of modern software development.

With each new AIDA capability Harness unveils, we're taking a significant step towards a future where the synergy of AI and DevOps not only accelerates delivery but also fosters a culture of creativity and innovation.

Want to Try Harness AIDA®?

AIDA® empowers DevOps teams and team members to align seamlessly with their business goals by streamlining the process of delivering and testing software. Its AI-driven capabilities not only enhance efficiency, but also elevate the overall software development experience, making it a valuable asset for modern DevOps practices.

Interested in learning more about AIDA®? Request a demo today!

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/october-2023-product-updates>

October 2023 Product Updates

At Harness, we have been hard at work so you can delight your customers without facing software delivery toil. Here is what has been changing in the previous month across Harness Products.

Continuous Delivery & GitOps

Full Continuous Delivery & GitOps Release Notes

Continuous Integration

Full Continuous Integration Release Notes

Cloud Cost Management

Full Cloud Cost Management Release Notes

Service Reliability Management

Chaos Engineering

Full Chaos Engineering Release Notes

Internal Developer Portal (IDP)

Full Internal Developer Portal Notes

Software Engineering Insights

Continuous Error Tracking

Full Continuous Error Tracking Release Notes

Platform

Here are some important improvements to our platform that should enhance your user experience:

Full Platform Release Notes

Self-Managed Enterprise Edition

Full Self-Managed Enterprise Release Notes

Early Access

Continuous Delivery

Continuous Integration

Security Testing Orchestration

Chaos Engineering

Software Engineering Insights (SEI)

Self-Managed Enterprise Edition

Full Early Access Release Notes

Continue the Journey

- Learn more hands-on with our increasing list of Tutorials.
- Further your knowledge with one of our Certifications.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/harness-product-modules/software-supply-chain-assurance>

Source URL: <https://www.harness.io/blog/harness-developer-hub-ease-of-authoring-with-git-triggers>

Harness Developer Hub - Ease of Authoring with Git Triggers

It's been about a year since we launched Harness Developer Hub [HDH] in Beta. Today, HDH is GA and is serving tens of thousands of unique visitors every month and hundreds of thousands of pageviews every month all across the globe. All of this while supporting hundreds of contributors with varying levels of skills. The traffic and number of contributors in the public repository continues to grow as we expand the capabilities of HDH.Â

Looking at how HDH is architected, HDH is a Docursarus Implementation. Our site embraces documentation-as-code as a paradigm and is no different than any other modern TypeScript [Javascript] based application. We have an application that multiple contributors need to contribute to and needs to be built and deployed all throughout the day.Â

Over the previous year we have made two shifts in how we build and deploy. We now treat every commit as a potential release and build multiple times throughout the day with every git commit and also deploy multiple times throughout the day with every merge to our main branch. Let's look at our current solution and then jog down memory lane how we evolved.Â

Current HDH Pipeline Strategy

We leverage several Harness capabilities to deliver HDH to the world.

Starting at the Repository

Our source code management solution is the source of truth for HDH and the genesis of changes being published. We have webhook events that fire on several SCM events to Harness to process.Â

- Branch or tag creation / deletion.Â
- Pull Request Events - Created/merged/synchronized/updated/closed
- Git Pushes

These events are then processed by Harness.

Harness Build and Deploy - Conditionally from Git Hooks in the Cloud

Our goal is to provide preview/ephemeral builds for changes that are represented in a Pull Request. To do this, we need to remotely build the Docursarus instance which leverages Yarn and NPM to facilitate the build. We build on every net new commit to the PR.Â

We build via a Harness Cloud [hosted] build node so we do not have to manage build infrastructure and dependencies on the build node. We also leverage for performance Cache Intelligence on a conservative estimate sped up our builds more than 30%. From when we implemented the current setup, we have had over 9000 builds.Â

From a deployment standpoint, we deploy to our static host which is Netlify. The flexibility and extensibility of Harness allows us to bring a plugin that interacts with Netlify's APIs. We have a decision that we make in JEXL if a build needs to head to a preview environment or if a build needs to be published to production.Â

Preview Logic [if branch is not main]:

Production Logic [if branch is main]

Configuring this Harness Trigger, here is our YAML configuration looking out for a few events.

Based on the condition, we fire a slightly different request to the Netlify API. Once we get the results of the Netlify API call, we comment back to the GitHub PR. This allows the contributor to preview their work in a live site if a preview flow is executed. In totality, the Pipeline looks as follows in the Harness Editor:

For example the Cache Intelligence step is easy to weave in during the Build Stage. Once execution will look as follows in the Harness UI:

Pipelines are designed to evolve. We had two other renditions of the Pipeline which we optimized over the year to produce what we are currently leveraging today.

Pipelines Should Evolve

We have embraced two principals as we evolved our pipelines. The KISS Principle to take a more simplistic approach and DRY Principle to cut out duplicate steps/tests. Our second rendition was Kubernetes heavy for the static site before we optimized on calling the Netlify APIs directly for a preview build; we used to maintain our own preview environment when Netlify provided this out of the box. Because we learned of this feature, we were able to easily modify our HDH Pipeline to leverage this new methodology.Â

If you like to continuously improve your software delivery capabilities, I would implore you to consider signing up andÂ using the Harness Platform to help you with your goals.

Cheers,

-Ravi

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/cost-reporting-for-chargeback-and-showback-why-choosing-harness-beats-building-in-house>

Cost Reporting for Chargeback and Showback: Why Choosing Harness Beats Building In-House

Deciding whether to adopt and implement a new product or build an in-house solution is a critical decision in any organization. Especially when the desired solution requires comprehensive features and deep functionality, pouring vast amounts of engineering time and resources into a homemade tool can be counterproductive. It's often more prudent to channel that investment and your team's time into bolstering your core business offerings.

So, why should you buy instead of build for cloud cost reporting? Building a chargeback and showback system for cloud costs may appear straightforward. However, a comprehensive cost reporting solution requires careful consideration of several aspects before deciding.

What is Chargeback and Showback?

The short answer is that both of these are processes that task departments using shared IT resources to understand and be accountable for the costs of using those resources.

- **Chargeback** is really what it sounds like - your department is getting a charge back to your budget for your cloud resource consumption, rather than the central infrastructure team bearing the responsibility for costs. This is generally driven by Finance.
- **Showback** is all about cost visibility - your department gets reports on what your cloud usage is, as well as the associated costs, but it isn't charged to your department.

It sounds straightforward, but for anyone who has seen a cloud bill, you know that it really isn't. Attributing costs to the right teams is a big challenge.

Managing Shared Costs

Leveraging simple Business Intelligence (BI) tools for cloud cost management may seem efficient at first, but they fall short when it comes to allocating shared costs across various departments, teams or applications across the organization. Without the ability to accurately attribute costs, chargeback and showback end up being best guess efforts from a centralized reporting team.

Harness CCM's Cost Categories offers a comprehensive solution for cost attribution, and is adept at handling intricate cost allocation tasks including multiple strategies to allocate shared costs transparently. With Cost Categories, you can quickly and easily create chargeback and showback reports with a high degree of accuracy.

Associating Costs with Recommendations

Once you've begun to accurately report and chargeback/showback cost per department, the obvious next step is taking initiative to reduce those costs. An integral part of efficiently managing cloud costs is right-sizing over-provisioned resources and cleaning up unused cloud resources. While building an in-house solution might provide raw cost data, effectively linking these costs with right-sizing recommendations does not come out of the box with a vanilla BI tool.

Harness CCM doesn't just tell you what you're spending, it also provides actionable recommendations on how to optimize cloud resources. By associating cloud costs with optimization recommendations, businesses can identify wastage, make informed adjustments, and realize significant savings while ensuring optimal performance.

Associating Costs with Cost Anomalies

Cost anomalies can indicate underlying issues or inefficiencies, and waiting for monthly chargeback/showback reports could result in surprise charges of thousands of dollars of surprise cloud spend. Detecting these anomalies is just the first step; understanding their financial implications is crucial for effective cloud cost management. With an in-house solution, this association can be a manual, time-

consuming review process.Â

In contrast, Harness CCM not only provides cost anomaly detection out-of-the-box but also provides a clear view of their financial impact as they occur. This enables businesses to quickly address and rectify costly unplanned spikes in cloud consumption, ensuring both financial prudence and operational efficiency.

Budgeting and Forecasting Management

By definition, chargeback and showback reports show you what happened in the past. Planning for the future is imperative for efficient cloud cost management. Unfortunately, in-house systems often lack the advanced capabilities to integrate budgeting and forecasting functionalities. Vanilla BI tools usually do not provide the ability to create, manage and track cloud cost budgets.Â

Harness CCM fills this gap, allowing teams to set budgets, anticipate future costs, and make informed financial decisions. Additionally, CCM enables you to create group budgets or cascading budgets at multiple levels across your organization that are interrelated.

Dynamic Team and Project Management

In a fast-paced business environment, teams and projects shift, merge, or divide. Managing cloud costs for such dynamic entities requires a system that's just as agile. An in-house solution might struggle to keep up with the changing definitions of teams, projects, departments and business units - or whatever construct you want to use for your chargeback/showback cloud cost reporting.Â Â

Harness CCMâs Cost Categories offers the flexibility to manage the definitions of these constructs across cost reports and dashboards for ever-evolving teams and projects effortlessly. This ensures that your chargeback/showback reports are fresh and accurate over time.Â

Tracking Resource Utilization Metrics

Beyond just costs, understanding resource utilization metrics like CPU usage can offer insights into efficiency and areas of potential optimization. Harness CCM shines here, offering an integrated view of both costs and critical utilization metrics that leverage cloud provider APIs out-of-the-box, which a DIY approach or standalone BI tool could struggle to incorporate and maintain. This can be particularly true with the ever-changing nature of cloud provider APIs and integrations.

Management of updates to the CUR and Billing Export

Crafting a system that dynamically detects, understands and ingests new dimensions in the Cost and Usage Report (CUR) or billing export of AWS, Azure or GCP, when required, may not always be very straightforward. But itâs essential for accurate chargeback and showback reporting. If your team doesnât keep up with these changes, your reports can lose accuracy quickly.

This goes beyond a simple one-time configuration given constant changes being introduced. This would demand constant updates and attention. Opting for Harness CCM eliminates this ongoing manual effort, ensuring that such changes are seamlessly integrated without the fuss.

Performance and Scalability Concerns

As your business grows, so does the data you're processing. Building an in-house solution means constantly addressing performance bottlenecks and scalability issues as reports expand. Harness CCM is designed with scalability in mind, effortlessly adapting to increasing data loads and ensuring consistent performance even as your reports grow in size and complexity.

Conclusion

Together, these reasons strongly suggest that investing in Harness CCM offers tangible benefits over the long, arduous journey of building and maintaining an in-house cloud cost management solution.

Looking for an easy, out-of-the-box solution for chargeback and showback of your company's cloud costs? **Book a demo** of Harness CCM or **sign up for free**.

â

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/demystifying-trunk-based-development>

Demystifying Trunk-Based Development

What is Trunk-Based Development?

Trunk-based development or TBD is one of a few different branching strategies that your team, or even you can use to bring features in quickly to the main branch of your workflow. When we talk or think about branching strategies, the first one that comes to mind is the GitFlow. GitFlow definitely has its place in the creation of software.

â

In order for your team to run like a well oiled machine, having a branching strategy at the forefront of your development practices will be key to getting code quickly out the door. All branching strategies can achieve that, but TBD, will give you that extra leverage without merge hell. We all love merge hell don't we lol?

â

By now I am sure we have all heard the term "merge hell" and one of the catalysts for that is what I call the "rebase race". Hurrying up to rebase changes that are pulled in from main in order to get your changes merged in, and fingers crossed nothing breaks. We have ALL been there. It isn't fun. Another benefit of TBD is avoiding merge hell with small commits to the main branch.

Breaking TBD Down

At the core of trunk-based development you have a main branch or trunk. The reason we call it a trunk is because it is very much like a tree. A tree has the main trunk where all branches and roots come from. In varying branching strategies you are creating off shoots or branches from that main trunk.

â

Nowadays, the trunk is called main, at least in the git world. I have worked in other VCS's (version control systems) like SVN and ClearCase where the main branch was called trunk. I am showing my age here. With those systems, once someone had control of the trunk, it was locked. You as an engineer couldn't push or pull any changes. Sometimes, it made for a horrible time when that developer forgot they locked it. No one on the team could get features merged.

â

In GitFlow, the branching strategy 9 times out of 10 has long lasting feature branches that go on until a feature or bug fix is complete, a PR is submitted, then after a successful rebase, the code is merged. This can create a few different problems, but the biggest one being "yep you guessed it, merge hell". As other developers work on their features and fixes, their code is getting rebased and oftentimes,

if the team is committing code frequently there are changes that need to be fixed.

â

TBD differs in that you are creating small, short lived branches that once approved, get merged into the main branch. You keep the feature or bug fix as close to the main branch as possible so the risk of merge hell is almost non-existent. Adopting a practice like TBD is an integral part in creating a culture of CI (Continuous Integration).

How to get started with trunk-based development?

Getting started with TBD isn't as painful as you might think. Yes, there are versions of TBD where you work directly in main and only in main. I would advise against doing that. Before you ever start a process when it comes to the SDLC, best practices, and architecture decisions, make sure that you discuss it with your team. Having your team's input will benefit everyone in the long run.

â

For newer projects do an initial commit to the main branch once your structure is set up and you have added all your directories and files.

â

Once you verify using git status that all changes have been added to the working directory, proceed with your initial commit.

â

â

Verify that your changes are committed and ready to be pushed using **git status** once again. Git status is one of the most important commands you should be running when working with a project. It will give you a piece of mind in knowing where your directories and files are in the flow.

â

From here, you would branch out as you normally would. Depending on how your team best practices are, branch names might look different. I always liked using the `last name/task # + name of feature/bug` style. What that looks like is this:

â

â

Now, you are working in your feature branch. This branch isn't going to be long lived. Once you are done with your feature, that can even be behind a feature flag, make sure to submit a PR and get it merged in!

â

â

As you can see from the image above, this is what your branches should look like when using TBD.

Trunk-based development best practices

There are a lot of best practices and pros when using trunk-based development as your branching strategy. Below I have listed out some of the important ones.

â

- Merge hell becomes almost non-existent.
- Pulling in code from main should be 1:1 with your branch, or very close to it
- Removing the rebase race from long lived branches being merged in with older code
- Short lived branches
- The main branch is always the source of truth

â

Wrapping Up

TBD is a very efficient way to keep code close to main. If you are looking for a way to keep your projects in order without the risk of over-the-top merge conflicts, TBD is surely the way to go. Pairing feature flags with trunk-based development removes the worry of cutting a release with unfinished code. If the code is hidden behind a flag, there is no impact to the release of the newest features that

should go out. Next time you are spinning up a new project, give trunk-based development a try!

â

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/why-platform-engineers-trust-harness-gartner>

Why Platform Engineers Trust Harness

In the ever-evolving world of DevOps, platform engineers constantly seek tools that can scale with their needs. We at Harness believe in delivering an excellent developer experience where best practices can be shared and governance can be effective without being burdensome.

We feel that this strategy has been reaffirmed - **Harness ranked 1st** in the Gartner report for the Platform Engineering use case in the 2023 Critical Capabilities for DevOps Platforms report.

The Report

The research allows buyers of DevOps platforms to rank the ability of 12 vendors to provide 12 critical capabilities across four common use cases." The report defines Platform Engineering as "Ability to target a broad set of cloud-native environments and an overall strong outer-loop capability supporting progressive release, containers and package management." This makes sense to us at Harness. Platform teams are scaling new workloads, cloud infrastructure and consumable software delivery.

We are proud to have received the highest scores in the Platform Engineering use case, but we believe that's only part of why Platform teams turn to Harness to improve developer experience and effectiveness.

Why Platform Engineers Love Harness

Platform engineering teams are typically challenged with scale. They need to support a large developer community with a relatively small team. They know that handcrafting pipelines, infrastructure and governance for each team just won't work.

Harness acts as a force multiplier for these teams, having been architected to scale configuration.â

- **Templated configuration** enables platform teams to capture automation that works and reuse it by reference. As they learn and improve the templates, pipelines based on those templates get those changes. Templates ensure that adopting best practices is the easiest way for application teams to proceed, driving standardization and jump starting productivity.
- **Policy as code** techniques help platform teams balance freedom for the individual application and stream-aligned teams with the need to apply governance policies consistently. With Open Policy Agent (OPA) policies, they can establish rules like âFeature Flags must be enabled in QA before Prodâ while still permitting platform consumers to edit their pipelines.
- **Easy to configure:** Whether crafting a new pipeline or customizing an existing template, Harness strives to simplify the experience. With Harness, a platform engineer can get more done with less hassle.Â
- **Modular and Coherent:** With a move towards tool consolidation, Harness provides a coherent all-in-one platform. At the same time, it doesnât require total adoption. If an existing tool is doing well, you can skip that Harness module and Harness will integrate with what you have.
- **Multi-cloud native:** Platform teams are often aligned with cloud-native workloads, and Harnessâs deep integration with cloud-native platforms and approaches makes it a great fit.

Doubling Down on Platform Engineering

â

This analysis for this report predates the release of two new Harness modules that focus on the platform engineer:

â

â

Looking Forward

â

The success of platform engineering teams reduces burnout and increases developer effectiveness. This is good stuff. At Harness, we look forward to continuing to partner with and deliver for the Platform engineers.Â

â

Want to give it a try? Sign up for free.

â

â

Gartner, âCritical Capabilities for DevOps Platformsâ, Published 4 September 2023. By Thomas Murphy, Manjunath Bhat, Joachim Herschmann, Daniel Betts, Chris Saunderson, Hassan Ennaciri, Bill Holz, Peter Hyde

â

GARTNER is a registered trademark and service mark of Gartner, Inc. and/or its affiliates in the U.S. and internationally and is used herein with permission. All rights reserved. Gartner does not endorse any vendor, product or service depicted in its research publications, and does not advise technology users to select only those vendors with the highest ratings or other designation. Gartner research publications consist of the opinions of Gartnerâs research organization and should not be construed as statements of fact. Gartner disclaims all warranties, expressed or implied, with respect to this research, including any warranties of merchantability or fitness for a particular purpose.

â

â

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/trunk-based-vs-feature-based-development>

Trunk-Based vs. Feature-Based Development

Trunk-Based vs. Feature-Based Development

Choosing the right branching strategy for your software development project can significantly impact the efficiency and quality of your projects. Two popular methodologies, trunk-based and feature-based development (often referred to as GitFlow), offer unique approaches. With the rise of DevOps and Continuous Integration/Continuous Deployment (CI/CD), understanding these methodologies becomes even more crucial to your software engineering projects. Let's dive in.

What is Feature-Based development?

Feature-based Development, also known as GitFlow, is a branching model where new features, enhancements, or bug fixes are developed in separate long-lived branches. This structured approach ensures that the main codebase remains stable while new features are being developed. This approach can be useful, for example, when you need to maintain multiple software versions for a long duration. In such a case, the feature-based model allows you to maintain a long-lived branch for each version and manage their code separately and safely.

A common process for feature-based development, at a high level, looks like this:

- **Branching off to work on Feature Branches** - Developers create separate branches from the `develop` branch when they start working on new features. These distinct branches, known as feature branches, let developers work independently until their code is stable and functional.
- **Merge feature branch back to `develop`** - Once a feature branch is finalized, it's integrated back into the `develop` branch.
- **Create release branch for testing (release candidate)** - When the software is set for a release, and all finalized features are part of the `develop` branch, a new release branch is formed from it.
- **Combine release changes** - After ensuring the release branch is free from issues and ready for live deployment, it's combined with both the `main` branch, for deployment purposes, and the `develop` branch to include any bug resolutions made during the release's finalization.

Benefits of feature-based development:

- **Isolation:** Developers can work on features without affecting the main codebase.
- **Flexibility:** Allows parallel development of multiple features at scale, minimizing pull request number and avoiding merge hell .
- **Clear Release Management:** Features are merged into the main branch when ready for release.
- **The main branch remains stable**, ideal for CI/CD pipelines targeting stable releases.

Challenges of feature-based development:

- **Complex Merges:** Longer-lived branches can lead to intricate merge conflicts, due to large scope of changes in the feature branch.
- **Potential for Stale Code:** Feature branches can become outdated if not regularly merged with the main branch.
- **Delayed Integration:** Integration delays can pose challenges, especially in fast-paced environments.

What is Trunk-Based development?

Trunk-based development is a very popular branching strategy where all members of the developers' team manage their source code in the same branch, aka the "trunk" (typically named "master" or "main" branches). In this model developers either work directly in the same branch or short-lived branches. The emphasis is on integrating code frequently, ensuring a consistent and up-to-date codebase, and making sure the application remains in a deployable state with the latest features and fixes in place.

Benefits of trunk-based development:

- **Rapid Integration:** Frequent integrations mean errors are detected and fixed quickly.
- **Simpler Merges:** Short-lived branches reduce merge conflict complexity and enable development teams to collaborate effectively and remain in sync with each other.
- **Consistent Codebase:** Everyone works off the latest codebase version.
- **Promotes Continuous Integration:** Naturally aligns with CI/CD practices, ensuring that code is always automatically tested, verified, and in a deployable state.

Challenges of trunk-based development:

- **Requires Discipline:** Developers need to integrate changes frequently.
- **Potential for Broken Builds:** Without proper management, the trunk can become unstable.
- **Depends on Automated Testing:** Rapid changes necessitate robust testing to prevent regressions.

Trunk-Based development and CI/CD

Continuous integration (CI) and Continuous Deployment (CD) represent the forefront of modern software development practices. Their primary goal is to accelerate both the development and delivery of software, ensuring that quality remains uncompromised.

Trunk-based development workflows align well with the CI/CD as at their core, they both emphasize rapid, frequent, and reliable software development and delivery. Using continuous integration, developers combine trunk-based development with automated tests that execute after every commit or pull request to the main branch, guaranteeing that issues (security issues, defects) are found as possible in the development process. It allows quick iteration and keeps code ready to deploy. Continuous Deployment (CD) ensures that after passing all tests, changes are automatically deployed to production. This seamless transition, supported by trunk-based development, enables quicker releases and immediate user feedback.

Feature Flags and Trunk-Based development

Feature flags (FF), also known as toggles, are a powerful tool in modern software development. They provide developers with the ability to turn features on or off without changing the code, allowing for safer deployments and A/B testing. This flexibility means that features can be tested in production, rolled out to specific user groups, or disabled quickly if issues arise.

Feature Flags enhance trunk-based development by allowing developers to encapsulate new features within dormant code segments. This means that instead of branching off to develop new features, code can be committed directly to the main branch, but kept inactive under the feature flag until it's ready.

This approach fosters a culture of incremental updates. Developers can introduce a feature flag in an initial commit and then progressively build upon the feature in subsequent commits. The main branch remains stable, and the feature can be tested, refined, and eventually activated when it meets the desired criteria.

By integrating feature flags, teams can maintain a steady development pace while ensuring that new additions are introduced seamlessly and safely. They can continuously deploy all changes, while progressively testing and releasing the features when ready.

When to use which branching strategy?

The choice between trunk-based development and GitFlow largely depends on the specific needs of your project, the team's composition, and the desired management style.

For projects that demand rapid iterations and frequent releases, trunk-based development shines. It's particularly effective when you're launching new products or pivoting an existing application in a fresh direction. This approach grants developers considerable autonomy, reflecting trust in their expertise. With unrestricted access to the source code, it's crucial to have a team of seasoned developers who can harness this freedom responsibly. The streamlined processes inherent in trunk-based development can significantly boost software development speed.

Conversely, GitFlow is tailored for larger projects characterized by distinct features or those operating on scheduled release cycles. It's an excellent fit for open-source projects, large enterprises, or companies with established products. It's especially beneficial when the team consists of junior developers or when there's a mix of experience levels, as GitFlow ensures strict access control. This rigorous oversight can be a double-edged sword: while it guarantees thorough review and quality, it might sometimes lead to micromanagement or slow down the development pace.

In terms of team size and dynamics, trunk-based development is the go-to for smaller teams that thrive on streamlined communication. In contrast, GitFlow is advantageous for larger teams, especially when features are being developed by distinct units or sub-teams.

When it comes to release management, if your aim leans towards continuous delivery or frequent releases, trunk-based development is your best bet. However, for projects with scheduled releases or features that require extensive review before launch, GitFlow offers a structured and controlled environment.

In essence, both branching strategies have their merits. Trunk-based development expresses confidence in the team's abilities, making it ideal for experienced developers. GitFlow, with its structured approach, provides a safety net, especially when working with a diverse team. Understanding the nuances of each will empower you to select the workflow that aligns seamlessly with your project's objectives and team dynamics.

Conclusion

Choosing between trunk-based and feature-based development largely depends on the specific needs and dynamics of your project and team. While trunk-based development offers rapid iterations and aligns seamlessly with CI/CD practices, feature-based development provides flexibility and isolation, especially beneficial for larger projects with multiple parallel developments. Ultimately, understanding the nuances of each approach and their implications in a CI/CD context will guide teams in making informed decisions, optimizing both development and delivery processes for maximum efficiency and quality.

Try Harness

Try Harness to see how it can transform your software development workflow, sign up for a free plan or request a demo today. Sign up for a free plan or request a demo today.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/using-feature-flags-for-effective-incident-management>

Using Feature Flags for Effective Incident Management

Whether you call it an incident, outage, surprise, or unplanned work, your application isn't working as expected, and you need to deal with the problem. That's where dependable incident management comes in. You never want to be responding to an incident without a plan or prior systems set in place. You may not always be able to stop an incident, but you can always dampen and remediate its effects with good planning.

How Feature Flags Can Help for Incident Management

Feature flags can help you improve your incident management process in a number of ways:

- **Kill switches and circuit breakers:** You can use feature flags to disable features or services quickly and easily without having to redeploy your code. This can be helpful if you need to quickly isolate the source of an incident or prevent a problem from spreading. This can prevent a feature from causing problems for your users.
- **Throttling:** You can use feature flags to throttle traffic to certain features or services. This can be helpful if you're experiencing performance issues or if you need to prevent a feature from being overloaded. This can help to improve the performance of the service for more critical or demanding users.
- **Debugging:** Feature flags can help you debug incidents by providing you with insights into which features or services were recently enabled or changed. This can help you quickly identify the root cause of an incident. If it's not obvious at first, you can also enable and disable different features and services to see how they impact performance and study the affected systems.
- **Automation:** You can use feature flags to automate tasks related to incident management, such as adjusting logging levels, rate-limiting services, or disabling features. This can help you reduce the time it takes to resolve incidents.

Tips for Effective Incident Management

Here are some tips for using feature flags for incident management:

- **Create a flag management plan.** This plan should define how you will create, manage, and use flags for incident management. It should also include a process for reviewing and updating your flagging strategy on a regular basis. We've seen this be very effective for teams using feature flags at Harness. Remember: a little extra time spent on this before can save a lot of time (and money) after.
- **Use flags to implement operational resilience practice.** Operational resilience practices are designed to help you minimize the impact of incidents on your users. For example, you can use flags to implement circuit breakers or to throttle traffic to certain features during an incident.
- **Monitor your flags.** It is important to monitor your flags to ensure that they are being used as intended. You should also monitor the impact of your flags on your application and users.
- **Testing your incident response plan.** Use feature flags to simulate different types of incidents during your incident response testing. This will help you to identify any areas where you need to improve your plan. If, however, an actual incident occurs, treat it as another (harsher) test of your incident response plan and make sure to apply all lessons learned during it. Even if you had an effective incident response, it's important to acknowledge what went right.

By following these tips, you can use feature flags to improve your incident management process and reduce the time it takes to resolve incidents.

Benefits of Using Harness for Incident Management

In addition to the general benefits listed above, Harness Feature Flags also offer a number of differentiators that can be particularly helpful for incident management. Here are some specific examples of how you can use Harness Feature Flags with OPA, Pipelines, and Git Sync for incident management:

- **Use OPA to enforce policies on feature flags:** You can use OPA to enforce policies on feature flags, such as requiring that certain approvals be obtained before a feature flag can be enabled. This can help to prevent accidental or unauthorized changes to feature flags during an incident.
- **Use Pipelines to automate the process of enabling and disabling feature flags:** You can use Harness Pipelines to automate the process of enabling and disabling feature flags. This can help to streamline your incident management process and reduce the risk of human error. For example, you could create a pipeline that enforces that a feature be tested in non-production before the feature is enabled in production.
- **Use Git Sync to keep your feature flags in sync with your Git repository:** You can use Harness Git Sync to keep your feature flags in sync with your Git repository. This can help to ensure that your feature flags are always up-to-date and that you have a complete audit trail of all changes. For example, you could use Git Sync to create a branch for each incident fix and then use that branch to track all changes to feature flags related to that incident.

Beyond incident management, here are some other key benefits of using Harness Feature Flags for your project:

- **Automated lifecycle management :** Harness Feature Flags helps you manage your flag tech-debt by detecting potentially stale or deprecated flags and automatically removing them if needed.
- **Integrated with CI/CD:** Since Harness covers a range of software delivery products, you can easily integrate your feature flags into CI/CD as a unified pipeline that helps maintain the flags as part of the software development lifecycle.

Conclusion

By using Harness Feature Flags for incident management, you can improve the resilience of your application and reduce the impact of incidents on your users. Incident management starts before you write your first line of code and continues after your code is in production. By following the tips above, you can effectively contain the blast radius of any unexpected incidents and restore the user experience.

If you want to use Harness Feature Flags for incident management, get started for free now!

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/architecting-for-cost-savings-at-tyler-technologies>

Architecting for Cost Savings at Tyler Technologies

This is a guest post by Harness customer Chris Camire - Senior Manager, Technical Services, Tyler Technologies

â

When you host cloud infrastructure for your clients across the globe, cloud costs can easily spiral out of control, especially when client non-production environments are fully operational every hour of the day. At Tyler Technologies, that was certainly true for our Enterprise Permitting & Licensing solution. Tyler Technologies is the largest software provider in the U.S. that is solely focused on the public sector; its Enterprise Permitting & Licensing solution is used by government agencies to automate and streamline their community development and business management operations.

We originally tried to tackle this problem with AWS CloudWatch, using alarms to power down non-production infrastructure outside of working hours. We knew this wasn't a long-term solution; our clients often work late hours and need their non-production environments available on-demand. Inversely, we also had the challenge that some environments weren't used daily, or even weekly, but could be needed on a moment's notice.

Ultimately, we decided to adopt Harness Cloud AutoStopping® to help us get control of our idle cloud resources. It's been a massive success, we've gotten a great deal of benefit from it. You can read more about that in our case study that we did with Harness.Â

This blog doesn't focus on how we're using Cloud AutoStopping though, because we realized that before we created a single Cloud AutoStopping rule, we needed to take a good look at how our infrastructure was organized in order to really maximize cloud cost savings.

Organic Growth Leads to Infrastructure Sprawl

Each of our clients is provided with a set of non-production environments, on non-production infrastructure, organized into âpodsâ. These environments are used by our clients to test our software before deploying into their production environments. Given the regulatory nature of the permitting and licensing services we provide to government agencies, ensuring that new features or bug fixes are

fully tested and vetted before pushing to production is critical.Â

As our client base grew over the years, we continued to build additional pods to meet demand, and ensure clients had secure, reliable environments to test in. What we didnât do was create a strategy for how we organized our clients across that cloud infrastructure.Â

What we ended up with was clients across very different time zones being hosted in the same pod. Which meant that, from one end of the country to the other, there would always be a client in that pod that needed access to their environments.Â

Being Intentional About Organizing Shared Infrastructure

As we thought about reorganizing our infrastructure for cloud cost savings, we started with one key goal: organize in such a way that Cloud AutoStopping could stop the instances in our pods as much as possible. Weâre taking a 2-step approach to this, first by client time zone, and then by client activity.Â

Organizing by Time Zone

Categorizing our clients by time zone was the obvious first choice, since it would group clients that start and stop their workdays at roughly the same time. Given that our client base consists solely of public agencies, we already knew which states and time zones those agencies worked in. We got to work and started migrating the underlying applications into new âpodsâ that were designated by timezone.

This new organization has definitely helped us optimize the efficiency of our pods; they are now generally fully idle at the same time, enabling Cloud AutoStopping to power down these instances until they are needed again. When a client needs their environment, they access it just as they normally would, Cloud AutoStopping detects the incoming traffic, auto-starts the instances, and the client is off and running.

The results have been amazing since we started the process 6 months ago. Our cloud cost savings have increased exponentially as weâve configured Cloud AutoStopping on more pods; initially saving us \$15K to \$20K a month and now passing the milestone of saving \$100K last month alone.

Organizing by Activity

When we started the process 6 months ago, trying to organize our clients by their activity level was nowhere on our radar. We didnât have any real visibility into that level of our clients' usage. But as weâve continued to roll out Cloud AutoStopping across our infrastructure, it has become clear just how much some clients are (and arenât) using their environments based on the idle / active times we see in the Harness Cloud AutoStopping console.Â

The processes and procedures that government agencies have in place to test new software patches or releases can vary greatly; some are testing or training every day - others much less, monthly or even quarterly. That means we have environments that are idle for days or weeks at a time. We realized that if we group these environments together, we can get to the point where pods could be powered down for much longer periods of time.Â

Weâre working with Harness to get a little more intelligence here to help us be more efficient with this categorization, but weâve already started the reorganization with the data we have in hand today.Â

Whatâs Next?

For now, Iâm focusing on getting as close to 100% coverage for Cloud AutoStopping rules as I can, since it has such a massive, ongoing positive impact on our bottom line. Then Iâll start exploring other features more in-depth like Cloud Asset Governance and implementing recommendations.Â

Itâs been a great journey for us so far, and you can read more about it in our case study.Â

Editors Note: This is one of a new series of blogs that are focused on Harness Cloud Cost Management users are âArchitecting for Cost Savingsâ, highlighting their experiences where cloud architectural decisions and changes have positively impacted cloud costs.Â

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/road-to-kubecon-2023>

Road to KubeCon 2023 - Join Harness at KubeCon NA 2023

As the weather starts to get colder in the northern hemisphere, that means KubeCon + CloudNativeCon North America is around the corner. KubeCon + CloudNativeCon North America this year will take place in Chicago in early November. In its eighth year since the inaugural KubeCon in 2015, KubeCon now coincides with CloudNativeCon and is now flagship conference gathers adopters of Cloud Native Compute Foundation [CNCF] projects. These events are now held yearly in different regions across the globe.Â

Harness will be back at KubeCon NA this year and also participating in Co-Located events [the old Day Zero event format] as we have expanded our participation in the Linux Foundation. As the cloud native ecosystem continues to evolve, the CNCF has been an excellent custodian for many projects and you can expect these projects and practitioners to be represented.Â

What Can You Expect This Year?

The cloud native community has done tremendous work ushering in the cloud native paradigmÂ e.g scale, robustness, portability, and scalability to anyone who needs those pillars with open source and cloud technologies.Â

We have shared our project/conference predictions with VMBlog which we delve into greater detail on analysis of the talks this year and leverage DevStats for a data driven approach. There has been some consolidation in the CNCF as some cards exited / did not maintain their maturity level. For example in 2020, there were over 1500 cards in the CNCF Landscape. As of this week in October 2023, there are 1246. As the landscape continues to evolve, Harness will be your partner in cloud native technologies and paradigms as they continue to evolve.Â

Harness, Your Cloud Native Partner

The cloud native space certainly moves quickly. Harness has been a steward to the cloud native communities for several years and an active participant in the Linux Foundation. This year Harness is dedicating more full time resources to contributing back to the cloud native ecosystem.Â

Most recently, with what has been going on in the Terraform Ecosystem, Harness has pledged to be a charter member and has dedicated full time resources to OpenTofu which is a Linux Foundation project.Â

Harness has contributions across the CNCF. As a core member of the Litmus Chaos Project which is an incubating maturity project inside the CNCF, Harness leads contributions to this project. Litmus Chaos supportsÂ the open source Chaos Engineering ecosystem. Harness has also embraced the GitOps paradigm by starting to contribute back to upstream Argo.Â

Additionally Harness participates in the FinOps Foundation and Continuous Delivery Foundation which we have been supporting with thought leadership giving our expertise back to those communities.Â

If you have not already, make sure to sign up for the Harness Platform as we head to Chicago. We are happy to continue to support the ecosystem that allows us to help build the next generation of scale together.Â

See You in Chicago

Harness will have booth presences at two co-located events, ArgoCon and BackstageCon on November 6th and have a booth [B15] and the main event. Our very own Himanhsu will be talking at BackstageCon this year about looking beyond two core use cases/plugins in

Backstage.Â

Make sure to stop by the Harness Booth at KubeCon [B15]. We are big fans here at Harness of K9s and K8s. Stop by for a chance to get a harness for your favorite K9, from Harness.

â

The event this year in Chicago will surely be a great one, in its eighth rendition as it gathers adopters and technologists. We champion for all of those who consume, maintain, and are excited to learn about cloud native technologies, to attend. In the meantime execute a Harness Tutorial to ramp up your cloud native delivery skills with ease.

Cheers,

-Ravi

â

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/become-harness-certified-in-ci-and-cd-gitops-at-any-level>

Become Harness Certified in CI and CD & GitOps at Any Level

As easy as âgetting your ideas to productionâ might seem in principle, there can be a lot of order of operations and nuance to do so. Code that is human readable needs to be placed in a distribution that machines understand and then those distributions need to be deployed. All while taking into account infrastructure/application infrastructure and also done in a safe manner. Core CI/CD & GitOps skillsets are needed to accomplish this. Today, we are pleased to announce that no matter where you are in your CI/CD & GitOps journey, Harness can help take you to the next level with multiple levels of certifications.Â

Harness now offers three levels of certifications in both our CI and CD & GitOps modules. Building upon the certification framework we have started this year at the developer level and extending our administrator level, we have introduced an architect level as a capstone level as our most rigorous exams.

Which Certification is Right For You?

With the three levels, the certifications do build off of each other in a series of increasing levels of difficulty. The certification exams are designed to be taken at any time asynchronously.

Developer

The developer level certifications are designed to show mastery of usage of a module. The developer level focuses on the Free Harness Tier and the exam itself is free to take. The exam is a question and answer format and should take about 90 minutes to complete. For example in the CD and GitOps module, understanding how to deploy a Kubernetes Manifest in Harness would be a key point.

Administrator

The administrator level certifications are designed to show mastery of administration of a module. The administrator level focuses on the Team/Enterprise Harness Tiers. The exam is both a question and answer format and a practical hands on portion. The exam is designed to be three hours in duration for both portions. For example in the CD and GitOps module, understanding templating and how to validate your deployments would be key points.

Architect

The architect level certifications are designed to show strategic understanding and experience of a module. The architect level focuses on the Team/Enterprise Harness tiers. The exam is both a question and answer format and a practical hands on portion. The exam is designed to be three hours in duration for both portions. For example in the CD and GitOps module, using custom deployment templates and organizing your organization in Harness would be key points.

Hands-on with Administrator and Architect Levels

Both the administrator and architect level exams have a hands-on component to them. After the knowledge based exam, you will be presented with several hands on questions to solve in the specific Harness Module you are certifying in. There is no need to bring your own infrastructure or Harness Account as all of the infrastructure you need is available via a cloud shell in your web browser. You will be provided with an ephemeral Harness Enterprise account for the exam duration.

We have further instructions for the hands-on portion if you would like to further review.

Preparing and Taking Your Exam

The learning material is hosted on Harness Developer Hub. Each certification has a learning path to review material, for example the below CD & GitOps Architect level.

Registration

Registering for a certification is simple. Head to Harness Developer Hub, select which certification you would like to take, scroll down through the study guide, then click Register for Exam. You have the ability to take the exam at any time on your schedule. Once you have successfully completed the exam, a Credly Badge will be issued to you in a few days. Coverage on the CI and CD & GitOps modules are just the start for us.

Harness Certifications: Continue Learning

We continue to plan to roll out certifications at multiple levels covering every Harness Module. Even if you do not become certified, check out the learning paths that are open and available to increase your skill regardless of what level you are at.Â

Cheers!

-Harness Product Education Engineering Team

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/growing-the-harness-developer-hub>

Growing the Harness Developer Hub

Since announcing the general availability of the Harness Developer Hub, our writers have continued to publish new content and improve existing content. These updates, based on customer feedback, are aimed to enhance clarity, improve accuracy, and create a structure that better follows the user journey.

We want to take this opportunity to inform you about just some of these exciting changes.

Get Started with Tutorials

Our tutorials are a great place to get started with hands-on learning. We continue to add more tutorials and guides covering specific use cases for Harness modules. Recent additions include:

- Harness CI guides for various languages (C#, Go, Java, Node.js, Python, Ruby, and iOS/macOS) and publishing anything to the Artifacts tab.
- How to use CD to automatically deploy Kubernetes and GitOps services, AWS Lambda and Google Cloud Serverless functions, and traditional applications on cloud or physical data center VMs.
- How to integrate STO with GitLab and scan a Node.js app.Â
- Feature Flag best practices and how to use Feature Flags in trunk-based development.
- How to manage secrets in the Harness Platform.
- How to configure self-managed external databases for Harness Self-Managed Enterprise Edition.
- CCM tutorials on AutoStopping and optimizing cloud costs for Kubernetes.

Learn more with Documentation

In addition to new documentation sets for Continuous Error Tracking and the Internal Developer Portal, we've improved and expanded content for our existing modules, the Harness Platform, and the Harness Self-Managed Enterprise Edition.

â

Continuous Integration

The Harness CI documentation now has more introductory and summary information, such as the CI pipeline creation overview, to help you understand your CI pipelines at a high level. As you learn to use CI, the documentation moves from introductory to advanced configurations.

We've eliminated redundancy by combining reference topics with their âparentâ sections or pages. This reduces the number of pages you need to visit to find the information you need. For example, the Upload Artifacts to GCS page includes helpful information about using this step in a CI pipeline as well as reference information for this step's settings.

Almost all of the existing pages have been refreshed, and we've added several new topics. Here are some highlights:

- More migration guides.
- Addressed FAQs around code coverage and formatting test reports, with examples for different languages.

- Refreshed and expanded documentation on caching and artifacts management.
- A feature compatibility matrix comparing feature support across infrastructure types.
- Support for extensibility. You can create your own custom plugins, and you can use GHA, Bitrise, and Drone plugins in your pipelines.
- Check your license usage and manage your CI subscription directly in Harness.
- Troubleshoot with AIDA and debug with SSH.

Continuous Delivery & GitOps

The Continuous Delivery & GitOps documentation has been reorganized to provide a clearer path to onboarding and extending with Harness CD & GitOps. You can:

- Get started by creating your first CD pipeline (through YAML, API, the Harness Terraform Provider, or the Pipeline Studio's visual editor), understanding modeling in Harness, learning about services and environments, and integrating CD with other Harness modules, such as CI.
- Upgrade CD from Harness FirstGen to Harness NextGen. The FirstGen and NextGen CD parity matrix lists the differences between Harness CD FirstGen (FG) and NextGen (NG) capabilities.
- Deploy services on different platforms and learn about cross-platform functionality.

Cloud Cost Management

The Harness CCM documentation has been reorganized into three main categories: cost reporting, cost optimization, and cost governance.

We've added more detail about what's supported, and we've added several new topics about Asset Governance, including Asset Governance with AIDA and Asset Governance RBAC. Asset Governance is one of CCM's key features, providing users with essential information on managing cloud assets using a Governance-as-Code approach that includes real-time enforcement and auto-remediation capabilities.

We've also added new topics about optimizing costs by applying recommendations for Azure VMs, using currency standardization for consistent cost analysis across CCM, and AutoStopping proxy load balancers.

Service Reliability Management

The Harness Service Reliability Management (SRM) documentation has undergone a comprehensive reorganization, designed to enhance the user's learning journey. The restructured documentation provides a clear roadmap for users to progress from foundational concepts to advanced topics. We've refreshed existing topics and added several new topics.

- Get started with SRM introduces SRM's key features, including creating your first Service Level Objective (SLO). From there, explore and configure SLOs, monitored services, and notifications.
- Browse comprehensive catalogs of health sources and change sources supported by Harness SRM.
- The change impact analysis section explains how to use dashboards to correlate change events with service performance so you can make informed decisions and maintain service reliability.

Security Testing Orchestration

For STO, we've added definitions of key STO concepts like targets, baselines, variants, severity scores and levels, exemptions ("ignore rules"), and failing pipelines by severity.Â

We've added several new workflows describing how to use GitHub Actions and Plugin steps to run scans, ingest SARIF data, use AIDA with STO, download scan images from private registries, add artifacts to STO pipelines, and stop pipelines automatically based on governance policies and scan results.

The scanner references have been revised and expanded, including scan steps that support scanner templates and information about scanner support by scan mode, binaries in STO container images, and Docker-in-Docker root access requirements.

Other highlights

In the **Platform** documentation, we've added information about AIDA and refreshed content about code repo connectors, getting started with Harness APIs, and access control.

For **Feature Flags**, we've added instructions for using Feature Flags with monitored services and Jira.

For **Chaos Engineering**, you can learn how to integrate CE with other Harness modules. We've also added information about managing chaos hubs, running a GameDay, and how we calculate resilience scores for experiments.

For the **Harness Self-Managed Enterprise Edition**, we've completely reorganized the documentation and added topics about what's supported, how to install in air-gapped environments, and how to manage Feature Flags. On the HDH, you can also find information about using CCM on the Harness Self-Managed Enterprise Edition.

Become a Harness-Certified Expert

The only constant in technology is change. Since the new series was first released in March of 2023, the Harness-Certified Expert Certifications have expanded. There are now more certifications available, and there are three levels of coverage for the Harness Continuous Delivery & GitOps module: Developer, Administrator, and Architect. The Administrator and Architect levels also include a hands-on portion to validate practical Harness skills.

â

You can sign up for a free Developer level certification or try one of the more advanced hands-on certifications today. Visit the HDH for all the latest developments across Harness.Â

Whatâs Next?

As Harness continues to expand and evolve, so will our documentation. We have plans for more tutorials, more YAML examples, and other new and improved content. Check in on the HDH regularly for the latest updates.

Did you know you can contribute to the HDH? Select **Edit this page** at the bottom of any HDH page to suggest changes for a single page. For larger changes, please review our Contributing guide.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/how-we-use-our-own-feature-flags-at-harness>

How We Use Our Own Feature Flags at Harness

Have you ever wondered what using Feature Flags at a company looks like? Well, we thought weâd just share how theyâre used here. At Harness, we donât just build products to enable our customers but also to strengthen our own teams. Before even launching Harness Feature Flags, different teams were stress-testing our product for their use-cases and were our true initial customers. To this day, teams like our Customer Success Management (CSM) team use Feature Flags daily for various purposes.

Early access to new features

The CSM team provides select customers early access to new features or improvements. This allows them to gather feedback and

identify potential issues before rolling out the feature to a broader audience. This also gives other customers confidence that the new features are field-tested when they are generally available (GA).

Rollbacks and mitigation

If a new feature causes unexpected issues or negative customer feedback, the CSM team can quickly roll back to a previous version or disable the problematic feature. This helps to minimize customer disruption and maintain a positive user experience.

Gradual feature rollouts

The CSM team gradually rolls out new features to a larger user base, starting with a small group of customers. This gives them valuable observability and time to monitor the performance of the feature and make adjustments as needed. They also communicate with customers about the release process and let them know that feature flags are used to gradually roll out new features. Companies may not always be sharing their backend workflows with customers, but when trying to create trust and reliability, it can be key. According to Ruchira Bajaj, a CSM Manager, âTelling the customers that we use Feature Flags to gradually roll out new features gives them confidence in our release processâ.

Customer-specific troubleshooting

If a customer encounters an issue or has unique requirements, the CSM team can use feature flags to turn specific functionality on or off per customer. Dealing with time sensitivity issues or asks from customers requires flexibility that can always be hard to have foresight on. For Ruchira and her team, âthis level of granularity can be helpful for troubleshooting and providing tailored solutionsâ in order to ensure customer satisfaction.

Customer testing

The CSM team works with several customers who have two instances of Harness. They first enable a new feature on the customer's development/sandbox instance and ensure that there will be no impact on their production deployments. Once the customers have tested the new feature, the CSM team enables it on their production instance of Harness. This allows customers to test new features in a non-production environment before enabling them in production, which helps to minimize the risk of disruption to live customers.

How Harness Feature Flags can enable you

Just as the CSM team takes advantage of them for their own work, here are some of the key benefits of using Harness Feature Flags for your project:

- **Automated rollout pipelines:** Harness Feature Flags allows you to create automated rollout pipelines that can be used to deploy new features to production in a controlled and predictable manner.
- **Governance:** Harness Feature Flags has the most in-depth governance tool withÂ Harness Policy as Code, powered by Open Policy Agent (OPA), providing a number of capabilities such as role-based access control, approval workflows, and audit logging.
- **Automated lifecycle management :** Harness Feature Flags helps you manage your flag tech-debt by detecting potentially stale or deprecated flags and automatically removing them if neededâ
- **Integrated with CI/CD:** Since Harness covers a range of software delivery products, you can easily integrate your feature flags into CI/CD as a unified pipeline that helps maintain the flags as part of the software development lifecycle.

Conclusion

Teams like Ruchiraâs and other development teams around Harness count on our product just as much as our customers do to ensure reliable feature releases, among other use cases. If you want to start enabling your team with Harness Feature Flags, get started for free now!

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/harnessing-next-gen-ai-for-secure-software-delivery>

Harnessing Next-Gen AI for Secure Software Delivery: A Look at Harness AIDA™

The Convergence of DevOps and GenAI

The DevOps ecosystem is continuously evolving, and the infusion of GenAI is the latest milestone in this journey. As the software delivery landscape expands, the role of GenAI becomes increasingly pivotal, offering efficient, secure, and innovative solutions.

GenAI, or Generative Artificial Intelligence, represents the next frontier in computing, where machines can perform intellectual tasks. By integrating GenAI into DevOps, we're not just automating processes but introducing intelligent decision-making into the software delivery lifecycle.

Harness AIDA™ and Harness Security Testing Orchestration: Beyond Traditional Software Delivery

Harness AIDA (AI Development Assistant) and Harness STO (Security Testing Orchestration) are not just tools; they are transformative solutions that redefine the boundaries of software delivery.Â

AIDA can automatically identify security vulnerabilities and generate code fixes. Trained on all publicly known Common Vulnerabilities and Exposures (CVEs) and Common Weakness Enumerations (CWEs), AIDA significantly accelerates vulnerability remediation, reducing developer effort by 50-75%. AIDA seamlessly integrates with the Harness Security Testing Orchestration module.

The combination of these tools ensures that software delivery is fast and secure, meeting modern enterprises' high standards.

Why GenAI Matters in DevOps

In today's rapidly evolving tech landscape, AI-powered tools are becoming indispensable for businesses striving for excellence. AIDA stands at the forefront of this transformation, addressing three critical pillars of modern software delivery: efficiency, security, and innovation. Here's how:

Efficiency: GenAI automates repetitive tasks, streamlining processes and ensuring faster software delivery. GenAI can preemptively address potential bottlenecks by understanding patterns and making predictions, providing a smooth delivery process.

Security: With AI-driven insights, potential vulnerabilities can be identified and addressed proactively rather than sitting on a backlog of work that needs to be prioritized and assessed.

Innovation: GenAI opens the door to continuous learning and improvement. Analyzing past outcomes can suggest improvement areas, ensuring that software delivery solutions are always ahead of the curve.

The Role of Google Vertex AI in Harness Security Testing Orchestration

Google Vertex AI is a managed machine learning platform that offers everything you need to build and use generative AIâfrom AI solutions, to Search and Conversation, to 100+ foundation models, to a unified AI platform. It seamlessly integrates tools for the entire machine learning workflow, from data labeling and model training to deployment and monitoring. This unified platform simplifies the ML process, catering to both experienced data scientists and newcomers to the field.

Harness's collaboration with Google and the use of Vertex AI for the STO use case signifies a commitment to excellence. Google Vertex AI, known for its advanced machine learning capabilities, enhances STO's ability to address vulnerabilities precisely. This collaboration ensures that STO is efficient and accurate, providing users with unparalleled security in software delivery.

Looking Ahead: The Future of DevOps with GenAI

The integration of GenAI in DevOps, exemplified by tools like Harness AIDA & Harness Security Testing Orchestration, marks the beginning of a new chapter in software delivery. It's about embracing the potential of AI, leveraging the best tools available, and delivering unparalleled value to users. The role of GenAI in shaping the future of software delivery will become even more pronounced, setting new standards of excellence in the industry.

Are you intrigued by the transformative power of GenAI in DevOps? Request a demo of our Security Test Orchestration (STO) and explore the capabilities of Harness AIDA in a dynamic software delivery landscape.

Check out our Harness YouTube channel to see the product in action!

AIDA to Explain and Fix Vulnerabilities: <https://www.youtube.com/watch?v=RntaYiC7Umo>

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/the-journey-to-using-feature-flags-at-their-full-potential>

The Journey to Using Feature Flags at Their Full Potential

At Harness, feature flags aren't just a tool; they are integral to promoting a culture of innovation and continuous delivery in software engineering. They may seem like simple software switches to hide or activate features, however, using them to their full potential cannot be done in a day. Rather, it requires following steps of learning and usage to bring you closer to that full potential radius.

Even here at Harness, we still walk our teams through the different steps of interacting with Harness Feature Flags, from basic awareness, to full maturity. In fact, this process allowed our teams to release the 4 new modules introduced at Unscripted 2023 by using our feature flags to release the modules to the production environment.Â

If you want your team to incorporate feature flags into your workflow, you can't just jump to that full maturity step. Rather, you need to follow a journey of learning and bring your team along for the ride. Let's dive deeper into these steps and explore how you can elevate your team's proficiency and efficiency in utilizing a feature flag management tool.

The 4 Steps of Integrating Feature Flag Adoption

Step 1: Awareness

Awareness is where the journey begins. It's not just about knowing that feature flags exist, but understanding their potential impact on software delivery. Building a comprehensive awareness involves creating an internal ecosystem where knowledge about feature flags is easily accessible and shared.

Action Steps:

- **Educational Content:** Develop a repository of educational content, incorporating various media - articles, webinars, and interactive learning modules to cater to diverse learning preferences.
- **Internal Advocates:** Identify and nurture internal advocates who can share their firsthand experiences and successes with feature flags, if there are any.
- **Community Engagement:** Leverage external communities and forums in places like social media to bring in diverse perspectives and insights.

Step 2: Experimentation

Experimentation is the playground where teams immerse themselves in the practical aspects of feature flags. It's a phase of learning, making mistakes, and iterating - all contributing to a richer, deeper understanding.

Action Steps:

- **Learning Workshops:** Follow workshops tailored to your organization's specific use cases and challenges.
- **Feedback Mechanism:** Establish a feedback mechanism to gain insights, address challenges, and refine the experimentation process.
- **Cross-Functional Collaboration:** Foster collaboration among developers, operations, and product teams to explore diverse use cases. Everyone can benefit from feature flags, so hearing them out can bring light to its potential.

â

Step 3: Adoption

Adoption signifies the transition from isolated experimentation to ingrained practice. It's where the application of feature flags becomes more structured and systematic.

Action Steps:

- **Integration with CI/CD:** Begin integrating feature flags into your CI/CD pipelines to streamline the deployment process. This is the first true step where feature flags are now a part of your delivery process.
- **Governance Structure:** Implement a governance structure to manage the lifecycle of feature flags, ensuring they are used optimally and retired when obsolete. So you may see this step as lying in Step 4 but it's key to start using a governance structure early on so that you can ensure that things can grow smoothly.
- **Metrics and KPIs:** Develop metrics and KPIs to evaluate the impact of feature flags on software delivery and performance. This can both show the worth of your new tool while illustrating potential room for improvement.

Step 4: Optimization

Optimization focuses on refining and enhancing the use of feature flags. It's where teams become adept at leveraging feature flags for varied scenarios, from dark launches to canary releases.

Action Steps:

- **Automation:** Explore automation opportunities to manage feature flags efficiently, reducing manual oversight. This includes things like assisted stale flag detection to manage tech-debt.
- **Customization:** Customize feature flagging tools and practices to align with specific organizational needs and objectives. Practices like targeted rollouts and progressive delivery now become more fine-tuned.
- **Advanced Analytics:** Utilize advanced analytics to derive actionable insights for informed decision-making in order to improve your pipeline, align with business goals, and enhance the user experience.

Embark on Your Journey with Harness Feature Flags

Embarking on the journey of feature flag adoption demands not just robust technology but a partner committed to your success. Harness Feature Flags is designed to be that partner. Every feature, every innovation is inspired by our unwavering commitment to empower developers to build, test, and deploy with unprecedented speed, scale, and safety.

Here are some of the key benefits of using Harness Feature Flags for your project:

- **Automated rollout pipelines:** Harness Feature Flags allows you to create automated rollout pipelines that can be used to deploy new features to production in a controlled and predictable manner.
- **Governance:** Harness Feature Flags has the most in-depth governance tool with Harness Policy as Code, powered by Open Policy Agent (OPA), providing a number of capabilities such as role-based access control, approval workflows, and audit logging.
- **Automated lifecycle management :** Harness Feature Flags helps you manage your flag tech-debt by detecting potentially stale or deprecated flags and automatically removing them if needed.
- **Integrated with CI/CD:** Since Harness covers a range of software delivery products, you can easily integrate your feature flags into CI/CD as a unified pipeline that helps maintain the flags as part of the software development lifecycle.

Conclusion

The transformative journey of adopting feature flags is as intricate as it is rewarding. It's not a linear path but a dynamic, evolving process that adapts and molds according to organizational needs and challenges.

Embark on your transformative journey with a free trial of Harness Feature Flags and unveil a world where innovation, reliability, and speed coexist, driving unprecedented value for your teams and customers alike.

Engage, Explore, Empower!

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

[Documentation](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[Case studies](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[We want to hear from you](#)

Enjoyed reading this blog post or have questions or feedback?
Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/safe-testing-in-production-with-harness-feature-flags>

Safe Testing in Production with Harness Feature Flags

Testing in production is a delicate balancing act. On the one hand, you want to ensure your changes are safe and won't break anything. On the other hand, you don't want to slow down your development process by going through a lengthy testing phase before every release.

At Harness, testing in the production environment is part of our development process. Just last month, developers from our Cloud Cost Management team used Feature Flags to test the release of their new Cloud Asset Governance feature. By using a flag to enable the beta AWS support to a subset of users, they could test the feature's performance before rolling it out globally to all customers by GA.

Why Test in Production?

Testing in production can be a scary thought for some, but it can actually be a very beneficial practice. Here are a few reasons why:

- To catch bugs that are difficult to find in staging or development environments. Production environments are the most complex and realistic environments in which to test your software. This means that you are more likely to catch bugs that would have been missed in staging or development environments during unit and integration testing.
- To test your software under real-world conditions. Production environments are where your software will be used by real users with real data. This means you can test your software under real-world conditions and ensure it is performing as expected.
- To get feedback from real users early and often. By testing in production, you can get feedback from real users early and often. This feedback can help you improve your software and ensure that it meets your users' needs.

When to Test in Production

Testing in production is not always appropriate. Here are a few times when you should consider testing in production:

- When you are launching a new feature or making a significant change to an existing feature.
- When you are testing a feature that is difficult to test in staging or development environments.
- When you need to get feedback from real users early and often.

Using Harness Feature Flags

Harness feature flags can help you achieve this balance by allowing you to safely test new features in production without impacting your existing users. With Harness feature flags, you can:

- Target specific users or groups of users with new features. This allows you to test your changes with a subset of your users before rolling them out to everyone.
- Gradually roll out new features to your users. This allows you to monitor the performance of your new features and make changes as needed before they are exposed to all of your users.
- Quickly and easily revert to a previous version of a feature. If you encounter any problems with a new feature, you can simply disable it with a feature flag and roll back to the previous version.

Here are some examples of how Harness feature flags can be used to safely test new features in production:

- A new e-commerce website is launching a new product recommendation algorithm. The team wants to test the new algorithm with a small group of users before rolling it out to everyone. They use a Harness feature flag to target a subset of users with the new algorithm. The team then monitors the performance of the new algorithm and makes changes as needed before rolling it out to everyone.
- A mobile banking app is launching a new feature that allows users to deposit checks using their phone's camera. The team wants to test the new feature with a small group of users before rolling it out to everyone. They use a Harness feature flag to target a subset of users with the new feature. The team then monitors the performance of the new feature and makes changes as needed before rolling it out to everyone.
- A social media platform is launching a new feature that allows users to create and share groups. The team wants to test the new feature with a small group of users before rolling it out to everyone. They use a Harness feature flag to target a subset of users with the new feature. The team then monitors the performance of the new feature and makes changes as needed before rolling it out to everyone.

Harness feature flags are a powerful tool that can help you safely test new features in production. By using Harness feature flags, you can ensure that your changes are safe and won't impact your existing users while avoiding the need for a lengthy testing phase before every release.

Here are some additional benefits of using Harness feature flags to test in production:

- **Reduced risk:** By testing new features with a subset of users before rolling them out to everyone, you can reduce the risk of impacting your existing users.
- **Increased speed:** By eliminating the need for a lengthy testing phase before every release, you can increase the speed of your development process.
- **Improved quality:** By monitoring the performance of your new features in production, you can identify and fix any problems before they impact all of your users.

Overall, Harness feature flags are a valuable tool for any organization that wants to test new features in production safely. By using Harness feature flags, you can reduce risk, increase speed, and improve quality. If you want to start testing in production, get started for free now!

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/introducing-harness-software-supply-chain-assurance-ssca>

Introducing Harness Software Supply Chain Assurance (SSCA)

If you've instituted a shift-left security approach where application developers are handily remediating code vulnerabilities without last-minute toil, your DevSecOps practice is off to a great start. But like the vast majority of organizations building modern applications, your company's software probably comprises a variety of open-source artifacts and other third-party components whose integrity the company is on the hook to ensure. Why? Look no further than the Solarwinds, Log4j, and Codecov breaches to see that a single, compromised artifact can wreak havoc for tens or hundreds of thousands of the software's consumers. For the purpose of preventing these types of destructive cyberattacks, a ubiquitous framework for hardening the software supply chain is required.Â

Executive Order 14028 was issued for the purpose of strengthening the United States' cybersecurity posture and requires organizations to implement safe development practices and maintain greater visibility into their software and its artifacts. EO 14028 has accelerated the adoption and standardization of Software Bills of Material (SBOMs) as a means of accounting for and ensuring the integrity of an application and all of its artifacts. The SBOM provides a critical layer of transparency and security, and its lifecycle management is a vital capability of a successful DevSecOps program. To comply with EO 14028, organizations need to adopt a framework like SLSA that aligns with NIST recommended SSDF to ensure artifact integrity, and generate SBOMs.

Introducing Harness Software Supply Chain Assurance (SSCA)

Building on the integrated, platform approach to DevSecOps that the majority of software-producing organizations are seeking, we're excited to introduce Harness SSCA today at Unscripted!

Harness SSCA extends application security beyond your own application code to the whole supply chain, enabling you to monitor and control open source software components and third-party artifacts, generate comprehensive SBOMs for enhanced visibility, and guarantee software integrity in accordance with SLSA and Executive Order 14028 requirements.

Here's an overview of the SSCA module's key capabilities:

Software Integrity based on SLSA

Supply-chain Levels for Software Artifacts (SLSA) is an important framework for creating a secure software supply chain. It lays out practices and guidelines to help you securely build your software and prove that it is not tampered with as it moves through different stages of your pipeline. Harness SSCA ensures the integrity of software by generating and validating the provenance (such as build source, branch, etc.) as per SLSA v1.0 specifications.

SBOM Orchestration and Lifecycle Management

The SSCA module offers users the flexibility to select their preferred tools for generating SBOMs in both CycloneDX and SPDX formats. Moreover, it empowers users to sign and validate SBOMs using their private keys, ensuring secure storage and sharing with software consumers.

Visibility and Control Of Open Source Software

With approximately 80% of a typical software application relying on open source software, the SSCA module offers deep visibility into the usage of open source components across all artifacts and their deployments. Further, it enables users to enforce policies by granting or restricting the use of components based on their versions, license, suppliers, and PURL.

A Look At Harness SSCA

Let's take a brief tour of Harness SSCA. We'll step through some of the workflows DevOps engineers and security analysts would undertake in generating SLSA provenance and SBOMs, verifying SLSA provenance, and enforcing policies to ensure that any risky open source software artifacts do not pass through to the deployment.

â

Guaranteeing that your application maintains its integrity throughout its journey through the pipeline begins with enabling SLSA.

SLSA Generation And AttestationÂ

With the Harness SSCA module, you can achieve SLSA Level 2 compliance by generating SLSA provenance according to the SLSA v1.0 specification. While SLSA level 1 stipulates that provenance exists, showing how a particular software package was built (what entity built the package, what build process they used, and what the top-level input to the build were), SLSA level 2 adds the requirement that the particular build runs on a hosted build platform that generates and signs the provenance itself.Â

In generating SLSA provenance within the build pipeline, you need to first generate a key pair using Cosign, as shown in the screenshot below.

SBOM Orchestration And Generation

A Software Bill of Materials (SBOM) is essential for understanding the components and dependencies within an application, which in turn enables your organization to manage open-source component risks effectively.

The Harness SSCA module generates, manages, and analyzes SBOMs for software artifacts. Below is the SSCA workflow for generating a SBOM:

â

SSCA supports SPDX and CycloneDX formats for SBOM generation and tools such as Syft and Cosign. When an SBOM is generated, the SSCA module generates and signs the attestation, ensuring that the information is accurate and trustworthy. The attestations are then securely stored in your artifact repository, where you can access and analyze them as needed.

OSS Governance Through Policy

With Harness, you have full control over their use throughout CI and CD stages via the SSCA module's policy management and enforcement capabilities. As you seek to deploy only compliant artifacts, you can put two types of policies into effect:

- **Deny list policies:** Define components, or combinations of component attributes, that are not allowed. If an artifact includes a component that is part of the deny list, the artifact's policy evaluation fails.â

- **Allow list policies:** Define components or combinations of component attributes that are allowed. If an artifact includes a component that is not part of the allow list, the artifact's policy evaluation fails.

When an artifact moves through your pipelines, the SSCA module checks the artifact and its associated SBOM against your defined policies. Policy violations are tabulated and displayed in detail.

SLSA Verification

You can use Harness SSCA to verify SLSA provenance using an OPA policy. In the example below, we are ensuring integrity of the application by validating the branch name on which it was built in the deploy

Software Supply Chain Assurance, the Harness Way

More and more enterprise organizations are taking a platform approach to building out their DevSecOps practices, and a big reason why customers come to Harness is the seamless integration of critical security capabilities such as Security Testing Orchestration (STO). Harness SSCA follows suit, delivering powerful OSS governance and SLSA compliance features.

To learn more about Harness SSCA, visit <https://www.harness.io/products/software-supply-chain-assurance>

Interested in seeing Harness SSCA in action? Sign up for a demo!

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/introducing-flag-archiving-and-restoration-for-smart-feature-flags>

Introducing Flag Archiving and Restoration for Smart Feature Flags

Progressive Delivery with Flag Archiving

Every day, we talk to teams that use feature flags to drive their modern software development process—simplifying deployments or providing much-needed flexibility to roll changes to groups of users and targets to implement progressive delivery. After the software release is complete and the user experience verified, the flag has served its purpose, and it's time to delete it - first from the code, then from Harness. Once in a while, though, someone deletes a flag *before* removing it from the code.â

Deleting a feature flag can result in the wrong values being served to your end customers, as Harness Feature Flags will fall back to the in-code default rule if it can't find a flag configuration to serve. Having this happen in a production environment can enable or disable specific features that a product manager didn't intend for the user base, creating an urgent issue needing resolution. Until now, when this happened, you'd have to create a new feature flag with a matching identifier to the deleted flag and recreate the flag-specific configuration.â With the introduction of flag archiving into our feature flag management lifecycle, a user can just click *restore flag*.

â

â

How It Works for Development Teams

Harness makes archiving a flag through the normal flag management process easy.

Developers can restore this flag within 30 days, otherwise, the archived flag will be deleted.

More than a Feature Toggle

Developers change, update, and create code every day. At Harness, we are addressing developer toil across the entire software delivery lifecycle to decouple the code deployment from feature releases. Removing the burden of complex lifecycle flag management is a straightforward improvement this new feature provides.

Create a free Harness Feature Flag account and leverage our Harness Developer Hub documentation on archiving and restoring flags.

â

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

[Documentation](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[Case studies](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[We want to hear from you](#)

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

[Sign up for our monthly newsletter](#)

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Automate Your CI/CD Pipeline Using Triggers

Continuous integration (CI) and continuous delivery (CD) are the two important aspects of a DevOps pipeline. While CI helps you with building and testing, CD helps you deploy your applications and services to the target environment. A great DevOps approach always starts with having the best set of tools to perform CI and CD and some best practices. The aim of DevOps is automation and accelerating the time to market. In this sense, CI/CD is a superpower to help organizations speed and automate. A CI/CD pipeline should always be configured in an automated fashion so that whenever a developer merges his/her code to the repository's main branch, the CI/CD system should get triggered.Â

â

In this article, let us learn how to set up CI/CD pipelines in an automated fashion using triggers.Â

CI/CD Pipeline Automation

Today, organizations like to move fast and break things early. Developers have become even more smarter with the DevOps concepts of CI/CD, GitOps, IaC, etc. As you march towards the DevOps approach, you will understand that automation becomes part and parcel of this journey. Having a great CI/CD platform/tool like Harness is necessary.Â

â

CI/CD (Continuous Integration/Continuous Delivery) pipeline automation is pivotal to modern software development and deployment. It refers to automating the steps involved in building, testing, and deploying software applications. The CI/CD pipeline enables development teams to streamline their workflows, improve collaboration, and accelerate the delivery of high-quality software. By automating the integration and delivery processes, developers can quickly identify and fix bugs, ensure compatibility across various environments, and continuously deliver new features and updates to end users.Â

â

The pipeline typically consists of multiple stages, including code compilation, unit testing, integration testing, and deployment to production. Automation tools and frameworks facilitate the seamless flow of code changes through these stages, allowing for rapid iteration and feedback cycles. With CI/CD pipeline automation, organizations can achieve faster time-to-market, higher software reliability, and enhanced overall efficiency in their software development lifecycles.

CI/CD Pipeline Automation Steps

- Developer Pushes Code to SCM (Source Control Management) Tool

â

- CI Build Gets Triggered

â

- Build Process

â

- Artifact Generation

â

- Artifact Storage

â

- Staging Environment

â

- CD Tool Takes Charge

â

- Testing in Staging

â

- Approval and Promotion

â

- Deployment to Production

â

By automating these steps, CI/CD pipeline automation streamlines the software delivery process, reduces manual errors, and enables teams to rapidly deliver reliable software updates to production.â

â

If you notice carefully, **Triggers** serve as the connecting links between these CI/CD steps, automatically initiating actions such as builds, deployments, and tests based on events like code commits, repository changes, or approval signals, enabling seamless automation and integration across the entire CI/CD pipeline.

â

Let us practically understand how to automate a CI/CD pipeline using triggers.â

Prerequisites

- Free Harness account
- Sample GitHub application repository to understand how triggers work. Fork the sample application code and use.â

Harness Triggers Tutorial

Triggers in the Harness platform are used to automatically start a pipeline or workflow in response to a specific event. They can be configured to run pipelines in response to a variety of events, such as:

- **Git events:** When a change is made to a Git repository, a trigger can be configured to run a pipeline. This can be used to automate the build, test, and deploy process for any application that is hosted in a Git repository.
- **Artifact events:** When a new artifact is created or updated, a trigger can be configured to run a pipeline. This can be used to automate the deployment of new features or bug fixes to production.
- **Schedules:** A trigger can be configured to run a pipeline on a schedule, such as once a day, once a week, or once a month. This can be used to automate the deployment of new features or bug fixes to production on a regular basis.
- **Manual triggers:** A trigger can be triggered manually by a user. This can be useful for troubleshooting or for deploying changes to production outside of the normal schedule.

Git Event Trigger

For example, you could create a trigger that starts a pipeline whenever a new commit is pushed to a specific branch in your Git repository.

Signup at **Harness**, create a project and create a pipeline in the project.â

Go to your pipeline studio and click on the âTriggersâ tab.

â

â

Start configuring your trigger.

When you click âAdd New Triggerâ, you will be presented with all the possible options for your trigger.â

â

Letâs select âGitHubâ and continue adding all the required details.â

You can also add conditions for more specific trigger events.â

As we donât have any runtime inputs, keep it as is.â

Create the trigger and your newly created trigger will appear under the âTriggersâ tab.â

â

Now, letâs push something to the main branch of the application repository. You should see an automated pipeline trigger as soon as the new code push happens to the main application repo.â

â

Similarly, you can create triggers for different events, as shown in the image below.Â

â

Docker Registry Trigger

Let's consider one more scenario where you want your pipeline to trigger when a new image or an artifact is pushed to the defined path. Let's pick "Docker Registry" from the available options below.Â

â

You define the artifact source and add the required details, such as your artifact type, artifact repo connector, location, etc.Â

â

So the location is very important here. Make sure you add it correctly. When any new image or artifact gets pushed to that path, the pipeline gets triggered.Â

To make it more specific, you can add conditions as to when you want the trigger to occur. We will keep it blank as of now.

There is no runtime input required, so keep it as is.Â

â

Create the trigger, and you can see the newly created trigger under the "Triggers" tab.

It is time to see if this new trigger is working as expected or not. Let's push a new image to the path/location specified in the artifact source with a different tag (other than "latest").

We can see the pipeline triggering action as soon as the new image gets pushed.

â

Click on pipelines to see the execution details.

â

This way, you can configure any events to trigger the pipeline automatically.Â

â

You can also schedule pipelines using triggers at Harness. Select ' Cron ' in the "Triggers" tab and mention when you want your pipeline to run.

Benefits of Using Triggers

Harness platform provides several benefits when it comes to using triggers. Triggers are essential components of a CI/CD pipeline that help automate the deployment process based on specific events or conditions. Here are some benefits of using Harness triggers:

Overall, using triggers in Harness platform enhances automation, streamlines deployment processes, and improves development efficiency by enabling event-driven workflows and providing visibility and control over the CI/CD pipeline.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

[Documentation](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/elevating-aida-harness-unveils-7-new-innovative-capabilities>

Elevating AIDA: Harness Unveils 7 New Innovative Capabilities

An AI-Infused SDLC Like Never Before

At the {unscripted} conference, Harness, the Modern Software Delivery Platform® company, announced 7 new features for our AI Development Assistant, AIDA.Â

In the dynamic realm of technology, where the lines between human ability and machine efficiency constantly blur, AIDA stands as a beacon, evolving beyond mere "artificial" intelligence.Â

"At Harness, we firmly believe in the concept of augmented intelligence, where our tools don't replace human talent but rather amplify it, acting as robust collaborators. We are unlocking the power of generative AI to empower developers. Our goal isn't just to remove developer toil from the SDLC, but eliminate it completely" said Harish Dodalla, VPÂ Product Management, Harness.

True to this vision, AIDA's seven new capabilities are not just features; they're revolutionaries. Their primary mission? To empower every developer by drastically reducing the toil that often dampens creativity, efficiency, and productivity.

We are thrilled to announce these new capabilities that promise to set a new benchmark in developer empowerment and the elimination of toil.Â

Transform Your SDLC Experience with AI For DevOps

Let's dive into these new features and how they span across each stage of the software delivery life cycle (SDLC).

AIDA brings to the table a powerful semantic search capability, ensuring you can navigate and manage your codebase with unparalleled precision. Starting with intuitive automated onboarding processes, AIDA simplifies initial Feature Flag setups, enabling seamless integration into your development environment. Have any questions? AIDA support is there to answer them quickly. Dive deeper with AIDA's advanced build and deploy troubleshooting capabilities, ensuring your code runs optimally. Enhance pipeline system integrity with our dynamic OPA policy generation and validate the resiliency of your infrastructure while using ChaosGuardâ€ to provide security governance. Concluding the cycle, AIDA's custom dashboard configuration offers granular analytics and real-time data visualizations.Â

Start Developing With Ease

A Leap in Precision: Semantic SearchÂ

Harness AIDA has several functionalities as a development assistant. Starting with semantic search, Harness AIDA allows a developer to quickly search repositories for relevant code in our Code Repository product.

AIDA's Semantic Search capability leverages Natural Language Processing (NLP) to understand the context around code, making the answers to search queries more relevant. It understands the meaning behind code, not just the keywords.

AIDA's Semantic Search capability streamlines code discovery, allowing developers to efficiently find what they need.

Feature Flags Unleashed: Seamless Integration with Automated Onboarding Code Samples

For Harness's Feature Flags product, AIDA's generative AI facilitates the onboarding experience by removing the manual efforts of learning how to integrate the product by providing code examples to use the SDKs. After simply selecting the desired language, framework, and version, AIDA seamlessly generates tailored code to help get you started immediately.

This isn't just a tool, it's a powerful developer asset, designed to elevate efficiency and precision in the development workflow. Now,

developers are able to swiftly onboard Feature Flags and experience language-specific SDK code examples without the conventional manual overhead of starting.

AIDA Support: Get Your Developer Questions Answered Efficiently

Harness AIDA further transforms the developer experience with the power of AIDA Support.Â

Harness AIDA Support is designed to assist users in real-time. Have a question? Simply ask, and AIDA Support dives deep into the Harness Developer Hub and documentation on your behalf.

With its swift and precise search capabilities, it offers answers without the need for manual navigation or extensive reading. AIDA ensures that every query is met with accurate information, making user support more streamlined and efficient than ever.

Build, Test, and Deploy Seamlessly

Continuous Delivery Error Analyzer: Unmatched Build and Deployment TroubleshootingÂ

Harness AIDA is continuing to build out its capabilities in build and deployment troubleshooting.

With AIDAâs Continuous Delivery (CD) Error Analyzer, upon a deployment failure, AIDA can get to the root cause in seconds, correlating the error with the issue and suggesting possible remediations.

Instead of manual developer efforts, AIDA sifts through the complex logs and dependencies, making Root Cause Analysis far more accurate and efficient.

The CD Error Analyzer doesn't just identify problems; it provides efficient actionable remediations.Â

The Next Frontier: AI-Assisted OPA Rego Policy Generator

Harness AIDAâs OPA Policy Generator leverages generative AI techniques to assist in writing Open Policy Agent (OPA) Rego policies, specifically tailored for the Harness platform.

Writing OPA policies manually is complicated and time-consuming. AIDA understands the contextual requirements of your environment and automates the tedious task of policy writing, allowing you to expedite the policy development process to your entire company.

This enables users to create effective policies with reduced manual effort. Moreover, AIDA democratizes the policy development process and allows users to focus on solving customer problems and driving new business opportunities.

Ensure Reliability Without Difficulty

Fortifying Systems: Introducing ChaosGuardâ¢ for Unmatched Resilience and Security Governance

Navigate the intricacies of chaos engineering with precision using ChaosGuard. Our platform allows you to define who can initiate chaos experiments, where they can conduct those tests, and when they're permitted to execute them. Safeguard sensitive environments by prohibiting chaos testing in areas like production or designate specific intervals, such as weekends or holiday seasons, for test runs. As your organization delves into chaos engineering, ChaosGuard provides rigorous security governance measures. This ensures that you can expand your initiatives with confidence, fostering resilience and innovation. With ChaosGuard, experience the full potential of chaos engineering, securely and efficiently.

Optimize Effortlessly

Crafting Insights: The Magic of Custom Dashboards

Leveraging AIDA's capabilities, users can transition smoothly through every stage of the SDLC. At last, exploring data visualizations becomes intuitive with the new AI-infused Custom Dashboards feature from Harness.Â

AIDA enables dashboard creation by simply prompting the interface with the domain of exploration, the type of visualization desired, and the specific metrics to track.

AIDA assists in crafting a variety of visualizations, including, but not limited to a single value representation, a comprehensive bar chart, and tables.Â

Organize dashboards seamlessly and derive insights that drive your business.

Eliminate Toil Across the SDLC By Harnessing the Power of AIDA

AIDA stands as a testament to the revolutionary potential of integrating AI into the SDLC, offering a holistic solution to streamlining

processes and optimizing outcomes.

As Harness looks ahead, the roadmap for AIDA's capabilities promises even greater advancements in this realm.Â

Interested in Learning More?

For developers seeking to be at the forefront of innovation, now is the time to integrate AIDA into your workflow. Witness the transformative power of AIDA firsthand and usher in a new era of development efficiency and excellence. Sign up for a demo today!

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/ci-cd-pipeline-as-code-with-harness>

CI/CD Pipeline as Code with Harness

As every company is considered a software company today, the world of DevOps is getting more attention than ever. The DevOps practices make software developers follow a standard approach to building and shipping software. Code and developers are the two main ingredients you need to build any successful software organization. The 'as code' movement has recently gained huge momentum in the industry. The 'as code' movement in software development and DevOps is revolutionizing how developers write code, provision infrastructure, create continuous integration and delivery pipelines, etc.Â

Today, we will be talking about 'as code' and 'pipeline as code' in particular. We will show you how 'pipeline as code' works in Harness.Â

The 'as code' Movement

Software development has come a long way; the tools like Docker, Kubernetes and many more have impacted the developers' way of building, testing and releasing software. Traditionally, infrastructure and configuration were managed manually, often leading to inconsistencies, errors, and difficulty in replicating environments. The "as code" approach expresses these aspects as code using domain-specific languages or configuration files, allowing them to be versioned, tested, and deployed alongside the application code.

The "as code" movement in software development refers to treating everything as code, such as infrastructure, configurations, and other

operational aspects the developer focuses on while building a software delivery pipeline.Â

The "as code" movement ultimately favours DevOps principles. It promotes collaboration, automation, and infrastructure-as-code practices. Moreover, it leverages YAML-like simple languages to edit and manage everything as code in a pipeline easily. Hence, it increases the developer productivity significantly.Â

What is Pipeline as Code?

'Pipeline as Code' is a developer-friendly revolutionary software development concept that has recently gained momentum. The approach basically states to treat the entire software delivery pipeline as code. It means that all the pipeline *stages, steps, tasks, and configurations* can be seen as code. Pipeline as code can include everything the application needs as code but will be focusing on CI/CD pipeline as code in this article.Â

Traditionally, software development pipelines were defined using custom scripting languages in an ad-hoc way, but 'Pipeline as Code' uses a declarative or imperative programming language to define the software development pipeline.Â

Harness lets you store all your configurations in your Git repository.Â

This is a simple CI/CD pipeline for our application.

Harness CI/CD pipelines can be created using a drag-and-drop approach and also through a YAML wizard.Â

You can see the âVisualâ and âYAMLâ tabs in the pipeline studio. You can click on the YAML tab to see how your CI/CD pipeline is configured.

â

â

The interesting fact is, you can edit this YAML and customize it accordingly. Push it to your GitHub repository, we will show you how to do it in a minute.Â

Letâs take a scenario where your manager wants to approve the pipeline before the application gets deployed to production.

Letâs go to our YAML and edit it to add an approval stage. Letâs add the Harness approval stage between the CI and CD stages.

We added the below code in the YAML between CI and Deploy stages.

You can see the added stage if you click on the visual tab.

This way, you can easily manage your CI/CD pipeline with the declarative YAML approach as code. Edit the YAML file to add more configurations, stages, steps etc.

The whole deployment pipeline can be viewed as a code.

This code with all the configurations can be pushed to a Git repository for others to reuse and collaborate.

â

You can simply click on âMove to Gitâ and provide the name, repo details and branch. Finally, click on âMove to Gitâ.

â

You can see the Harness folder with all the configuration files in the pushed repo.

When you get into the Harness folder, you can see the sample pipeline yaml.

â

Now, your team can easily track, make edits, reuse the logic and reduce the time it takes to rebuild/set up the whole pipeline.Â

How to Start with Pipeline as Code?

If you want to start using Pipeline as Code, the very first thing you should do is to go ahead and signup at Harness. Create a project, and fork this sample application that has all the ingredients to do CI/CD with Harness.

Create a new pipeline. Select âRemoteâ under âHow do you want to set up your pipeline?â.Â

Â

Select your connector type from the various options available. Select the one where your application code is present.Â

Make sure the connection is successful.Â

Now you can see the details being fetched in the **Create new Pipeline** section.

Click **Start** and select the Stage type.

We have selected **Build** as the stage and added the required details, as shown below.

Set up the stage, and it will ask you to save the pipeline to Git. You can select the branch to commit and save.

After successfully saving your pipeline to Git, you can go to your repository and see if it is done.

You can click on the **.harness** folder and see the pipeline yaml stored.

You can get back to your Harness pipeline studio, save everything and run the pipeline.

In the commits, you can see the recent commits.

This way, you can easily customize your pipeline using the **Pipeline as code** concept. Add more steps, stages and configurations through the YAML editor, and the pipeline can be stored in a Git repository. This makes it easy for other team members to reuse the YAML configuration and save more time.

Start creating your CI/CD pipeline as code today!

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer

happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/how-organizations-can-reduce-devops-costs-with-ci-cd-pipeline-templates>

How Organizations Can Reduce DevOps Costs with CI/CD Pipeline Templates

Everything that ties to DevOps promotes automation in one or the other ways. Development teams are expected to write code with high velocity so the Ops can release features to the customers with the same high velocity. Developers always look for ways to automate their tasks; several tools help them do so. Maintaining a standard set of rules and a cultural mindset is crucial in the teams. A positive culture promotes developer experience as well as productivity. While the path of DevOps starts with Continuous Integration (CI) and Continuous Delivery (CD) at many organizations, the tools and platforms make it different.Â

In this blog, we will discuss âPipeline Templatesâ, what they are, how to configure Harness Pipeline Templates as an example, and finally, the best practices that organizations can follow to scale their CI/CD maturity with such templates.

What are Pipeline Templates?

Templates are predefined configurations or blueprints that enable users to quickly and consistently provision infrastructure, deploy applications, and define workflows. In the context of Harness, a modern software delivery platform, templates serve as reusable patterns for defining and automating various stages of the software delivery process.

Harness templates typically include settings and parameters specific to a particular task or workflow, such as provisioning infrastructure on cloud platforms, deploying applications to different environments, or executing tests. Templates can be created and customized within Harness to match the unique requirements of an organization's software delivery pipeline.

When working with templates in Harness, users can select and apply them to create and manage resources or perform specific actions. Templates offer flexibility and efficiency by abstracting complex configurations into reusable components, promoting consistency and reducing manual effort. They also help standardize the software delivery process across teams and projects.

A Pipeline Template allows you to easily share pipelines among other teams or within your own team. This saves you time and effort by providing pre-set parameters and configurations for the pipeline rather than starting from scratch. By adding a Pipeline Template, you can automate building code into artifacts followed by deploying those artifacts into live services. You can choose from several Templates, such as the Build stage for pushing artifacts to the registry, running tests, and security scans. The Staging deploy stage can be used for deploying to Dev and QA followed by an Approval stage for PROD approval. Finally, the Prod deploy stage can be used to deploy to Production.

By leveraging templates in Harness, users can easily define and manage their software delivery workflows, ensuring a streamlined and automated approach to building, testing, and deploying applications.

Why use Pipeline Templates?

CI/CD Pipeline sprawl refers to the proliferation of multiple, divergent, and uncoordinated CI/CD pipelines within an organization. It occurs when teams or individuals create pipelines in an ad hoc manner without considering standardization, reuse, or consistency across projects. This can result in a chaotic and inefficient pipeline landscape.

Using templates in CI/CD pipelines is beneficial for several reasons:

- Standardization:** Templates enable the establishment of standardized practices and configurations for CI/CD pipelines. They provide a consistent framework for building, testing, and deploying applications, ensuring that teams follow best practices and maintain a uniform approach.
- Reusability:** Templates allow for the reuse of pipeline configurations across projects. Instead of reinventing the wheel for each new project, teams can leverage pre-defined templates to set up pipelines with proven configurations quickly. This saves time and effort while promoting consistency.
- Scalability:** Templates facilitate scalability by providing a scalable infrastructure for managing CI/CD pipelines. As the number of projects and pipelines grows, templates make it easier to manage and maintain pipelines in a centralized and organized manner.
- Collaboration:** Templates foster collaboration and knowledge sharing among team members. By using templates, team members can easily understand and contribute to each other's pipelines. It encourages collaboration and promotes a sense of shared

ownership of the CI/CD process.

- **Maintainability:** Templates simplify the maintenance of CI/CD pipelines. When updates or improvements need to be made, modifying a template ensures that the changes propagate to all pipelines that use it. This reduces the maintenance overhead and avoids inconsistencies arising from manually updating each pipeline.

Creating Different Types of Pipeline Templates on the Harness Platform

The output of the Templates would be a `template.yaml` file that consists all the details of your pipeline (including stages and steps) and it would look like this.

The quickest way to get started from a zero state is to use the Harness UX the first go. So, let's see how to start creating these templates.Â

First things first, sign up to Harness account with CD free plan.Â

When you signup and login to your Harness pipeline dashboard, on the left-hand side, you should see the `Templates` tab under `Project Setup`.Â

â

â

Click on `Templates` to create new templates.

When you click on `New Template`, you will see several options to create templates, as shown below.

Let us show you how to create **Step, Stage, and Pipeline** templates.Â

Step Template

In the Harness, using Step Templates allows for the templatization of individual steps within a pipeline template. Let's say you have 26 different pipeline templates, each representing a specific deployment process. Within each of these pipeline templates, a common step involves deploying a container to a target environment. Instead of manually configuring this step in each pipeline template, you can create a Step Template for the container deployment action.

The Step Template encapsulates the instructions, configurations, and parameters required to deploy the container. It could include details like the container image, environment variables, resource allocation, and deployment strategy. Creating a Step Template lets you define this deployment process once and reuse it across all 26 pipeline templates.

Note: A specific set of Steps can also be combined into a Step Group Template. We won't be covering the step group template in this article.Â

Click on `New Template`, and select `Step` as the option.

â

Name the Template, add a version label and mention where to save it from the three options Project, Organization and Account. Let's select Project in this tutorial.

â

Click `Start` to continue, and you will be presented with the step library with various options.

â

Let's select `Canary Deployment` from the list under Kubernetes.

â

Add the required details in the step.

â

An `Advanced` tab has more options to add to your step where you can set up conditional execution, failure strategy, looping strategy and command flags.

After all the setup and step configuration, you can save the template.

Now, get back to the Templates tab; you should see the recently created template there.

Stage Template

A Stage Template in Harness allows you to template a CI (Continuous Integration), CD (Continuous Delivery) stage or any other

custom stage.

The primary motivation behind using Stage Templates is to promote reusability and consistency across different deployment processes. When you have multiple pipeline templates, you may find that certain stages, such as CI (building and testing an application), are repeated across those pipelines. Instead of recreating and configuring these stages individually in each pipeline, you can create a Stage Template.

Click on **New Template**, and select **Stage** as the option.

This time, let us show you how to store your templates in your own repositories. Below you can see instead of **Inline**, we will select **Remote** and connect our GitHub repo. Create a PAT (Personal Access Token) to connect your GitHub repository. Here is a guide on how to create a PAT.

We will specify the repo to be connected, branch and YAML Path gets generated automatically for us. Click **Start**, and you will land on the pipeline studio.

â

Pick the **Deploy** stage and select the **Deployment Type** in the next step.

Select the run time input expression/value for **Service** and **Environment** and continue.

â

Select the deployment strategy.

This is how our deploy stage pipeline looks now.

â

Save everything and select a branch on your repository to store your stage template.

Under Templates, you can now see your recently created Stage template.

Pipeline Template

The primary purpose of a Pipeline Template is to simplify and expedite the creation of pipelines by providing predefined configurations and parameters. Instead of building pipelines from scratch, users can leverage Pipeline Templates as a starting point. These templates already contain predefined stages, steps, and settings, along with built-in parameters that can be customized as per specific deployment requirements.

Click on **New Template**, and select **Pipeline** as the option.

Click **Start**, and you will get to this pipeline dashboard to pick a stage.

Select the **Deploy** stage.

Add the required details to the stage. Just for example, let's pick **Kubernetes** as the Deployment Type. Add the Service, Environment, and Execution details. You can add fixed, runtime or expression values.

You can select your deployment strategy.

Finally, this is what your pipeline looks like, as shown below.

â

â

If you go to the templates under project set up, you should see our newly created pipeline template.

Harness Pipeline Templates in Action

Harness Pipeline Templates save a lot of your developers' time, so they don't have to reinvent the wheel. Once set up, they are free to collaborate and work. Let us show you how to use these templates.

â

Create a new pipeline and add the details.

â

Select **Start with Template** and continue.

Once you click on **Start with Template**, you will be redirected to the Templates library, where you will see all your templates

created.Â

â

Click on the one that you would like to use. Let's click on the first pipeline template shown in the list above.

You can see the details, activity log, template inputs, yaml and referenced by options.Â

â

You can simply click on "Use Template" to use this template.

This is what you should see.

â

You can save and run the pipeline to deploy a new service.

â

You can also push this templated pipeline to your GitHub repository.

You can see the pushed changes on your repository.

This way, you can have the smoothest Git experience also.Â

Comparing Pipeline Templates: Harness vs GitLab

- **Ease of Use:** Harness Templates involve a user-friendly interface, allowing users to define deployment workflows and configuration variables easily. On the other hand, GitLab templates use a YAML-based syntax, which can be more complex and require a deeper understanding.
- **Built-in Intelligence:** Harness templates come with built-in intelligence and automation capabilities. They suggest optimal deployment strategies, detect errors, and provide recommendations for improvements. These intelligent features help streamline the deployment process and improve overall efficiency. On the other hand, GitLab templates lack such built-in intelligence and automation, requiring users to rely more on manual configuration and troubleshooting.
- **Advanced Pipeline Management:** Harness templates offer comprehensive support for building and managing continuous delivery pipelines. They allow users to define complex deployment workflows with multiple stages, steps, environments, and conditions. Harness also integrates with various external tools and platforms, providing seamless integration throughout the CI/CD pipeline. While GitLab also provides CI/CD capabilities, the flexibility and advanced pipeline management features of Harness templates make them more suitable for complex and enterprise-scale deployments.

Pipeline Templates Best Practices and Guidelines

When working with Harness pipeline templates, you can follow several best practices and guidelines. These practices help ensure efficient and effective software delivery processes. Here are some recommendations:

Following these best practices and guidelines can enhance your software delivery pipelines' efficiency, scalability, and reliability using Harness.

Conclusion

Harness templates provide various benefits and act accordingly with the DevOps principles of automation and deploying faster. These templates are a solid way to optimize your developers' time and can be used to onboard new teams faster. In addition, Reusing CI/CD templates fastens the software delivery process and help organizations reduce their overall DevOps cost. It is time to scrap your old and traditional methods of building a CI/CD pipeline from scratch and use Harness templates instead.

[Try Harness Templates!](#)

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/migration-utility-ci>

Easily Migrate Continuous Integration Pipelines to Harness

We are excited to unveil our newest innovation, the Harness Continuous Integration (CI) Migration Utility. This free, open-source tool simplifies the process of transferring your CI pipelines from various providers to Harness.

Switching between CI providers has traditionally been a challenging and time-consuming task. Our new utility eases this process by automating up to 80% of the migration work, saving you both time and effort. We facilitate transitions from a range of CI providers including GitLab, CircleCI, GitHub Actions, Bitbucket, Cloud Build, Drone, and Travis CI.

Watch It In Action

In this demo, we'll show how to use the Harness CI Migration Utility to rapidly migrate a CircleCI pipeline to Harness for running automated builds. You can apply a similar process to migrate CI pipelines from other supported vendors.

Why Harness CI is Your Ideal Choice?

Harness CI, powered by Harness Modern Software Delivery Platform, provides a comprehensive solution designed to accelerate your builds, streamline your workflow, and give developers more time to focus on what they do best - coding.

Harness CI has all the features needed to scale for the largest organizations while empowering developers with the autonomy they need to ship the best features quickly, efficiently, reliably, and securely.

Key benefits of using Harness CI

- **Up to 4X Faster** - Harness CI is the fastest CI solution in the market. It offers various features to speed up builds, including Proprietary Test Intelligence which reduces unit tests cycle time with ML-based selective tests execution, Cache Intelligence for automatic dependencies caching and Docker Layer Caching.
- **Cost-Effective Builds:** Harness offers competitive cloud runners pricing. Couple this with our reduced build durations, and you stand to make significant savings.
- **Support for any Git provider** - Harness CI integrates seamlessly with any Git-based Source Code Manager, including GitHub, Bitbucket Server & cloud, GitLab, Azure Repos.
- **Versatile Workload Support:** Build and test mobile apps, web servers, functions, and more, in any programming language and anywhere. We support both cloud-native & traditional applications, and allow builds and tests on Linux, macOS, Windows, on a VM, or in a container. Use Harness Cloud or your self-hosted runners for maximum flexibility.
- **Thousands of Integrations** - Tap into the collective wisdom of community plugins by leveraging Drone Plugins, Github Actions, and Bitrise Steps in your CI Pipelines. It allows you to keep using your favorite community integrations directly from Harness CI, simplifying the transition from these vendors further.
- **Increase Collaboration and Standardization** with granular centralized templates. Capture best practices and onboard team members and projects easily.
- **Declutter your DevOps Toolbox** consolidating your entire software development lifecycle (SDLC) under the Harness umbrella to streamline your Build, Deployment, Release and Security processes with unified pipelines and
- **Enterprise-Ready** - Like other modules in the Harness suite, Harness CI is powered by the Harness Modern Software Delivery

platform. Our platform provides robust features that cater to enterprise needs, including granular Role-Based Access Control (RBAC), Policies-as-Code (OPA-based), Audit Trails, Advanced Dashboards, and more. See Harness platform capabilities to learn more about our platform.

Join the Open-Source Journey

In the spirit of collaborative innovation, we're making the Harness CI Migration Utility available as an open-source Git project. We invite developers worldwide to contribute to its development and enhancement, furthering our mission to streamline DevOps processes and eliminate developer toil.

Get Started with Harness CI

Sign up to Harness CI, and enjoy 2,000 free monthly credits for building and testing your code on Harness Cloud. Request a demo to understand how our CI/CD pipeline and Software Delivery Platform can revolutionize your software development process.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/introducing-the-harness-idea-portal-share-your-insights-shape-our-future>

Introducing the Harness Idea Portal: Share Your Insights, Shape Our Future

Today, we're excited to announce that we're changing how we engage with our customers to collect feedback and prioritize our roadmap.

Our journey so far

Up until now, we've utilized multiple channels such as calls, Slack messages, and Zendesk, to interact with customers, have them submit feature requests and report issues.Â

However, this approach had challenges, including:

- Challenging to communicate updates and ETA for specific feature requests
- Difficult to collaborate with multiple customers asking for the same feature
- Product managers had to collaborate in silos with numerous customers
- Customers couldn't collaborate with other users on similar topics

As our company grows and the number of customers using the platform increases, we have decided to adopt a more collaborative and streamlined approach

The Harness Idea Portal

After doing some research and input from customers, we've decided to standardize on the canny.io platform to power the Harness Idea Portal. This dedicated customer feedback portal is designed to address the challenges we faced and foster improved communication with our customers. Adopting the platform will help us work closely with our customers and improve their satisfaction by communicating more frequently on product updates.

How to Share Your Ideas

Submitting your ideas is super easy, simply follow these steps:

1. Navigate to ideas.harness.io (directly or from the product)
2. Select the module you'd like to propose an idea to:
3. Search to see if your idea has already been proposed. If so, you can "vote up" for it, increasing its chances of being considered. You will be notified of the status and progress of your idea.
4. If your desired feature hasn't been suggested yet, complete the form (provide a title and a short description). Once submitted, you will be notified of the status and progress of your idea.

And that's it!

What's Next? The product management team will triage these submitted ideas, ask questions if needed, allow all users interested in participating in the conversation, and provide updates on the progress and ETA. Users will automatically be notified on each change of tickets that they are subscribed to (either voted on or created the feature request).

In Conclusion

We are confident that using this platform will help us to work even more closely with our customers and keep us all up to date with the progress on features customers are interested in. We're encouraging all customers to be active in this forum so we can make Harness (even) better!

Have a fantastic idea? Please go to our Idea Portal and submit it :-)

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

[Documentation](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[Case studies](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer

happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/theres-a-good-chance-your-companys-cloud-costs-are-sky-high-take-these-5-steps-to-save-money-on-them>

There's a Good Chance Your Company's Cloud Costs Are Sky High. Take These 5 Steps to Save Money on Them.

â

This article was originally published in Entrepreneur on Aug 23, 2023

Many businesses donât give it much thought, but thereâs a huge expense lurking in their books that can easily spin out of control. Iâm talking about the cost of cloud services, which almost every company needs to compete in todayâs world.Â A

Just how volatile are cloud costs? It isnât a pretty picture. In a survey of 750 US enterprises from a wide range of industries, more than a third had cloud budget overruns of as much as 40%, and one in 12 topped that number. The global situation is equally shocking. Worldwide, businesses will invest almost \$600 billion in cloud spending this year. Conservative estimates indicate that nearly 30% of thatâaround \$180 billionâis wasted.

Most companies wouldnât tolerate such wastefulness in any other part of their business. But runaway cloud costs remain an exception, partly thanks to opaque billing. A typical scenario: A business learns that its tab from Amazon Web Services or another big cloud provider has jumped from \$100,000 to \$150,000 in just one month. What gives? Cloud may be simple to buy, but good luck deciphering that invoice, which can list thousands of acronym-filled services used by company software engineers.Â

Having served as CFO of several tech companies, Iâve seen how quickly those costs can add up. Think of it as the Wild West of spendingâmassive, unpredictable costs with little or no accountability. Thatâs why itâs so important to have a strategy for managing cloud expenses. For entrepreneurs and their companies, taming the beast means more money to invest elsewhere.Â

Hereâs how cloud costs became such a big problemâplus five tips for reining them in.Â

Why cloud is so easy to buyâand costs are so hard to control?

In the old days, businesses bought and maintained their own servers. Scaling up meant buying more hardware, a time-consuming task. Then, the cloud came along and changed all that, catering to companiesâ growing appetite for on-demand computing resources. The good news: software engineers could quickly buy what they needed without waiting for lengthy approval and procurement processes, helping accelerate innovation. The bad news? Lack of control over spending, which continues to balloon as offerings grow ever more complex.

For most businesses, the dirty little secret is that they donât understand how much cloud computing power, storage and other features they actually need. Thereâs often poor visibility into what other teams are doing, plus minimal accountability, with no one setting or enforcing budgets. This is compounded by a lack of tools to help them look under the hood.

How To Save Your Company Money On Cloud Costs

Working with Fortune 1000 companies, from big banks to airlines, Iâve seen up close how dramatic the cost savings can be. Here are five ways to take action:

1. Spread the word that everybody wins by cutting cloud costs

Reining in cloud spending starts with education and awareness. Simply sharing with employees the true magnitude of the problem can be powerful. We arenât talking about saving a few dollars. At many companies, the waste from cloud spend amounts to one of the single biggest budget items.

Then, rather than take a Big Brother approach, sell teams on the benefits of lower costs. The more a business can control cloud expenses, the more money it will have to hire another software engineer to develop a new product, or another sales rep to penetrate a new market. The message: Everybody wins by getting it right.Â

2. Get FinOps on the case

FinOps (a mashup of finance and DevOps) might sound technical, but it's just a name for the team that creates a process and framework for managing cloud costs. From sales to HR, nearly every department has a dedicated, expert operations team these days. As a major operational expense, cloud needs the same attention.

The FinOps team might be just two or three peopleâsay, a senior finance executive and the CIO or CTO. Have them create a framework that encourages accountability by assigning ownership of cloud spending to different business units. To get a clear, detailed picture of costs, give each team responsibility for its own budget and how much cloud it consumes.

3. When in doubt, automate cloud controls

Manually reviewing cloud bills each month for overruns and inefficiencies might sound archaic. Yet far too many companies still rely on this ad hoc approach. A far better strategy: leverage the growing number of tools on the market that help companies gain visibility into cloud spend in real time, flag overruns, automatically optimize where resources are allocated, and even offer suggestions for economizing spend.Â Â

For example, an alert system to detect spending anomalies should be table stakes. Besides catching questionable purchases by staff, this alarm can catch intrudersâfor instance, crypto miners mooching off the companyâs servers.

To avoid shelling out for idle cloud computing power, organizations can also use auto-stopping tools. Letâs say that each day from 9 pm to 6 am, usage of a subscription service drops to zero. Dispensing with manual controls, auto-stopping takes that expense off the board.

4. Make cloud part of procurement

Automation of governance and approvals is crucial, too. Would a company approve the purchase of a large piece of equipment with no questions asked? Any business spending millions of dollars a year on cloud should have procurement controls. With a cloud asset policy tool, it can establish guardrails that require people to justify their cloud spending.

5. Keep âtending the cloud gardenâ

Like a garden, cloud costs require consistent pruning. After taking a weed whacker to the biggest, most wasteful expenses, keep tending the smaller ones, or theyâll quickly grow out of control again. Cloud cost forecasting can help reduce the uncertainty around future usage.

All that yard work is well worth the trouble, because the potential savings are enormous, as much as 30% to 50% for many businesses. Thatâs real money better spent somewhere elseâon product development, on customer acquisition, and on the teams for whom cloud should be a means to drive innovation, not a costly headache.Â

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/harness-cloud-asset-governance-powered-by-aida-the-path-to-a-well-managed-cloud>

Harness Cloud Asset Governance powered by AIDAâ€¢: The Path to a Well Managed Cloud

We've all gotten busy and let our refrigerators get a bit messy over time with forgotten and expired food which takes up space, wastes money, and could lead to health risks. And that's with only one or two people using it. Imagine how bad the problem would be with dozens of people sharing it, losing track of what is theirs, and no one taking responsibility to clean out the science experiments hatching there. Scary right?

But that is exactly what happens in your cloud environment. The ability to self provision infrastructure leads to great agility in development, at the cost of keeping everything neat and orderly. Overtime, without someone chartered to take charge of the mess, your cloud becomes cluttered with unused resources, inefficient deployments and compliance and security issues, which together waste a large part of the \$490 billion cloud computing market spend.

Wasted spend means lost opportunities for innovation. In the current economy, efficiency is paramount as tightening budgets force teams to take a hard look at their cloud efficiency. Trying to manage your cloud assets by relying on ad-hoc scripting and manual processes leads to a massive overhead for developers, and lacks centralized reporting on the how well your cloud is being managed.Â

You need a better alternative, one that automates the governance of your cloud assets, while also giving your teams strong visibility into their cloud spend and efficiency.

Introducing Harness Cloud Asset Governance

Harness Cloud Asset Governance helps customers find and eliminate cloud waste automatically, while also providing a policy engine to ensure cloud resources are in compliance with corporate standards. By leveraging policy-as-code, it automates resource optimization, security, and compliance tasks, freeing your engineers to focus on creating innovative products and services that drive your revenue.

Harness Cloud Asset Governance is built on top of the popular open source software Cloud Custodian, designed to streamline cloud asset management and governance across your multi-cloud environments. It's a widely used open-source tool backed by CNCF, that enables users to modernize cloud governance with a *governance-as-code* approach that simplifies and automates setting up the guardrails needed to proactively manage your cloud assets.

However, using Cloud Custodian at scale comes with challenges:

- **No GUI:** It's a CLI driven tool only, requiring knowledge of how to create and edit YAML files in the correct syntax.
- **No Reporting:** Without a GUI, there's not way to provide for centralized visibility of rules and enforcement across stakeholders
- **No Security/Governance:** There is no ability to apply RBAC, or have audit trails for changes made.Â
- **Operational Overhead:** As with any open source tool, it requires ongoing maintenance, high management overhead, and needs dedicated infrastructure provisioned
- **No Generative AI:** No ability to provide guidance on rules, or smart policy authoring that allows user to create rules using natural language

Harness Cloud Asset Governance leverages all of the goodness of Cloud Custodian, such as its comprehensive coverage of governance policy support across cloud providers, while eliminating all the major pain points of self-hosting Cloud Custodian. Harness provides a rich set of preconfigured governance-as-code rules that make it easy to implement out of the box, as well as introducing AI Development Assistant (AIDAâ€¢) to power Cloud Asset Governance with a natural language interface that eliminates the need to understand YAML syntax to author policies.

Comprehensive Cloud Resource Coverage

Since we built Cloud Asset Governance on top of the open source Cloud Custodian, we can take advantage of all of the comprehensive coverage that currently exists, and new coverage as it's created. That allows us to support all major cloud assets, across all major cloud providers. Support for AWS is already generally available, and we've just launched the Azure beta availability, with GCP soon to follow.Â

Harness also offers a wide range of policies which are available out of the box, which you can leverage on day 0 to optimize your cloud resources and setup guardrails against future wastage.

AWS Resource Coverage (Comprehensive list)

- EC2 instances
- S3 buckets
- Lambda functions
- RDS (Relational Database Service) instances
- CloudFormation stacks

Azure Resource Coverage (Comprehensive list)

- Virtual Machines (VMs)
- Storage accounts
- App services
- Cosmos DB accounts
- Key Vaults

GCP Resource Coverage (Comprehensive list)

- Compute Engine instances
- Cloud Storage buckets
- App Engine applications
- Cloud SQL instances
- Cloud IAM policies

Our customers have seen immediate results from their use of Cloud Asset Governance. Jay Patel, Head of DevOps at Advanced, had this to say about it:

How Does Harness Cloud Asset Governance Work?

The possible combinations of governance-as-code rules that you can implement is nearly endless. You can keep it simple, and just look for orphaned, unused and idle resources, and shut them down. You can focus on governance and compliance, and write rules to enforce data retention policies, or ensure new assets are properly tagged. Or, you could focus on right-sizing assets such as compute, database or storage, as in the example below.Â

For some companies, data storage costs can be a significant portion of their cloud bill, especially if engineers overestimate the I/O speeds they require for their applications. Let's take Elastic Block Storage (EBS) volumes as an example. As new generations of SSDs are introduced, they become faster and less expensive than previous generations, and AWS passes these benefits to their customers with new EBS volume types.Â

EBS gp3 volumes are up to 20% less expensive than the older gp2 volume type, while also allowing independent provisioning of volume size, IOPS, and throughput. It's almost a no brainer to upgrade to gp3 as they are cheaper and offer better throughput than gp2.Â

But, how can you enforce this as a guardrail, and ensure existing and new volumes are all using less expensive, faster storage? With Harness Cloud Asset Governance, you can easily create a policy for this, but you won't have to because this policy is available out-of-the-box.Â

You simply need to pick the accounts & regions where you want to enforce the governance policy. Harness Cloud Asset Governance will automatically identify all the EBS gp2 volumes which can be auto upgraded to EBS gp3 and perform the upgrade. **All with a single click!**

If your teams are like most engineering teams, they're going to be a bit hesitant to allow rules to be implemented without at least a little bit of oversight. At least at the start. That's why Harness Cloud Asset Governance includes a "Dry Run" feature that lets you see what the rule would do, without enforcing the actions. Harness Cloud Asset Governance also keeps comprehensive logs of the actions that have been taken, so your teams have an audit trail if questions arise.

You can also set up Enforcement rules which will periodically scan through your cloud asset inventory to enforce policies automatically over time to keep your cloud assets well managed.Â

Harness AIDAâ€¢ (AI Development Assistant)

Cloud Asset Governance policies play a crucial role in automatically governing cloud assets and proactively optimizing cloud costs. However, authoring these governance-as-code policies can be challenging and confusing, especially when trying to master the correct syntax for your needs.Â

The Harness AI Development Assistant (AIDAâ€¢) was created to assist with understanding your existing policies, as well as creating new policies. Harness AIDA offers a user-friendly, natural language interface that serves as an excellent starting point for establishing the necessary policies for your cloud governance.Â

When you need to create a new governance policy rule, all you need to do is type in what you need, using a sentence format, such as âfind unused EBS volumes older than 90 days and delete themâ, or using our example above âfind all EBS gp2 volumes and upgrade them to EBS gp3â. Harness AIDA understands your requirements and generates customized policy rules to align with your needs.Â

How do you know that Harness AIDA created a valid rule (or that you even asked for a valid action)? Harness Cloud Asset Governance can validate the rule for you, before you ever try to implement it.

What about if you have a rule that was pre-defined, but you don't fully understand what it is going to do? Harness AIDA offers detailed descriptions of built-in rules and custom rules that others in the organization might have authored. This feature enables you to understand the purpose, scope, and implications of each rule, thereby facilitating informed decision-making during the policy enforcement process.

With AI becoming a critical capability in so much of our daily lives, it's no surprise that our customers are taking full advantage of Harness Cloud Asset Governance, powered by AIDA. Michal Malohlava, VP of Engineering at H2O.ai said:Â

What's Next?

Transform your path to a well managed cloud with Governance-as-code and try Harness Cloud Asset Governance now to receive automatic recommendations that can save you money, improve compliance, and reduce security risks. Still want to learn more? Talk to a Harness sales specialist today to get a free demo.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/comparison-guide/codefresh-cd-vs-harness>

Comparison Guide

Harness

Continuous Delivery & GitOps

vs

Codefresh

Harness Continuous Delivery & GitOps vs Codefresh

Continuous Delivery & GitOps

500-1000

2016

\$425M

Harness is categorized as:

- Continuous Integration
- Continuous Delivery
- Cloud Cost Management
- Cloud Cost Optimization
- Feature Flags
- Service Reliability Management
- Security Testing Orchestration
- Chaos Engineering
- Software Engineering Insights

Codefresh

Codefresh's DevOps automation platform brings CI/CD, Kubernetes, GitOps and more to help companies confidently deploy software faster.

51-100

2014

41m

Codefresh is categorized as:

- Continuous Delivery

â

What is the difference between Harness CD Vs. Codefresh?

Codefresh vs Harness: DevOps Tools Comparison

Updated

December 20, 2023

- Core GitOps Capabilities OOTB
- Visibility into Application State
- Simple, Intuitive UI
- Centralized Management
- Reporting & Custom Dashboards
- PR Pipelines to Automate Multi-Environments
- Wave Deployments
- Deployment VerificationÂ
- Policy-as-code
- Enterprise RBACÂ
- Audit Trails
- Config-as-code
- Security Testing Orchestration
- Chaos Engineering

<yes><yes>

<with><with>

<no><no>

<no><no>

<with><with>

<with><with>

<with><with>

<yes><yes>

<with><with>

<no><no>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<with><with>

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<with><with>

<yes><yes>

<with><with>

<yes><yes>

<with><with>

<yes><yes>

<yes><yes>

```
<yes><yes>  
<with><with>  
<yes><yes>  
<no><no>  
<yes><yes>  
<no><no>
```

No Scripting Required:

Codefresh doesn't rely on scripting too much. There's a massive library of steps (think: plugins) that covers many use cases. However, if there isn't a step for what you're looking for (or if you're simply looking for something a bit different), you'll have to write your own.

Scripting isn't the devil, but it certainly adds toil that could otherwise be avoided. For example, with Harness. Harness incorporates declarative pipelines with minimal maintenance overhead. Using Harness saves developers and DevOps alike time and effort.

Infrastructure Provisioners:

Codefresh offers Terraform and Pulumi as infrastructure provisioners. Surprisingly, they do not offer CloudFormation yet, but who knows what's on the horizon! Harness offers both Terraform and CloudFormation. There are no plans at this time to add a Pulumi integration.

Continuous Verification:

Continuous Verification is the process of monitoring your app for abnormalities after a deployment. For example, Continuous Verification could catch a latency issue or 5xx errors and automatically roll back your app to the previous version. The idea is to catch errors as quickly as possible – ideally, before customers notice – and make a seamless transition back to the prior version.

Codefresh doesn't provide CV capabilities out of the box. Harness does, effectively reducing risk and reputational damage from downtime. As for vendor integrations, Harness supports Prometheus, Datadog, AppDynamics, New Relic, StackDriver, CloudWatch, and custom monitoring and observability tools.

Change Management:

Codefresh provides an integration for Jira, along with some plugins that allow you to create, update, and validate issues. However, there is no integration for ServiceNow. Harness offers integrations for both Jira and ServiceNow in a substantial way: with our integrations, users can leverage Jira and ServiceNow as approval mechanisms – you'll be able to approve or reject a pipeline or workflow step all in one place!

Secrets Management:

Codefresh recently added a new feature called Secret Storage, which allows you to keep sensitive data on your cluster and for Codefresh to request it during pipeline execution on user's demand. This feature is only available on their enterprise plan.

Harness, on the other hand, offers multiple solutions when it comes to secrets management. It can be handled natively with our proprietary secrets manager, or it can integrate with third-party vendors like HashiCorp Vault, Amazon Secrets Manager, Google Secret Manager, AWS Key Management Service, Google Cloud Secret Manager, CyberArk, and Azure Key Vault.

Accelerate Metrics & Reporting:

There are four key metrics when it comes to software development: Lead Time (the average amount of time it takes from the time code is checked in to the version control system to the point in time where it is deployed to production), Deployment Frequency (the number of times deploys to production occur in a time period), Mean Time to Restore (MTTR: how long it takes to resolve or rollback an error in production), and Change Failure Rate (what percentage of changes to production fail).

These metrics are paramount in truly understanding performance. Codefresh currently does not offer an Accelerate metrics dashboard. On the other hand, Harness offers a beautiful dashboard specifically for these metrics and allows you to set alerts as needed – for example, you could set an alert to notify you if the Change Failure Rate goes above 1%.

**Please note: Our competitors, just like us, release updates to their products on a regular cadence. We keep these pages updated to the best of our ability, but there are bound to be discrepancies. For the most up-to-date information on competitor features, browsing the competitor's new release pages and communities are your best bet.*

[Try Harness For Free](#)

Continuous Delivery & GitOps

Interested in seeing what's under the hood? Browse through the Harness Continuous Delivery & GitOps Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/comparison-guide/digitalai-vs-harness>

Comparison Guide

Harness

Continuous Delivery & GitOps

VS

Digital.ai

Harness Continuous Delivery & GitOps vs Digital.ai

Continuous Delivery & GitOps

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

Digital.ai

Digital.ai develops Continuous Delivery and DevOps software, providing companies with the visibility, automation, and control they need to deliver software faster and with less risk.

101-250

2008

101-250

Digital.ai is categorized as:

Continuous Delivery

â

What is the difference between Harness CD Vs. Digital.ai?

Digital.ai (XebiaLabs) vs Harness: DevOps Tools Comparison

Updated

November 30, 2023

- SaaS & On-Premises
- No Scripting Required

- Ease of Use
- Cloud-Native App Support
- Traditional App Support
- Canary Deployments
- Infrastructure Provisioners
- GitOps (Pipelines as Code)
- Continuous Verification →
- Change Management Jira/SNOW
- Role-Based Access Control
- Secrets Management
- Audit Trails
- Accelerate Metrics & Reporting

â

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

CloudFormation and Terraform

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<with><with>

<yes><yes>

<yes><yes>

<with><with> **Manual Analysis**

CloudFormation and Terraform

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<with><with>

<yes><yes>

<yes><yes>

â

<yes><yes>

<yes><yes>

<yes><yes>
<yes><yes>
<yes><yes>
<with><with>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<with><with> Manual Analysis
CloudFormation and Terraform
CloudFormation and Terraform
<yes><yes>
<yes><yes>
<yes><yes>
<no><no>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<with><with>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>

SaaS & On-Premises:

A bit of history: Digital.ai, formerly XebiaLabs, was built and architected as on-prem software back in 2008. They released a lightweight, limited SaaS version in 2019. Updates are released every quarter. Reading between the lines, it can take a while for Digital.ai to resolve bugs. Harness offers SaaS and on-prem versions and they both offer the same features. Harness updates are done on a daily basis.

Ease of Use:

Digital.ai Deploy was recently named a leader in The Forrester Wave®: Continuous Delivery and Release Automation, Q2 2020. They received the highest scores possible in 19 criteria, including Product Innovation, Security, and User Experience. According to the report, customers were favorable on ease of deployment, the platform's overall flexibility, and the value the solution provides to their organizations in accelerating software delivery. Digital.ai Deploy provides a robust list of integrations and has plugins to provide extensibility. However, many of the reviews we could find online on Digital.ai Deploy were unfavorable when it comes to cost, ease of use/learning curve, and lack of features. Indeed, some users have referred to the initial setup/configuration as taking so much time they might as well have used a different tool, and the learning curve for complex issues was described as insane. Documentation being a nightmare or incomplete is referenced a few times as well. Harness, on the other hand, saves developers and DevOps time and effort. Harness also boasts a super simple, sleek UI. Additionally, Harness eliminates plugin and integration maintenance by running everything as containers. Getting a simple pipeline running takes 15 minutes, ensuring time-to-value is minimal.

Canary Deployments:

To utilize Canary deployments in Digital.ai Deploy, users will need to manually define the Canary process and manually configure their

load balancers. Digital.ai Deploy has helpfully documented the process. However, it's a cumbersome process when you can have a platform like Harness that provides Canary deployments out of the box â no coding required, only some minor config.

GitOps:

Digital.ai Deploy provides limited GitOps capabilities (pipelines as code) where YAML is linked to a single git project vs. one or more branches (for teams). No bidirectional sync exists, manual applies must be done to retain metadata, and diffs are not visualized between versions. Harness provides full GitOps capabilities.

Continuous Verificationâ¢:

Continuous Verification is the process of monitoring your app for abnormalities after a deployment. For example, Continuous Verification could catch a latency issue or 5xx errors and automatically roll back your app to the previous version. The idea is to catch errors as quickly as possible â ideally, before customers notice â and make a seamless transition back to the prior version. Digital.ai Deploy only supports Dynatrace (time-series metrics only) for deployment verification, and this capability is not linked to automatic rollback or verifying canary phases. Harness, however, provides Continuous Verification out of the box, effectively reducing risk and reputational damage from downtime. As for vendor integrations, Harness supports Prometheus, Datadog, AppDynamics, New Relic, StackDriver, CloudWatch, and custom monitoring and observability tools.

Secrets Management:

Digital.ai Deploy currently only offers secrets management through HashiCorp Vault. They do not offer native management or any other third party provider support. Harness, on the other hand, offers built-in secrets management. No third parties are required, but all of the major secrets managers are supported.

Accelerate Metrics & Reporting:

There are four key metrics when it comes to software development: Lead Time (the average amount of time it takes from the time code is checked in to the version control system to the point in time where it is deployed to production), Deployment Frequency (the number of times deploys to production occur in a time period), Mean Time to Restore (MTTR: how long it takes to resolve or rollback an error in production), and Change Failure Rate (what percentage of changes to production fail). These metrics are paramount in truly understanding performance. Digital.ai Deploy does offer DORA metrics tracking. Harness also offers a beautiful dashboard specifically for these metrics and allows you to set alerts as needed â for example, you could set an alert to notify you if the Change Failure Rate goes above 1%.

**Please note: Our competitors, just like us, release updates to their products on a regular cadence. We keep these pages updated to the best of our ability, but there are bound to be discrepancies. For the most up-to-date information on competitor features, browsing the competitorâs new release pages and communities are your best bet.*

[Try Harness For Free](#)

Continuous Delivery & GitOps

Interested in seeing what's under the hood? Browse through the Harness Continuous Delivery & GitOps Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/blog/september-2023-product-updates>

September 2023 Product Updates

At Harness, we have been hard at work so you can delight your customers without facing software delivery toil. Here is what has been changing in the previous month across Harness Products.

Continuous Delivery & GitOps

[Full Continuous Delivery & GitOps Release Notes](#)

Continuous Integration

[Full Continuous Integration Release Notes](#)

Feature Flags

Full Feature Flags Release Notes

Cloud Cost Management

Full Cloud Cost Management Release Notes

Security Testing Orchestration

Full Security Testing Orchestration Release Notes

Chaos Engineering

Full Chaos Engineering Release Notes

Internal Developer Portal (IDP)

Full Internal Developer Portal Notes

Software Engineering Insights

Harness Platform

Here are some important improvements to our platform that should enhance your user experience:

Full Platform Release Notes

Self-Managed Enterprise Edition

Full Self-Managed Enterprise Release Notes

Early Access

Continuous Delivery

Continuous Integration

Security Testing Orchestration

Chaos Engineering

Self-Managed Enterprise Edition

Full Early Access Release Notes

Continue the Journey

- Learn more hands-on with our increasing list of Tutorials.
- Further your knowledge with one of our Certifications.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/hashicorps-bsl-license-change-what-this-means-for-harness-cd-customers>

HashiCorp's BSL License Change: What this means for Harness CD customers?

HashiCorp recently announced that it is changing the licenses on their open source projects from MPL to BSL, which includes the Terraform and Vault projects. Although we understand and respect HashiCorp's decision to change to a BSL license, this change has left many users wondering how it will affect their use of Harness's CD product module. Ultimately, since Harness CD focuses on automating and optimizing the software release and delivery process, and Terraform is primarily used for infrastructure as code to provision and manage cloud resources, and Vault primarily helps manage and provision secrets securely in infrastructure, Harness CD does not significantly overlap the capabilities of Terraform, Vault, or any other HashiCorp commercial product, and is therefore not competitive. So the licensing change to the BSL license, as of the date of this publication, does not impact the following Harness CD use cases which incorporate Hashicorp products. You can also find these use cases at <https://www.hashicorp.com/partners/tech/harnessio>.

Looking Ahead

Harness is committed to ensuring the ecosystem around the use cases that connect Terraform, Vault, and Harness, continues to thrive. We will continue to invest in the Harness Terraform Provider, plus add support for Harness resources, along with Terraform and Vault features.

If you have any further questions or concerns, please contact your Customer Success Manager.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/products/cloud-cost>

Control Cloud costs with Intelligent Automation

Automate and save up to 70% on your cloud bill.

Gain insights into detailed unit-level cloud costs.

Automate and save up to 70% on your cloud bill. Gain insights into detailed unit-level cloud costs.

Cost Reporting

Attribute & Manage Costs with Granular Visibility

Simplify hierarchical cost attribution for chargeback and show-back. Allocate shared and complex resource costs. Reuse team, department, BU definitions across reports.

Features: Perspectives and Cost Categories

Cost Optimization

Automated Cost Optimization: Dynamic Idle Resource Detection & Remediation

Automatically shut down idle cloud resources when inactive, bring them back when activity is detected saving up to 70%.

Feature: Cloud AutoStoppingTM

Cost Governance

Manage Cloud Assets with Governance-As-Code

Automated governance-as-code with cost, security, and compliance policies. Right-size under-utilized resources and clean up unused ones for any cloud, resource, and action. AI-powered policy generation with Harness AIDA.â”

Feature: Cloud Asset Governance

Integrations

Trusted by DevOps and Developers

By choosing Harness for CI and CD, we were able to give the governance policies to the developers and create the guardrails we needed. Harness gives us a platform rather than just a DevOps tool.

Synopsys

"All developers in Synopsys have access to recommendations within Harness CCM. We recommend that developers use this information when deploying new microservices to the cloud. This is what makes this cloud cost management tool a game changer for us as we balance the speed of innovation with its cost."

Carvana

âWe understand now, when we run this environment for five days, this is what the cost is. So now I'm going to run it for three days or I'm going to power it off when I don't need it. Cloud Cost Management helps us immensely.â

Industry Recognition

Get started with smart Cloud Cost Management

Take control of your cloud costs with intelligent automation and granular cost visibility

Source URL: <https://www.harness.io/comparison-guide/jenkins-vs-harness>

Comparison Guide

Harness

VS

Jenkins

Harness Continuous Delivery & GitOps vs Jenkins

Continuous Delivery & GitOps

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

Jenkins

Jenkins is an open source automation server with an unparalleled plugin ecosystem to support practically every tool as part of your delivery pipelines.

11-50

2012

-

Jenkins is categorized as:

Continuous Delivery

â

What is the difference between Harness DevOps Tools Vs. Jenkins?

Jenkins vs Harness: DevOps Tools Comparison

Updated

November 30, 2023

- **SaaS & On-Premises**
- **No Scripting Required**
- **Ease of Use**
- **Cloud-Native App Support**
- **Traditional App Support**
- **Canary Deployments**
- **Infrastructure Provisioners**
- **GitOps (Pipelines as Code)**
- **Continuous Verification â€**
- **Change Management Jira/SNOW**
- **Role-Based Access Control**
- **Secrets Management**
- **Audit Trails**
- **Custom Dashboards**

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

CloudFormation and Terraform

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no> **On-prem Only**

<no><no>

<no><no>

<with><with>

<yes><yes>

<no><no>

<no><no>

<no><no>

<with><with>

<with><with>

<with><with>

<no><no>

<yes><yes>

<no><no>

On-prem Only

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<with><with>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

```
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<with><with>
<yes><yes>
<with><with>
<yes><yes>
<with><with>
<yes><yes>
<no><no>
<yes><yes>
<with><with>
```

SaaS & On-Premises:

Jenkins is an open-source solution, and as such, only offers an on-prem version. Harness provides both on-prem and SaaS versions of the product, which adds appeal for those who don't want to manage or maintain backend servers.

No Scripting Required

One of the major pain points of Jenkins is the sheer amount of toil it takes to operate. This includes initial setup, scripting to extend the solution to a viable CD candidate, and then maintaining it. In fact, we've calculated that it takes between 2 to 5 engineers just to maintain Jenkins on a daily basis. Many organizations use Jenkins because it's an open-source solution, but the cost required to maintain it often makes it an inefficient solution. Harness, on the other hand, incorporates declarative pipelines with minimal maintenance overhead. Using Harness saves developers and DevOps time and effort.

Ease of Use:

Jenkinsâ| Where did you go wrong? Is it that you're a decade old? Is it that you depend on plugins and scripts? Is it that you weren't initially designed to be cloud-native? It's a little bit of âall of the above.â The simple fact is that engineers will need some number of plugins/scripts in order to get Jenkins to work the way they need it to. That, in turn, requires maintenance â and opens engineers up to the dependency hell that is Jenkins. Harness, on the other hand, has a super simple, sleek UI. While there may be a learning curve, there is no need for scripting as pipelines are declarative.

Cloud-Native App Support:

Jenkins wasn't designed to be cloud-native, but developers and DevOps teams have been making it work for years. There is a group of contributors and collaborators focusing on improving Jenkinsâ cloud capabilities. For instance, they've created a Jenkins Kubernetes operator. While it's definitely progress, albeit slow, we frankly don't believe that CI/CD should be that hard.

Traditional App Support:

As we mentioned, Jenkins wasn't designed to be cloud-native, so this is the one thing it does well. But, Harness does it too â among a plethora of other things.

Canary Deploymentsâ¢:

There is absolutely no native Canary deployment strategy with Jenkins. Sure, you can manually script a Canary deployment to a Kubernetes cluster with some serious edits to a Jenkinsfile, but it's not easy. Harness provides guided Canary deployments out of the box â no coding required, only some minor config.

Verification:

Continuous Verification is the process of monitoring your app for abnormalities after a deployment. For example, Continuous Verification could catch a latency issue or 5xx errors and automatically roll back your app to the previous version. The idea is to catch

errors as quickly as possible â ideally, before customers notice â and make a seamless transition back to the prior version. Jenkins does not provide Continuous Verification. Harness, however, provides Continuous Verification out of the box, effectively reducing risk and reputational damage from downtime. Harness supports many vendors, including Prometheus, Datadog, AppDynamics, New Relic, StackDriver, CloudWatch, and custom monitoring and observability tools.

Secrets Management:

Jenkins does not offer native secrets management capabilities. There are many ways to do it through a third party, such as HashiCorp Vault or Helm Secrets. Harness, on the other hand, offers built-in secrets management. No third parties are required, but all of the major secrets managers are supported.

Audit Trails:

Jenkins does not feature native audit trail capabilities. To get audit trails, you must depend on plugins. Harness provides audit trails on every pipeline, workflow, step, execution, and change. Itâs all audited by Harness so you have a complete trail of all user activity.

Accelerate Metrics & Reporting:

There are four key metrics when it comes to software development: Lead Time (the average amount of time it takes from the time code is checked in to the version control system to the point in time where it is deployed to production), Deployment Frequency (the number of times deploys to production occur in a time period), Mean Time to Restore (MTTR: how long it takes to resolve or rollback an issue in production), and Change Failure Rate (what percentage of changes to production fail). These metrics are paramount in truly understanding performance. Jenkins does not provide native Accelerate metrics dashboards â you must depend on third party integrations to achieve desired reporting. Harness offers a beautiful dashboard specifically for these metrics and allows you to set alerts as needed â for example, you could set an alert to notify you if the Change Failure Rate goes above 1%.

**Please note: Our competitors, just like us, release updates to their products on a regular cadence. We keep these pages updated to the best of our ability, but there are bound to be discrepancies. For the most up-to-date information on competitor features, browsing the competitorâs new release pages and communities are your best bet.*

Try Harness For Free

Continuous Delivery & GitOps

Interested in seeing what's under the hood? Browse through the Harness Continuous Delivery & GitOps Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/comparison-guide/gitlab-cd-vs-harness>

Comparison Guide

Harness

Continuous Delivery & GitOps

VS

GitLab

Harness Continuous Delivery & GitOps vs GitLab

Continuous Delivery & GitOps

500-1000

2016

\$425M

Harness is categorized as:
Continuous Integration
Continuous Delivery

Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

GitLab

GitLab Continuous Delivery (CD) helps streamline and automate the application release process to make software delivery repeatable and on demand.

1000+

2014

424.2m

GitLab is categorized as:
Continuous Delivery

What is the difference between Harness CD Vs. GitLab?

GitLab CD vs Harness: DevOps Tools Comparison

Updated

January 12, 2024

- **SaaS & On-Premises**
- **No Scripting Required**
- **Ease of Use**
- **Cloud-Native App Support**
- **Traditional App Support**
- **Canary Deployments**
- **Infrastructure Provisioners**
- **GitOps (Pipelines as Code)**
- **Continuous Verification**
- **Change Management Jira/SNOW**
- **Role-Based Access Control**
- **Secrets Management**
- **Audit Trails**
- **Software Engineering Insights (VSM)**
- **Pipeline Templates**
- **Non Git SCM support (ex. BitBucket)**

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

CloudFormation and Terraform

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>
<yes><yes>
â
<yes><yes>
<yes><yes>
<with><with>
<with><with>
<yes><yes>
<yes><yes>
<with><with> **Manual Analysis**

Terraform

<yes><yes>
<no><no>
<with><with>
<with><with>
<with><with>
<yes><yes>â
<yes><yes>
<yes><yes> GitLab has Auto DevOps and Auto Deploy reusable templates
<no><no>
<yes><yes>
<yes><yes>
<yes><yes>
<with><with>
<yes><yes>
<with><with>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<with><with> Kubernetes Only
<yes><yes>
<with><with> Terraform Only
<yes><yes>
<with><with> Flux Only
<yes><yes>
<no><no>
<yes><yes>

```
<with><with> SNOW Only  
<yes><yes>  
<with><with>  
<yes><yes>  
<with><with>  
<yes><yes>  
<yes><yes>â  
<yes><yes>  
<yes><yes>  
<yes><yes>  
<yes><yes> GitLab has Auto DevOps and Auto Deploy reusable templates  
<yes><yes>  
<no><no>
```

Modularity

While Harness and GitLab each claim to provide an end-to-end software delivery platform, only Harness empowers its customers to pick and choose which elements to adopt, making it easy to plug-in to a heterogenous environment. Already have source control and builds solved?Â That's fine, start with Harness Continuous Delivery and GitOps, layering on Feature Flags, Security Test Orchestration and other capabilities as appropriate to you. Harness plays nice with others in the ecosystem.

SaaS & On-Premises

Both GitLab and Harness offer SaaS and on-prem versions of their product. No matter your use case, both solutions can deliver.

Ease of Use

A plus to using GitLab is that it was, initially, a source code management tool / Git repository. As such, converting to their CI/CD platform would have advantages when it comes to easy integration. However, if GitLab is your SCM tool of choice, rest assured that Harness easily integrates with it as well. When it comes to ease of use, some features are buggy and that the overall system can be quite slow. Documentation was found to be lacking for more complex setups. UI is clean, but not intuitive â definitely has a learning curve and needs improvements in order to be less confusing. Lastly, CI can be hard to integrate with automatic and manual tests users have created in the past with their prior CI tool.

Policy Governance

GitLab provides some Policy capabilities around its security scanning capabilities with a proprietary security policy definition.

Harness has implemented the Open Policy Agent, enabling central teams to establish common policies across the wider organization. These may be security centric, such as requiring passing scans in any pipeline moving towards production, or not. For example, OPAÂ can require a feature flag to be enabled in Test before it is enabled in Production. Or it can generate a warning in infrastructure changes will increase costs. The possibilities are nearly endless.

As of January 2023, GitLab has been considering moving to OPAÂ for the past four years, but has not done so:Â <https://gitlab.com/gitlab-org/gitlab/-/issues/55651>

Cloud-Native App Support

GitLab supports AWS, Azure and GCP clouds. It also supports Kubernetes, Helm, and ECS for container orchestration. Harness supports all of the above, and additionally, all other cloud providers. Harness was designed to be cloud-native.

Traditional App Support

GitLab supports JAVA, .NET, and custom traditional apps. But, Harness does it too â among a plethora of other things.

Canary and Blue/Green Deployments

GitLab provides some support for advanced deployment strategies, particularly on Kubernetes. Harness provides full support for Canary, Blue/Green and Rolling deployments on Kubernetes, but also on additional deployment targets including VMware Tanzu Application Services, Azure Web Apps, Google Cloud Functions and more. GitOps style deployments leveraging ArgoCDÂ implement

the Rollouts capability.

With Continuous Verification (see below) Harness is able to detect problems in releases automating the decisions to rollback or proceed in progressive delivery.

Infrastructure Provisioners

GitLab offers a Terraform integration for infrastructure provisioning.

Harness offers a Terraform integration too, but also goes further. Harness provides first class support for CloudFormation, Terragrunt, Azure Resource Management, Azure Blueprints, and AWS CDK. All other infrastructure provisioners are supported via shell script.

Harness even offers a powerful Infrastructure-as-Code Management module, adding pipeline orchestration to IaC, as well as governance capabilities with OPA.

Continuous Verification

Continuous Verification is the process of monitoring your app for abnormalities after a deployment. For example, Continuous Verification could catch a latency issue or 5xx errors and automatically roll back your app to the previous version. The idea is to catch errors as quickly as possible – ideally, before customers notice – and make a seamless transition back to the prior version. GitLab does not provide Continuous Verification capabilities, only a manual process with a Prometheus integration. Harness, however, provides Continuous Verification out of the box, effectively reducing risk and reputational damage from downtime. As for vendor integrations, we mentioned Gitlab's Prometheus – sadly, it's currently their only tool integration. Harness supports many vendors, including Prometheus, Datadog, AppDynamics, New Relic, StackDriver, CloudWatch, and custom monitoring and observability tools.

Change Management Jira/SNOW

GitLab offers a Jira integration, but that's more so for record keeping purposes. You can't do anything substantial, like using Jira to approve or reject a pipeline or workflow step. Harness, however, offers this functionality. With a few simple steps, you can easily leverage Jira and ServiceNow as approval mechanisms.

Role-Based Access Control

When it comes to configuration, GitLab provides only 5 predetermined roles that are not customizable at a granular level, and permissions cannot be separated by environments. For deployments, GitLab does not support RBAC, except for their native Kubernetes integration – that means no granular support for deployments that would allow for specific user, group, environment, or namespace mappings. Harness, on the other hand, provides fully-configurable CRUD access across every entity, whether services, environments, workflows, pipelines, or provisioners. Harness also provides full Deployment RBAC for an unlimited number of user groups across every application and environment. There are also separate permissions for governed pipeline execution than for individual workflow execution. For example, developers can kick off pipelines that meet all security and quality checks and have all required approvals in order to progress through environments, but can't deploy directly to specific environments, like staging or prod.

Secrets Management

GitLab does not offer native secrets management capabilities. They have selected Vault by HashiCorp as their first supported secrets management partner, which means you must first configure your Vault server. Harness, on the other hand, offers proprietary, integrated secrets management. No third parties are required, but all of the major secrets managers are supported.

Audit Trails

GitLab provides good governance and compliance features, but many of them are only available on their Premium or higher plans, audit trails included. GitLab provides a list of every [audit event](https://docs.gitlab.com/ee/administration/audit_events.html), some which are pruned after 30 days. Harness provides audit trails on every pipeline, workflow, step, execution, and change. It's all audited by Harness so you have a complete trail of all user activity.

Software Engineering Insights (VSM)

Harness offers a robust Software Engineering Insights module helping engineering leaders discover bottlenecks in delivery and help teams deliver more effectively and predictably. A GitLab released a Value Stream Management product.

**Please note: Our competitors, just like us, release updates to their products on a regular cadence. We keep these pages updated to the best of our ability, but there are bound to be discrepancies. For the most up-to-date information on competitor features, browsing the competitor's new release pages and communities are your best bet.*

Try Harness For Free

Continuous Delivery & GitOps

Interested in seeing what's under the hood? Browse through the Harness Continuous Delivery & GitOps Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/blog/intro-to-gitops-source-patterns>

Charting Software Delivery Flight Paths: Intro to GitOps Source Patterns

When you book a flight, you trust the airline to get you where you need to be in one piece. Ticket details like destination city, departure time, seat, and fare class comprise a **declared source of truth**, and the airline mobilizes its vast logistical apparatus to assign a plane, pilot, and flight plan to carry you to your destination.Â

Software development benefits from a similar abstraction approach. Layers of delivery complexity have pushed teams to incorporate **GitOps**. The desired states of applications and infrastructure are maintained in version control (Git), and automation tools enforce changes to the target environments (Ops). Just like how I rely on the airline to engineer my transportation to Cabo, software developers can now focus on business logic while letting CD tools handle the deployment plumbing.

Implementing proper **source control patterns** has therefore become critical to a successful software delivery practice. Before GitOps, engineering efforts largely went toward creating tedious deployment runbooks. With imperative deployment steps automated, a core design problem now involves properly structuring the repositories hosting our declarative configurations.

Well-Architected Route Maps

Consider our air travel analogy. An airline typically organizes its routes in one of two ways. The most common choice is a *hub and spoke* model. The airline centralizes operations in a few cities (the hubs), and those cities serve the outlying destinations (the spokes).

In this model, passengers generally need to go through a hub city to reach their destination, reducing the number of direct flights. However, airlines are able to better respond to emergencies and stresses on the network by surging planes and staff only within a few hubs. The hub model is *less efficient* on a per route basis but is overall *easier to administer*.

The second model is **point-to-point**. A point to point model connects cities directly without using central hubs. Southwest and other low-cost carriers are famous for this approach, implementing a mesh of routes connecting individual cities.

Point-to-point permits more nonstop flights (no need to connect through a hub), while also allowing airlines to easily add or remove new routes. However, managing all those route dependencies has high operational overhead. And a single route failure (say, a thunderstorm in the midwest) can have a cascading effect on the rest of the network, as it is more difficult to surge or reallocate resources than in the hub and spoke model. Routes in the point to point model are therefore *more efficient* but can be *administratively burdensome*.Â

It turns out that these approaches can inspire how we organize our software projects.

The Monorepo Approach:Â Integrated and Atomic

A monorepo approach means all source files for all services live in a single repository. This is comparable to the hub model described earlier. For example, the structure of an e-commerce app for âWidgets Co.â might look like the following:

The whole team collaborates on this single repository, with some roles potentially focusing on different folders. For example, UX designers might spend the bulk of their time in *frontend/*, and the technical writers might camp out in *docs/*.Â

Teams that choose to go the monorepo route cite some the following benefits:

- **Atomic changes:** The entire codebase shares a common commit history. Each new commit, whether a feature update or rollback, is applied to the entire application.
- **Ease of refactoring:** Changes in dependencies or application logic that affect multiple components can propagate all from one place.Â
- **Centralized tools:** CI/CD and infrastructure configurations are centralized alongside application code. Itâs straightforward to ensure the entire codebase must pass build and test pipelines before deployment.
- **Reduced administrative overhead:** Security policies, approval rules, and visibility tools all need to point to only one repository.

Monorepos can serve as a logical starting point for brand new initiatives. Developers can easily collaborate as they have a single point of access to the entire application. However, while monorepos handle scaling fairly well, they can encounter challenges as codebase complexity grows.

- **Repository size:** Git operations can start to show strain as the size and history of the monorepo grows, though this can be somewhat remediated through interventions like Git LFS and shallow cloning.
- **Longer build times:** Each build pipeline runs against the entire application, even if only a small part changed.
- **Tooling limitations:** Existing security and collaboration tools need to handle a massive, complex codebase. Challenges in enforcing granular access control can lead to over-permissioning.

Organizations have successfully maintained monorepos of massive scale. Still, some teams decide that as the business grows the most proactive way to manage complexity is to break up the codebase.

The Polyrepo Approach: A Focused and Decoupled

A polyrepo approach is more akin to the point-to-point airline model previously discussed. Instead of consolidating in a single repo, each application component (microservices, manifests, libraries) is version controlled separately. The example "Widgets Co" app might therefore be structured into multiple repos like this:

Each piece of the application lives in its own Git repository. Each repo then has its own version history, pipeline configurations, and access control. Developers gain granularity and tailored focus. Other benefits include:

- **Isolation:** Each service has its own commit history. Changes to a single feature do not require a rebuild or rollback of the entire app. A smaller codebase means newly onboarded developers have a more manageable learning curve.
- **Decoupled CI/CD:** If each repo has its own pipelines, new builds and deploys are faster and more targeted.
- **Granular access control:** Permissions can be managed at the repo level and even outsourced to a managed RBAC tool, rather than having to manage ownership of file paths in a single large repository.

The polyrepo approach encourages teams to specialize and experiment, and alleviate fears of breaking the larger whole. Yet the model does pose some challenges.

- **Dependency management:** Teams must manage tracking and securing inter-repo dependencies, and ensure updates and versions are correct across all projects.
- **Duplication:** Strict modularity is difficult, and duplicated tooling or CI/CD configurations across repos can lead to configuration drift.
- **Coordination overhead:** Teams must invest in communicating and coordinating changes that span multiple repositories.

Companies large and small have found success leveraging the polyrepo approach, particularly for running federated service models. Investing collaboration and dependency management is key to ensure consistency across the many codebases.

Which model works best? Leveraging Harness to manage GitOps complexity

At this point you're probably wondering which model is the "best practice". In short, it depends. Some organizations advocate for and use monorepos with a host of custom tools and integrations, while others prefer the granularity and flexibility of polyrepos.

As a general starting point, a team devoting the bulk of its engineering effort toward a singular core application could benefit from keeping work centralized in a monorepo while they scale. Meanwhile, teams developing a federated service model, or heavily leveraging microservices, could find a polyrepo setup to be a more natural fit.

Whichever model you choose, a robust toolset is needed to manage security and access control, application builds, and deployment pipelines. Harness provides declarative GitOps that connects to your existing source code management system, and leverages ArgoCD to continuously reconcile your repo's declarative state with your live environment. The UI then offers visibility to a monorepo model, while also offering centralized organization if you choose to go the polyrepo route.

Looking ahead

This intro just scratches the surface of GitOps source patterns. Like modern airlines preparing intercontinental route maps, today's software landscape requires intelligent logistical planning and strategic design decisions in order to quickly and reliably deliver features to users. The next posts in this series will dive deep into advanced GitOps patterns and practices.

In the meantime, we encourage you to get started with GitOps with a free Harness account, and then try out some of the guided tutorials to help you take to the skies on your continuous delivery journey.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

[Documentation](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/comparison-guide/buildkite-vs-drone>

Comparison Guide

Harness

Continuous Integration

VS

Buildkite

Harness Continuous Integration vs Buildkite

Continuous Integration

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

Buildkite

Buildkite is a platform for running fast, secure, and scalable continuous integration pipelines on your own infrastructure.

11-50

2013

28.2m

BuildKite is categorized as:

Continuous Integration

â

What is the difference between Harness DevOps Tools Vs. BuildKite?

Updated

November 30, 2023

- Open Source Version
- GitHub Stars
- Self-Service (Simple)
- No Scripting Required
- Container & Cloud-Native
- Traditional App Support
- GitOps (Pipelines as Code)
- Any Source Code Manager
- Containerized Pipelines
- Containerized Plugins
- Secrets Management
- Command Line Interface
- Scalability (Required Infra)
- Admin & Maintenance
- Total Cost of Ownership
- Pricing

Free & Paid

24800

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Vault/KMS/3rd

<yes><yes>

Lightweight

<yes><yes> **.25 FTE**

<yes><yes>

<yes><yes> **Per User**

Paid

540

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<with><with>

<yes><yes>

<yes><yes>

<with><with>

<yes><yes>

Lightweight

<yes><yes> **.25 FTE**

<yes><yes>

<yes><yes> **Per User**

Free & Paid

Paid

24800

540

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<with><with>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Vault/KMS/3rd

<with><with>

<yes><yes>

<yes><yes>

Lightweight

Lightweight

<yes><yes> **.25 FTE**

<yes><yes> **.25 FTE**

<yes><yes>

<yes><yes>

<yes><yes> **Per User**

<yes><yes> **Per User**

Detailed Feature Comparison Harness DevOps Tools Vs. Buildkite

Open source vs. Open core:

Buildkite is not an open-source product. Their agent is, but it stops there. Buildkite offers a free plan specifically for open-source projects, students, and teachers/schools. It also provides deep discounts for charities and non-profits. Regular businesses, however, will pay full price for the product. On the other hand, when Harness acquired Drone, it committed to keeping it open-source forever. Harness recently reaffirmed its investment in the open-source solution with a massive release where a sleeker interface, new visual pipeline builder, governance and security features, and real-time debugging tools were added. While this feature-rich version is free, there is also a paid version of Drone that provides access to enterprise support and more integrations and features yet. Additional features include secrets management options, autoscaling, custom plugins, and more.

Self-Service (Simple):

Buildkite is flexible and customizable, and boasts an easy setup and configuration process. With their agent hooks, it's possible to create custom logic that overrides the tool's innate logic. There are even sample scripts that can be used – but anything else will result in some amount of manual scripting, which isn't ideal. The tool is extensible, with 113 containerized plugins available at the time of this writing. These plugins cover a wide variety of integrations and extensions, and the tool itself covers most popular languages. Additionally, in our CI/CD Buyer's Guide (another great resource for engineers shopping for a new platform), we stated that Buildkite is a CI/CD tool – not just CI – since it can handle deployments as well. Drone offers an easy 'get started' experience where you can be up and running in 5 minutes. Drone also benefits from roughly 150 containerized plugins, profoundly extending the functionality of the tool. Drone scales on demand. All of this means less person hours spent by engineers maintaining the tool or waiting for slowness/downtime to resolve, and more time on what matters: getting that code to artifact.

No Scripting Required:

If you have an extremely simple setup, you can possibly avoid scripting in Buildkite thanks to their agent hooks as described above – but realistically, there will be scripts. While scripting isn't the bane of our existence, it is time that could be spent shipping code. It is toil. Drone is much more intuitive and is mostly configuration as code. Gone are the days of scripting.

Secrets Management:

Buildkite does not offer native secrets management capabilities. They recommend using a third party provider and leveraging plugins to read and expose secrets. It appears that, so far, AWS Secrets Manager, HashiCorp Vault, and GCP Secret Manager are supported through plugins (search 'secrets' to reveal plugin availability). Drone offers encryption on its open-source version. Meanwhile, the enterprise version offers these alternatives: encrypted, native, or external, through third-party providers such as AWS Secret Manager, Kubernetes Secrets, and HashiCorp Vault. No matter how you want your secrets to be handled, Drone can rise to the occasion.

Pricing:

Buildkite enjoys giving back to the community by providing a free plan for open-source projects, students, and teaching organizations. This plan is obviously lacking in features, but provides the basics on an unlimited basis. As far as paid plans go, Buildkite offers a standard plan at \$15 per user per month that includes support and SSO. Lastly, the enterprise plan will run companies \$2999 per month (annual payment only, which means you'll need to fork over ~\$36,000 in one payment), for up to 100 users. Additional users cost \$29 per month. Drone offers an open-source version that is free, and while the enterprise (paid) version does provide an arguably more robust product, the free version is already quite feature-rich and will suffice for many use cases. Download Drone now. To familiarize yourself with enterprise pricing, please contact sales.

**Please note: Our competitors, just like us, release updates to their products on a regular cadence. We keep these pages updated to the best of our ability, but there are bound to be discrepancies. For the most up-to-date information on competitor features, browsing the competitor's new release pages and communities are your best bet.*

Try Harness For Free

Continuous Integration

Interested in seeing what's under the hood? Browse through the Harness Continuous Integration Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/blog/unit-testing-vs-integration-testing>

Unit Testing vs. Integration Testing

Building software can be likened to the intricate art of constructing a car. Think of a car not just as a vehicle, but as a symphony of

meticulously designed parts, each playing its own crucial role. Every part, whether it's a tiny bolt or a major component like the engine, is essential. Each one must be flawlessly crafted and stand resilient on its own. But the true magic happens when all these parts come together in a coordinated dance, working in unison to power the vehicle forward.

Drawing parallels in the software world, this is where unit testing and integration testing come into the spotlight. Much like examining each individual car part, unit testing delves deep into the specifics of each software component, ensuring its independent robustness. On the other hand, integration testing plays the role of the quality inspector, making certain that when these components are pieced together, they interact flawlessly, echoing that harmony we see in a well-oiled machine.

When visualizing the testing strategy for a software development project, one of the most commonly referenced models is the testing pyramid.

While diagrams can paint a neat picture, the real world of software testing is a bit messier. Imagine unit tests, integration tests, and other test categories as different tools in a toolkit. Just like a hammer and screwdriver serve different but complementary purposes, so do these testing methods.

Yes, developers sometimes engage in friendly debates about which test is superior. However, the true goal is to combine their strengths in our continuous integration pipelines. To do this well, it's crucial to really understand what each test brings to the table.

So, Unit Tests vs. Integration Tests. Let's dig deeper into each one and see how they contribute to producing quality software.

Checking Individual Components: Unit Testing

Unit testing is a method where individual parts of a software application, such as functions, methods, or classes, are tested in isolation. The objective is to ensure that each discrete component of the software functions as expected, verifying its logic, behavior, and outputs. By focusing on these smallest testable units, developers can identify and rectify issues at an early stage, ensuring that the foundation of the software is robust.

Within the realm of Test-Driven Development (TDD), the dynamics shift slightly but meaningfully. Here, unit tests aren't just subsequent validators but they lead the development process. These tests act like detailed blueprints, guiding the code that follows and making sure it meets set standards. This proactive approach helps reduce errors and encourages a more planned, forward-thinking development process.

One notable advantage of unit tests is their speed. Given their isolated nature, unit tests don't rely on external components or services to run. This makes them incredibly fast and an excellent choice for immediate feedback loops in development.

Ensuring Cohesive Functionality: Integration Testing

Once we've confirmed that each software piece works on its own, it's crucial to see how they connect and interact. This is where integration testing comes into play. Integration Tests enables you to verify the interactions between different parts or units of the software, ensuring the entire processes operate seamlessly.

However, compared to unit tests, integration tests are inherently more complex. They often require more setup, including background services, and provisioning dynamic test environments. As such, they tend to be slower than unit tests. This isn't a drawback per se but rather a characteristic that developers must be mindful of. While unit tests offer rapid feedback, integration tests provide a deeper assurance, validating the harmony between interconnected components.

Integration Testing vs. System Testing

While there are many other types of testing, a common question we see is what is the difference between Integration Testing and System Testing. System Testing is a higher-level testing phase that focuses on evaluating the entire software system as a whole. This phase examines whether the entire system meets the specified requirements and performs as expected in a real-world environment. System testing is usually performed after integration testing and considered a part of E2E testing.

The Value Of Automated Testing in CI/CD Pipelines

In today's fast-paced tech landscape, delivering software quickly isn't just a goal; it's a competitive advantage. Continuous integration (CI) and continuous delivery (CD) are established practices in software development that streamline the processes of building, testing, and deploying applications. Automated testing is an essential part of CI/CD pipelines. It can help to ensure that software is working as expected, and that it meets all of its requirements.

Here are some specific benefits of using automated testing in CI/CD pipelines:

- **Faster feedback:** Automated tests can be run quickly and repeatedly, which allows for continuous feedback on the quality of the software. This can help to identify and fix bugs early, before they cause problems for users.
- **Reduced risk of defects:** Automated tests can help to reduce the risk of defects in software by catching bugs early and often. By addressing issues upfront, you not only save time but also avoid costly fixes once the software is in production.
- **Improved quality:** Automated testing can help to improve the quality of software by ensuring that it meets all of its requirements. This can lead to a better user experience and increased customer satisfaction.
- **Increased productivity:** Automated testing can free up developers' time so that they can focus on more creative and strategic work. This can lead to increased productivity and a faster time to market for new features.

In Conclusion

In the journey of software development, every phase has its unique challenges and requirements. The importance of maintaining a high standard of quality remains a constant through all these phases. This is where practices like unit testing and integration testing shine, serving as invaluable checkpoints. By integrating these testing methods into CI/CD pipelines, we establish a consistent rhythm of evaluation and improvement, ensuring fast delivery of quality software.

Try Harness To Speed Up Your Test Cycles

Harness CI is the fastest CI solution on the market. Harness can help you cut your test cycle time by up to 80% with ML-based Test Intelligence. It identifies and runs only the tests relevant to your code changes so you only run tests that matter. Coupled with automatic tests splitting and concurrent tests execution, you'll experience faster builds, shorter feedback loops, and significant cost savings.

To experience how automated testing with Harness can speed up your software development and boost productivity, sign up for a free plan or request a demo today.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

[Documentation](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[Case studies](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[We want to hear from you](#)

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

[Sign up for our monthly newsletter](#)

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/introducing-harness-infrastructure-as-code-management>

Introducing Harness Infrastructure as Code Management

Last week, during Harness {unscripted}, we announced the Beta program of a new module - âInfrastructure as Code Management.â In this blog, we will cover the background of investing in this area and why we're so excited about it.â

Background

In the last few years, we have helped hundreds of customers improve their software delivery process with modules like CI, CD, Feature Flags, and Security Testing Orchestration. These customers saw significant improvements in their release process by removing manual steps and adding intelligence to their pipelines. As we were working with these customers, we learned about the need to provide similar functionality to the Infrastructure layer.â

The challenges of using Infrastructure as code at scale

Many of our customers (especially platform engineers) use infrastructure as code solutions, like Terraform, which help them manage resources efficiently and repeatedly, using git as the single source of truth. Terraform is a great tool to do that when there is a small team managing a small number of resources, but as they try to scale and increase the adoption to other teams, they will likely hit several roadblocks:

Learning all these requirements, Infrastructure as code management felt like a natural extension we should be adding to our portfolio - so we did!

How Harness IaCM addresses those challenges:

The product is in its Beta phase, and we're inviting users to join the program and give us feedback. For now, we have decided to support Terraform as the primary IaC tool, but the plan is to go broad and provide support for additional IaC providers (more on that later in this blog).

To address the challenges our customers face, we have built the the following functionality:

Infrastructure Pipeline

Users can create advanced pipelines for infrastructure changes - you can also hook multiple plugins into the flow (like Checkov and tfsec) and run steps in parallel to expedite the execution.

Resource Visibility

â

Users can see the resources they manage, including each resource's attributes and the Terraform-generated output.

Cost Estimation

Users can estimate how each resource's cost will change based on the new configuration.

â

Approval Step

Users can review changes before applying them to the target environment. The approval step clearly shows estimated cost changes. At the resource level, the approval dashboard shows how many changed, deleted, and added resources there are and empowers users to inspect each resource's attribute level changes.

â

PR Automation

Harness will populate all the changes to git, allowing developers to review the resource change as part of the PR process.

OPA rules

Harness embeds OPA as the policy agent for its platform. Use out-of-the-box or custom written OPA rules to validate that the resources in the Terraform plan or state comply with the requirements of the organization (for example - Ensure specific AMIs are used when launching a new VM)

â

State Management

To reduce the burden of managing and hosting the backend for State files, IaCM provides a fully managed state management, inc., the ability to see each state's revision history and compare different revisions.

â

What benefits do Harness customers have?

Existing Harness customers have the unfair advantage of utilizing Infrastructure as Code Management with a low effort - they can leverage the same pipeline, connectors, delegates, and other Harness components currently configured to work with other modules such as CI and CD and utilize them for Infrastructure use cases. This approach reduces the friction and effort needed to onboard the new module. We already have customers that were able to start using the product within just a few minutes!

Harness to support OpenTofu

Harness is a proud member of the OpenTofu community. We plan to support all open-source versions of Terraform and OpenTofu releases.Â

The future

The product is now in its Beta phase and is still on the path to GA with many more capabilities, such as

- Automatic drift detection
- Private Module registry,Â
- Real-time time debugger
- Reports
- Templates
- Additional IaC solutions such as Pulumi, AWS CloudFormation, CDK, etc.
- Local workspace
- and much more!Â

If you'd like to give it a spin, please head over to the module website and sign up - we will work with you to make this happen!

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/harness-joins-opentf-initiative>

Harness joins OpenTF Initiative

Harness joins OpenTF Initiative

We are thrilled to announce that Harness is joining the OpenTF Initiative.

Since 2014, Terraform has been one of the most popular open source projects, having a large community built around it. With a huge

ecosystem of providers, plugins, extensions, and supplemental libraries, Terraform became a critical component for countless number of companies.Â

In order to support the community moving forward, Harness is excited to support OpenTF - a true open-source project that forked out of the last version of Terraform based on the MPL license, with the aim of being included in Cloud Native Computing Foundation (CNCF), to ensure neutral governance and build a vibrant community.

To do that, Harness has committed to provide five of our engineers to work on the OpenTF project and we will be also joining the OpenTF Technical Steering Committee. By doing so, we intend to be very active in the community and collaborate with the other members to benefit from a true open source solution that is being actively maintained and improved.Â

If you are also passionate about this project, we would like to invite you to support the project by creating a pull request on the OpenTF manifesto GitHub repository.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/comparison-guide/cloudbees-vs-harness-ci-comparison>

Comparison Guide

Harness

Continuous Integration

VS

CloudBees

Harness Continuous Integration vs CloudBees

Continuous Integration

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

CloudBees

Cloudbees Enterprise is an end-to-end automated software delivery system that allows companies to balance governance & developer freedom.

501+

2010

121.2m

Cloudbees is categorized as:

Continuous Integration

What is the difference between Harness CI Vs. CloudBees?

CloudBees vs Harness CI | Harness

Updated

November 30, 2023

- **Open Source Version**
- **GitHub Stars**
- **Self-Service (Simple)**
- **No Scripting Required**
- **Container & Cloud-Native**
- **Traditional App Support**
- **GitOps (Pipelines as Code)**
- **Any Source Code Manager**
- **Containerized Pipelines**
- **Containerized Plugins**
- **Secrets Management**
- **Command Line Interface**
- **Scalability (Required Infra)**
- **Admin & Maintenance**
- **Total Cost of Ownership**
- **Pricing**

Free & Paid

24800

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Vault/KMS/3rd

<yes><yes>

Lightweight

<yes><yes>**.25 FTE**

<yes><yes>

<yes><yes> **Per User**

Free & Paid

500

<with><with>

<no><no>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<with><with>

<no><no>

<with><with>

<yes><yes>

Heavyweight

<no><no> **2-5 FTE**

<no><no>

<yes><yes> **Free**

<yes><yes> **SaaS, Hybrid, and On-Premises**

With Harness you can choose between using Harness Cloud (Harness hosted builds), using a SaaS platform with self-hosted build environment, or a fully self-hosted solution

<with><with> **Self-Managed only**

CloudBees CI is fully self-hosted

<yes><yes> **SaaS, Hybrid, and On-Premises**

With Harness you can choose between using Harness Cloud (Harness hosted builds), using a SaaS platform with self-hosted build environment, or a fully self-hosted solution

<with><with> **Self-Managed only**

CloudBees CI is fully self-hosted

<yes><yes> **Cloud-Native**

Harness CI offer modern Cloud-Native enterprise-ready CI solution

<with><with> **Based on Jenkins**

CloudBees CI is based on Jenkins, a traditional automation server, carrying many downsides of the Grandfather of CI such as security vulnerabilities, heavily scripted pipelines, etc

<yes><yes> **Cloud-Native**

Harness CI offer modern Cloud-Native enterprise-ready CI solution

<with><with> **Based on Jenkins**

CloudBees CI is based on Jenkins, a traditional automation server, carrying many downsides of the Grandfather of CI such as security vulnerabilities, heavily scripted pipelines, etc

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Declarative Yaml

Declarative, readable and easy to use yaml syntax that enabled any developer to create and maintain pipelines.

<with><with> **Scripted (most common)**

While declarative yaml is available, most users use Groovy-based domain specific languages with complex scripting, and use scripted templates.

Declarative Yaml

Declarative, readable and easy to use yaml syntax that enabled any developer to create and maintain pipelines.

<with><with> **Scripted (most common)**

While declarative yaml is available, most users use Groovy-based domain specific languages with complex scripting, and use scripted templates.

<yes><yes> **Built in Visual Editor**

Built-in visual editor that is fully integrated with Git, allowing anyone to easily author pipelines.

<no><no> **Inactive Blue Ocean Plugin**

Pipelines are typically defined in a file in a source repository, and edited as text. There is a community project Blue Ocean that provides limited visualization but is no longer maintained.

<yes><yes> **Built in Visual Editor**

Built-in visual editor that is fully integrated with Git, allowing anyone to easily author pipelines.

<no><no> **Inactive Blue Ocean Plugin**

Pipelines are typically defined in a file in a source repository, and edited as text. There is a community project Blue Ocean that provides limited visualization but is no longer maintained.

<yes><yes> **Built in YAML Editor**

IDE-like built-in yaml editor with auto-complete and schema validation, makes it easy to author pipeline as code. The editor is fully integrated with Git

<no><no>

<yes><yes> **Built in YAML Editor**

IDE-like built-in yaml editor with auto-complete and schema validation, makes it easy to author pipeline as code. The editor is fully integrated with Git

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes> **Multiple types of Community plugin**

Harness allows you to leverage many types of community plugins: Drone plugins, Github Actions and Bitrise integrations

<with><with> **Community + Proprietary plugins**

CloudBees CI includes proprietary plugins designed for large organizations, and supports Jenkins Community plugins.

<yes><yes> **Multiple types of Community plugin**

Harness allows you to leverage many types of community plugins: Drone plugins, Github Actions and Bitrise integrations

<with><with> **Community + Proprietary plugins**

CloudBees CI includes proprietary plugins designed for large organizations, and supports Jenkins Community plugins.

<yes><yes>

<no><no>

<yes><yes> **Native and Simple**

Open Policy Agent (OPA) based pipeline policies integrated in the platform, allowing you to meet security needs

<no><no> **with scripted pipelines**

Scripted pipelines are dynamic in nature, making it challenging to set pipeline policies.

<yes><yes> **Native and Simple**

Open Policy Agent (OPA) based pipeline policies integrated in the platform, allowing you to meet security needs

<no><no> **with scripted pipelines**

Scripted pipelines are dynamic in nature, making it challenging to set pipeline policies.

Flexible & Intuitive

<with><with> Limited & Complex

Flexible & Intuitive

<with><with> Limited & Complex

Free & Paid

Harness CI is licensed per developer. Credits for Harness hosted builds are available as add-on.

<with><with> Paid

Free & Paid

Harness CI is licensed per developer. Credits for Harness hosted builds are available as add-on.

<with><with> Paid

<yes><yes> Native and Simple

<with><with> Complex Configuration

<yes><yes> **Low efforts (if any)**

Our SaaS platform take this burden of the customer, while upgrading our self-managed solution require low efforts

<with><with> Efforts intensive

<yes><yes> **Low efforts (if any)**

Our SaaS platform take this burden of the customer, while upgrading our self-managed solution require low efforts

<with><with> Efforts intensive

<yes><yes> with paid plans

<yes><yes>

<yes><yes> with paid plans

<yes><yes>

**Please note: Our competitors, just like us, release updates to their products on a regular cadence. We keep these pages updated to the best of our ability, but there are bound to be discrepancies. For the most up-to-date information on competitor features, browsing the competitor's new release pages and communities are your best bet.*

Continuous Integration

Interested in seeing what's under the hood? Browse through the Harness Continuous Integration Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/products/software-supply-chain-assurance/features>

SBOM Generation and Attestation

Automate SBOM generation in your Harness pipelines

Generate a Software Bill of Materials (SBOM) with every build in either SPDX or CycloneDX format using your preferred tools. SBOMs are attested using your private key to ensure integrity and authenticity, and are stored alongside the image in the artifact repository. The SBOM Generation step is available in both the Build and Deploy stages of the pipeline

Policy Enforcement

Allow only compliant artifacts

Use enforcement policies to proactively mitigate risks by blocking components based on factors such as component name, version, supplier, PURL, and licensing attributes. The Policy Enforcement step is available in both Build and Deploy stages of the pipeline

SLSA Provenance Generation

Build trust in your artifacts

Generate provenance in accordance with Supply-chain Levels for Software Artifacts (SLSA) specifications to achieve Level 2 compliance, thereby improving trust and credibility. SLSA provenance captures data like the repository and branch used to build the artifact, as well as who triggered the build.

SLSA Provenance Verification

Ensure artifact integrity before deployment

Verify SLSA provenance to confirm the integrity of the artifact and assure it was produced as expected, guaranteeing no tampering has occurred. The SLSA Verification step in the Deploy stage can be used to verify the provenance generated by Harness or third-party build systems using OPA policies.

License Compliance

Mitigate legal risks

Manage risks associated with open-source licenses by defining an allowed list of licenses for use within your organization. Use Policy Enforcement to define license usage policies.

Artifact composition and usage view

COMINGÂ SOON

Track component usage to deployment

Gain in-depth understanding of each artifact's composition, including all open-source components used in the artifacts and their deployment environments. Track changes across versions to identify when a new component is introduced.

Component Usage Insights

COMINGÂ SOON

Comprehensive component visibility

Achieve deeper visibility into the usage of open-source components across all your artifacts and deployments. Track key attributes such as license, supplier, package manager, and PURL for each component.

Remediation workflow

COMING SOON

Eliminate blind spots in your remediation effort

Instantly assess the impact of a zero-day vulnerability by identifying which artifacts are affected and where they are deployed. Notify owners, track progress, and generate compliance reports once remediation is complete.

Source URL: <https://www.harness.io/blog/introducing-the-harness-cli>

Introducing the Harness CLI

Why a CLI for Harness?

The Harness UI as well as the underlying Harness API have received critical acclaim from customers for their developer-first UX. Additionally, the Harness Terraform Provider has also become extremely popular among experienced Harness users for automating large-scale onboarding of their applications onto Harness. However, all three interfaces create a certain amount of friction when it comes to learning of Harness concepts for users new to the Harness Platform and its multiple modules. Following are some examples:

- Harness API is the foundation of all interfaces including the Harness UI and the Harness Terraform Provider. However, using the API requires writing new code on the client side using the stubs generated from Harness API Reference Docs.
- Harness UI may require navigating through multiple wizards, sometimes spread across developer-centric setup inside a Project and administrator-centric setup inside the Harness Account.
- Finally, the Harness Terraform Provider requires a basic understanding of Terraform (and its providers) which is another tool a new Harness user needs to be already familiar with.

To solve the above new user learning challenge, we are pleased to announce the public preview launch of the new Harness CLI. As the name suggests, it is a command-line interface (CLI) designed for interacting with Harness and it is fully open source under the MIT license. It lowers the barrier to entry and offers a simple and intuitive way for everyone to access their Harness resources. For developers, it offers the best of both worlds, which is developer-friendly automation while also remaining within the developer's IDE or familiar environment.

Key Features

- Manages the following Harness resources that are necessary for a new user to onboard onto the Harness CD & GitOps module. Coverage for other Harness resources used in other Harness modules is on the roadmap. You can request coverage for these resources as well as other new features through GitHub issues. - Pipeline, Service, Environment, Infrastructure Definition, Connector, Secret, GitOps Application, GitOps Cluster, GitOps Repository.
- Designed with inspiration from popular CLIs in the cloud native ecosystem such as helm and kubectl where input can be via a YAML file with value overrides on the command line. Under the covers, the CLI uses the same Harness API that the Harness UI and Harness Terraform Provider use.
- Support for multiple operating systems: Available for Mac, Linux (both ARM64 and AMD64 architectures), and Windows.
- Open source: The CLI source code is publicly available on GitHub, and we welcome contributions.
- Easy to install and upgrade: A single, lightweight golang-based binary that's only 11MB in size. Add to your \$PATH variable and you are off to the races!
- Full documentation: The CLI has built-in documentation, always available for all commands via a simple --help argument. Additionally, following documentation is also available on the Harness Developer Hub. - Install & configure Harness CLI , Harness CLI Examples, Harness CLI Reference

â

Get Started with Harness CLI - Video Tutorial

Thanks to Nicholas Lotz, our new Developer Advocate, you can get started here by simply watching a video.

â

Get Started with Harness CLI - Step-by-Step Tutorial

You can use the CLI for both push-based CD pipeline deployments as well as pull-based GitOps deployments. For this blog post, you will see how to get a GitOps deployment going with the CLI. This tutorial is also available in the Harness Docs.

Prerequisites

Setup your Kubernetes cluster

You should have access to a Kubernetes cluster. For simplicity, this post will use minikube that is running on the same local machine as the Harness CLI (you will download and install later in this tutorial).

Create free Harness account

You will need a Harness SaaS account to complete this tutorial. It is free to sign up if you don't have one already. After signing up, you will get access to the free tiers of many Harness modules including the Harness CD & GitOps module used in this tutorial.

Install Harness GitOps Agent

Harness GitOps Agent is a worker process that runs in your Kubernetes cluster and performs all the GitOps tasks you request in Harness. It makes secure, outbound connections to Harness Platform so that you don't have to open up ports to the Internet from your Kubernetes cluster.

Download Harness GitOps Agent from your Harness account

First you have to download the agent from the Harness UI. Click on GitOps â Settings â GitOps Agents â New GitOps Agent. Let's assume you do not have any existing Argo CD instance running in our Kubernetes cluster, so you will choose the NO option to the question "Do you have any existing Argo CD instances?" and click Start. In the next screen, enter *mygitopsagent* for the agent name and *mygitopsns* for the agent namespace.

Click Continue and then download the gitops agent helm chart (with Argo CD included) in the form of a tgz file.

Install Harness GitOps Agent on your Kubernetes cluster

Now that you have downloaded the agent, it is time to install it on our Kubernetes cluster which in this case is the local minikube. You will first create the namespace where you will install the agent. And then you will install the agent there using the helm install command.

You can check the status of the installation using the command below. Notice that your installation also includes an instance of Argo CD since that's the option you chose during the agent download from Harness UI in the previous step.

Set the environment variable `AGENT_NAME` to `mygitopsagent` so that we can use it during the later steps.

Install Harness CLI on your local machine

Download the CLI as shown below. Make sure to use the binary specific to your operating system.

Now login to your Harness account using the CLI using your account id and your API key. Account id is available on every Harness URL as well as your Account Settings page. Create a new personal API key from your User Profile page using these instructions.

After successful login, the API key and account id are stored locally on your machine and used by CLI for subsequent operations.

Use Harness CLI to create Harness GitOps resources

Fork <https://github.com/harness-community/harnesscd-example-apps> into your own GitHub account and then clone it to your local machine.

Now, you will create the Harness GitOps repository, cluster and application using the sample YAMLs provided for the guestbook app. Before that, you need to make the following edits to the YAMLs.

Perform Manual GitOps Sync via Harness UI

On the Harness UI, go to GitOps → Applications. Click on the gitops-application tile and then click SYNC at the top right to initiate a sync between the state of your GitOps application in Git and the state of your Kubernetes cluster. Since the application did not exist in Kubernetes till now, the application will get deployed now. You can check the status using the command below as well as from the UI.

Setup Automated Sync for Future Changes

Go to App Details in the *gitops-application* and scroll down to Sync Options & Policy. Toggle it to turn on Automated sync. Any changes to your Git repo will now be automatically synced to your Kubernetes cluster.

The Road to GA

Using the CLI to simplify Harness CD & GitOps onboarding for new users is just the start. We plan to enhance the CLI to cover all Harness resources that are managed by the Harness API. This will bring the same simple onboarding experience to other Harness modules as well. Meanwhile, your feedback goes a long way in helping us prioritize the most important use cases, especially as we aim to get the CLI to a GA stage over the next few months. You are welcome to file GitHub issues, ask questions on our Community Slack or send in contributions via Pull Requests.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer

happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/harness-announces-chaos-carnival-2024>

Harness Announces the 4th Edition of Chaos Engineering Conference focusing on Software Resilience

Get ready to dive deep into the excitement of chaos engineering! As Harness launches the 4th edition of Chaos Carnival in 2024, we're set to redefine the boundaries of software resilience in today's tech landscape. Uncover groundbreaking chaos studies, connect with developer visionaries, explore state-of-the-art open-source and enterprise solutions, and stay ahead of the latest developments in software resilience. Whether you're embarking on your chaos journey or looking to elevate your game, this is where transformative insights await!

Uniting Innovators Across the Globe

Dive into the dynamic world of Chaos Carnival, our annual virtual conference that's been setting the stage for Chaos Engineering since 2020. It's not just a conference â it's where the brightest minds in chaos engineering converge. Join a diverse group of CTOs, engineers, developers, pioneers, and thinkers, all tuned into the pulse of chaos innovation.

Riding high on the theme "Build Resilience Through Chaos," we emphasize not just the theory but the vital application of Chaos Engineering in navigating the unpredictable digital landscape we encounter today.

The power of chaos engineering? It enables businesses to channel their energy into groundbreaking innovations, leaving the dread of system failures behind. With advanced tools in chaos engineering, proactive enhancement becomes the norm, ensuring that software remains reliable and resilient.

Jyoti Bansal, CEO of Harness, sums it up: "Chaos Carnival isn't just a conference; it's a global conversation. The surging demand for software reliability and resilience underscores our mission â to foster expansive dialogues and cultivate a thriving community of forward-thinking developers."

Elevate Your Journey: Chaos Carnival 2024 Awaits!

Unlock a world of opportunities with this yearâs enhanced Chaos Carnival. We're rolling out two tailor-made tracks: Chaos Beginner for those embarking on their chaos journey and Chaos Expert packed with inspiring success tales, real-world use cases, strategies for enterprise adoption, thrilling GameDays, and beyond.

Be at the forefront of innovation as you immerse yourself in insights from four trailblazing keynotes, 30+ engaging talks, hands-on workshops, and riveting panel discussions. Dive deep into open-source software, like LitmusChaos, multi-faceted programming languages, cutting-edge product development, cloud innovation, and more. Yes, even the world of the renowned Chaos Monkey!

Chaos Carnival isn't just another event. It's where the global chaos community gathers to challenge, collaborate, and change the future.

Unleash Your Potential: Register & Contribute!

Why just attend when you can shape the conversation? Whether you're an SRE, developer, QA wizard, Cloud Architect, or any trailblazer in resilience practices, now's your time. Not only is registration completely complimentary, but we also invite you to amplify your voice by submitting a CFP.

Your Spotlight Awaits: Speak at Chaos Carnival 2024

From first-time speakers to seasoned experts, Chaos Carnival is the stage to share your passion. Illuminate the audience with your insights in breakout sessions, lightning talks, or stirring panel discussions. Hurry, the window for speaker applications closes on December 1st, 2023! Submit your CFP today!

Elevate Your Brand: Sponsor Chaos Carnival 2024

Align with excellence. As our sponsor, place your brand at the epicenter of innovation, gaining unparalleled exposure to a diverse global audience passionate about chaos engineering, open-source dynamics, cloud-native technologies, and beyond. This is your golden ticket to synergize with the best in the industry. For exclusive sponsorship opportunities, reach out to us at chaoscarnival@harness.io.

Elevate Your Game: Spark Software Resilience & Dive into Our Global Community

Curious about what's in store for 2024? Last year's Chaos Carnival garnered a staggering turnout with 3,200+ global participants and featured tech giants, including AWS, Google, Azure, Aqua Security, Mitigant, Blameless, Nagarro, Grafana, Cloud Native Computing Foundation, and many more.

Jumpstart your 2024 by being among the earliest to register and stand a chance to snag an Amazon gift card (and, of course, some serious bragging rights)! ☺

â

Stay Tuned and Amplify Your Connection:

ð LinkedIn

ð YouTube

ð Twitter

ð Facebook

â Harness Chaos Engineering

Join the momentum. Shape the future. See you at the Carnival! ðð

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/a-b-testing-for-feature-flags-what-it-is-and-what-it-shouldnt-be>

A/B Testing For Feature Flags, What It Is And What It Shouldn't Be

Intro

We talk to teams daily that have A/B testing on the list of their needs from a feature flag tool. However, what A/B testing is and where you may want to get it from usually becomes a more nuanced conversation than just âincludedâ or ânot included.â

In this post, we want to look at what people mean when they talk about A/B testing and share how we think about it in relation to feature flags.

Two Meanings of A/B Testing

There are two pretty different things people can mean when they talk about A/B testing in the context of feature flags:

- Engineering verification. This is when teams want to use their production environment as a safe test environment in order to see the performance or cost impact of a change and compare it to their baseline version to quantify the impact of a change before progressing it to a wider release. This is a great use case for feature flags.
- Behavioral analytics. This is when a team is trying to find out if a blue button converts better than a red button or if a different layout keeps people on the page longer. Other than being *implemented via a feature flag* we think that coupling this with aÂ *feature flag vendor* is the wrong way to go.

For the rest of this post, weâll focus on the confusion around the second point. In a future post, weâll talk more about the engineering side of A/B testing and the process of verifying the impact of your code changes on your overall systemâs performance and cost.

High-Value Feature Flags And High-Value A/B Tests

Letâs take this as an assumption - you want to get the absolute most value possible out of both your feature flags and your A/B tests.Â

To do that, it helps to understand that they were serving two different purposes, and often for two different people in the organization.

- Feature flags are a product and engineering process, breaking release apart from deployment in order to accelerate velocity, improve customer experience, and empower organizations to release more and run higher quality software.
- A/B testing, in the behavioral analytics sense, is about learning and optimizing the user experience for business goals. Itâs usually for product management and marketing, and the KPIs are usually growth or revenue related, not velocity, performance, and resiliency related.

A/B tests are implemented pretty similar to a feature flag - itâs essentially just a diff in the code serving one path vs. another conditionally - but the concerns before and after the implementation for the two are pretty different.

With feature flags, you want governance, developer experience, release automation, reporting, and lifecycle management. With behavioral analytics A/B testing, you want data science and growth/revenue-based correlations. This often involves delving into topics like true positive rates and false discovery rates and understanding the difference between type two error scenarios and accurate results.

From a user experience, itâs very unlikely that a tool that provides a great experience for engineering is also providing a great experience for marketing. These are significantly different audiences with different lifecycle concerns and different optimizations needed.

However, because the implementation is so similar, you do see on the market companies that bolt them together. This results in great A/B testing companies with very limited feature flag offerings or feature flag companies with very minimal, hard-to-use A/B testing offerings. We donât like either approach.

Our StrategyÂ

On top of the overall differences between feature flags as an engineering process and A/B testing as a growth and revenue process, we also find that increasingly, most tools have world-class analytics tools in place anyway and donât need more data siloing and more tools to log into.Â

So, hereâs how we see it - you should use the best data analytics tools in the world for A/B testing, and you should be able to implement those tests via Feature Flags that are focused on absolutely maximizing the value of feature flags in your software delivery process.

Increasingly, best-of-breed analytics tools with A/B testing - such as Amplitude Experiments, Statsig, and Growthbook - allow for Segment implementations of the data payload needed to run their experiences. We are connecting our Feature Flags to Segment to automate the process of using Harness Feature Flags with any of the best A/B testing analytics vendors on the market, allowing you to keep your data all in one place and letting you use the best tool for each job. Even until itâs fully automated, adding a simple segment call to your flags is simple and adds minimal overhead for the value it unlocks.Â

At Harness, we are laser focused on solving the problems associated with software delivery. It can be tempting to drift into adjacent areas, but when the problems are far apart, you deliver weak offerings that donât satisfy the end users â and thatâs just not our approach.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/comparison-guide/jenkins-vs-drone>

Comparison Guide

Harness

Continuous Integration

VS

Jenkins

Harness Continuous Integration vs Jenkins

Continuous Integration

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

Jenkins

Jenkins is an open source automation server with an unparalleled plugin ecosystem to support practically every tool as part of your delivery pipelines.

1-50

2012

â

Jenkins is categorized as:
Continuous Integration

â

What is the difference between Harness DevOps Tools Vs. Jenkins?

Jenkins vs Drone: DevOps Tools Comparison

Updated

November 30, 2023

- **Open Source Version**
- **GitHub Stars**
- **Self-Service (Simple)**
- **No Scripting Required**
- **Container & Cloud-Native**
- **Traditional App Support**
- **GitOps (Pipelines as Code)**
- **Any Source Code Manager**
- **Containerized Pipelines**
- **Containerized Plugins**
- **Secrets Management**
- **Command Line Interface**
- **Scalability (Required Infra)**
- **Admin & Maintenance**
- **Total Cost of Ownership**
- **Pricing**

Free & Paid

24800

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Vault/KMS/3rd

<yes><yes>

â**Lightweight**

<yes><yes> **.25 FTE**

<yes><yes>

<yes><yes> **Per User**

Free

17000

<no><no> Scripting TOIL

<no><no>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<with><with>

<no><no>

<with><with>

<yes><yes>

<no><no> **Heavy Weight**

<no><no> **2-5 FTE**

<no><no>

<yes><yes> **Free**

Free & Paid

Free

24800

17000

<yes><yes>

<no><no> Scripting TOIL

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<with><with>

<yes><yes>

<no><no>

Vault/KMS/3rd

<with><with>

<yes><yes>

<yes><yes>

âLightweight

<no><no> Heavy Weight

<yes><yes> .25 FTE

<no><no> 2-5 FTE

<yes><yes>

<no><no>

<yes><yes> Per User

<yes><yes> Free

Open source vs. Open core:

Jenkins was created in 2012 as open-source software, so there is no associated software price. Drone is also open-source. There is, however, a paid version of Drone that provides access to enterprise support and more integrations and features. Additional features include secrets management options, autoscaling, custom plugins, and more.

Self-Service (Simple):

Jenkins does it all â but at a cost. That cost is a large investment of time and effort (and FTEs). Weâll give credit where itâs due: Jenkins is quite customizable and flexible â both important things. But, it also has a high learning curve. To configure Jenkins, youâll most likely need to depend on plugins and scripts, both of which require maintenance and can open engineers up to dependency hell. Drone is built upon three pillars that enable engineers to build and test code quickly and accurately: simple, scalable, self-service. Drone installs in under 5 minutes, scales on demand, and all plugins run in containers on their latest version. This means less person hours spent by engineers maintaining the tool, and more time on what matters: getting that code to artifact.

Requires Scripting:

As mentioned above, Jenkins Scripting will require Java, Groovy, Javascript, Golang, Ruby or Shell knowledge. The more complex the setup, the more scripts/plugins needed. Drone is much more intuitive.

Container & Cloud-Native:

Jenkins wasnât designed to be cloud-native, but developers and DevOps teams have been making it work for years. There is a group of contributors and collaborators focusing on improving Jenkinsâ cloud capabilities. For instance, theyâve created a Jenkins Kubernetes operator. While itâs definitely progress, albeit slow, we frankly donât believe that CI/CD should be that hard. Jenkinsâ architecture has a clear audience in mind. From their docs: âThe audience is Java developers familiar with Jenkins (as users) who want to understand how Jenkins works internally.â Not that Java architectures canât be cloud native but itâs an architecture that wasnât designed with that in mind.

GitOps:

For GitOps on Jenkins, weâd recommend evaluating Jenkins X because it has built-in GitOps. Itâs possible to do it on Jenkins, but harder to set up. Drone comes with built-in GitOps functionality.

Containerized Plugins:

Jenkinsâ plugin index boasts almost 1800 plugins â with many being abandoned by their authors, unmaintained, or duplicated (ie, 10 plugins that do the same thing). Droneâs 150 plugins are much more manageable. All plugins are containerized and maintained to their latest version. No dependency hell, no updating â only simplicity and portability. Treat your Drone plugins as cattle. Scale them elasticity at will, kill idle ones as easily.

Secrets Management:

Jenkins does not offer native secrets management capabilities. There are many ways to do it through a third party, such as HashiCorp Vault or Helm Secrets. Drone offers encryption on its open-source version. Meanwhile, the enterprise version offers these alternatives: encrypted, native, or externally, through third-party providers such as AWS Secret Manager, Kubernetes Secrets, and HashiCorp Vault. No matter how you want your secrets to be handled, Drone can rise to the occasion.

Scalability:

Jenkins is less scalable than Drone, and hereâs why. Jenkins instances have to be manually spun up, while Drone has autoscalers (through Amazon EC2, Digital Ocean, Google Computer, Hetzner, Open Stack, and Packet). Jenkins also takes up more server space than Drone. Jenkins, in and of itself (and compared to other tools), may not be heavyweight â but when compared specifically to Drone, it is.

Admin & Maintenance:

Jenkins is one of the most maintenance-intensive CI tools on the market, which is expected from a ten year old product. From our clients' data, we've extrapolated that Jenkins requires between 2-5 FTEs to maintain and administer every year. That time is spent writing scripts, doing research on plugins, maintaining and updating plugins, etc. Drone stamps out maintenance issues at a whopping .25 FTEs needed (that's not a typo). It's an extremely portable solution without scripting, plugin maintenance, or dependency hell or much else in terms of maintenance, for that matter. The plug-and-play nature of Drone ensures the only work you have to perform, other than the initial setup and configuration of course, is administration such as adding and removing users, permissions, etc.

Total Cost of Ownership:

Free doesn't always mean free. While Jenkins itself doesn't cost a cent, it does cost a pretty penny in terms of employee need and time. As mentioned above, our clients have stated it requires between 2-5 FTEs to keep Jenkins up and running. Let's say your average engineer costs you \$180,000, that's almost a million dollars per year dedicated to a single piece of software. Drone, however, only costs .25 FTEs, which on our \$180,000 example, results in \$45,000. A bit of a difference in total cost of ownership.

Pricing:

Jenkins is open-source so there is no pricing. Drone is free and available for download. It also has an enterprise version that is extremely feature-rich, but does have pricing attached to it. To familiarize yourself with enterprise pricing, please contact sales.

**Please note: Our competitors, just like us, release updates to their products on a regular cadence. We keep these pages updated to the best of our ability, but there are bound to be discrepancies. For the most up-to-date information on competitor features, browsing the competitor's new release pages and communities are your best bet.*

Try Harness For Free

Continuous Integration

Interested in seeing what's under the hood? Browse through the Harness Continuous Integration Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/comparison-guide/github-actions-vs-drone-devops-tools-comparison>

Comparison Guide

Harness

Continuous Integration

VS

GitHub Actions

Harness Continuous Integration vs GitHub Actions

Continuous Integration

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration

GitHub Actions

GitHub Actions works to integrate code provided by your team in a shared repository. Developers share the new code in a merge request (MR).

1000+

2014

434.2m

GitHub is categorized as:
Continuous Integration

â

What is the difference between Harness CI Vs. GitHub Actions?

GitHub Actions vs Drone: DevOps Tools Comparison

Updated

November 30, 2023

- **Open Source Version**
- **GitHub Stars**
- **Self-Service (Simple)**
- **No Scripting Required**
- **Container & Cloud-Native**
- **Traditional App Support**
- **GitOps (Pipelines as Code)**
- **Any Source Code Manager**
- **Containerized Pipelines**
- **Containerized Plugins**
- **Secrets Management**
- **Command Line Interface**
- **Scalability (Required Infra)**
- **Admin & Maintenance**
- **Total Cost of Ownership**
- **Pricing**

Free & Paid

24800

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Vault/KMS/3rd

<yes><yes>

Lightweight

<yes><yes> **.25 FTE**

<yes><yes>

<yes><yes> **Per User**

Free & Paid

-

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<with><with>

<yes><yes>

<yes><yes>

<with><with> **2 FTE**

<no><no>

<yes><yes> **Per GitHub product and a OS multiplier**

Free & Paid

Free & Paid

24800

-

<yes><yes>

Vault/KMS/3rd

```
<with><with>  
<yes><yes>  
<yes><yes>  
Lightweight  
<yes><yes>  
<yes><yes> .25 FTE  
<with><with> 2 FTE  
<yes><yes>  
<no><no>  
<yes><yes> Per User  
<yes><yes> Per GitHub product and a OS multiplier
```

Open source vs. Open core:

As counterintuitive as it may sound, almost nothing in GitHub is open source. It may be one of the biggest open source hubs but its own tools are closed source. Drone, on the other hand, was and will always be open source. To quote Jyoti Bansal, Harness founder and CEO, at the time of the acquisition "Drone will always remain open-source, and Harness will invest significantly over the coming years to its community, platform, and mission. In addition, Harness will also contribute several internal projects later this year to the open-source community." Drone has changed the way we did CI and it can do the same for your company.

Self-Service (Simple):

GitHub Actions will recommend CI workflows based on the language and framework in your repository, which is nice but far from self-serve. A true self serve experience especially if you use an SCM tool different from GitHub. Being SCM agnostic has advantages since GitHub Actions is finely tuned to listen to events that happen in GitHub SCM and maybe not so much to your SCM tool of choice. Testing with GitHub Actions doesn't go beyond providing a test result on the Pull Request created. Drone offers an easy àget startedâ experience where you can be up and running in 5 minutes. Drone also benefits from roughly 150 containerized plugins, profoundly extending the functionality of the tool. Drone scales on demand. All of this means less person hours spent by engineers maintaining the tool or waiting for slowness to resolve, and more time on what matters: getting that code to artifact.

Any Source Code Manager:

While GitHub Actions is not limited to GitHub SCM it is indeed finely tuned to it so, the experience of using GitHub Actions with any other source code management provider will differ considerably. Drone, luckily, is vendor-agnostic and allows you to work with your preferred SCM, whether GitLab, GitHub, Bitbucket, or whatever else you choose.

Secrets Management:

GitHub Actions extends secret management capabilities from GitHub itself making it tough to do the same if you are hosting your code elsewhere. Drone offers encryption on its open-source version. Meanwhile, the enterprise version offers these alternatives: encrypted, native, or external, through third-party providers such as AWS Secret Manager, Kubernetes Secrets, and HashiCorp Vault. No matter how you want your secrets to be handled, Drone can rise to the occasion.

Testing and verification:

GitHub Actions comes with a set of out of the box test suites.Â They are pretty standard and apply to any language. Harness CI does so too but goes beyond that. Test Intelligence is one of Harness CI's most advanced features. It'll parse the test suite once to get a snapshot of it all. With that it'll be able to apply efficiency gains each time we run the suite by applying only the tests that have changed, run tests to incremental source code changes and, most importantly for security compliance, run test in order of flakiness, meaning, it'll run first those that are more prone to fail so that immediate action can be taken faster. These are just a few of the gains an advanced testing feature like Test Intelligence can provide to a modern CIÂ system compared to GitHubÂ Actions.

Total Cost of Ownership:

When comparing products, it is imperative to take all costs into account. With a product like GitHub, you're not just paying for the tool â you're paying for the 2 FTEs it takes to keep the tool in working condition. It's a large cost to take into account. With Drone, the end result is a tool that is very affordable, with a small commitment of .25 FTEs and low or free pricing (depending on if you decide the features for the enterprise plan are too good to pass up â who doesn't want autoscaling?).

Pricing:

GitHub Actions is freely available for public repositories but private ones have a 2000 minutes limit on their runner's build time. More

minutes are available in paid for plans. If a customer goes beyond a plan's allotted minutes, they're billed per minute based on the runner platform. Self-hosted runners are free and the CI/CD capabilities are also available in the GitHub Enterprise self-hosted version. Drone does offer an open-source version that is free, and while the enterprise version does provide an arguably more robust product, the free version is already quite feature-rich and will suffice for many use cases. Download Drone now. To familiarize yourself with enterprise pricing, please contact sales.

**Please note: Our competitors, just like us, release updates to their products on a regular cadence. We keep these pages updated to the best of our ability, but there are bound to be discrepancies. For the most up-to-date information on competitor features, browsing the competitor's new release pages and communities are your best bet.*

Try Harness For Free

Continuous Integration

Interested in seeing what's under the hood? Browse through the Harness Continuous Integration Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/blog/august-2023-product-updates>

August 2023 Product Updates

At Harness, we have been hard at work so you can delight your customers without facing software delivery toil. Here is what has been changing in the previous month across Harness Products.

Continuous Delivery & GitOps

- You can now edit Git details after the pipeline is configured and saved. This can be very useful in Git Experience workflows. For example, this enables you to move your YAML configs from one location to another in your Git configs repositories.

Full Continuous Delivery & GitOps Release Notes

Continuous Integration

- **Eliminate unnecessary connection tests for GitHub connectors:** Harness has updated how it handles connection tests for your GitHub connectors. Before, it would continually test connections even if they kept failing, which could be problematic.
- **What's new and why this change is beneficial:** a. If a test fails due to incorrect GitHub credentials, Harness will stop the tests. b. This avoids needless testing and can prevent issues like account lockouts, especially when a connector's access token is tied to an individual user's account. c. In simple terms, Harness is smarter about knowing when to stop trying a connection, saving resources and potential account troubles. d. This change makes managing connections more efficient and less prone to problems.

â

Full Continuous Integration Release Notes

Feature Flags

- **Feature Flags UI:** When specifying percentages for a rollout, the UI now provides feedback while you edit to let you know the percentage that requires assignment.

Full Feature Flags Release Notes

Cloud Cost Management

- **AWS AutoStopping proxy enhancement:** You can now select the subnet ID from the dropdown list for AWS AutoStopping proxy creation.
- **Auto Stopping page UI enhancement:** The page on which users select either a load balancer or an AutoStopping Proxy has been enhanced to include an additional API that retrieves information about proxies created previously in shared VPCs. Now users can use a proxy created in a shared VPC across projects and connectors.
- **Overview page UI enhancement:** a. The pie chart now shows a hover state. b. The forecast trend in the widget has been removed c. Added forecast time period in the forecast cost widget d. Changed the heading of the cloud cost widget from Top AWS

Accounts to Top 10 AWS accounts.

- **Recommendations page UI enhancement:** The Include dropdown on the Recommendations page has been removed. Instead, the following toggle options have been added in the Filter panel as shown in the screenshots below:

Full Cloud Cost Management Release Notes

Chaos Engineering

- Now supports Universal Base Images (UBI) in chaos components.
- Improved safety: Users can't edit or delete scheduled chaos experiments if the linked system isn't active.
- New storage option for keys and tokens: a. Choose between Config Maps or Secrets on the execution plane. b. Useful for storing connection details, authentication, and running experiments.

Full Chaos Engineering Release Notes

Internal Developer Portal (IDP)

- **Internal Developer Portal (IDP) Plugin Updates:** a. Added the Confluence search plugin to retrieve results from Confluence spaces. b. Introduced the Harness Feature Flags plugin. c. Launched an updated Harness CI/CD plugin. d. Features new annotation support. e. Enables filtering of pipelines across projects and organizations.
- **Polling Updates for Software Catalog:** IDP now checks linked Git repositories every 15 minutes, up from the previous 5-minute interval.
- **IDP Catalog API Access:** Use the Harness X-API-Key to interact with the IDP catalog APIs.
- **Software Template Changes:** The 'trigger:harness-custom-pipeline' action in the 'template.yaml' file is now synchronized with pipeline activity: a. It continues to run and displays the ongoing pipeline status. b. With the synchronous nature of 'trigger:harness-custom-pipeline', it's possible to use the 'catalog:register' action in a template to register the 'catalog-info.yaml' of the newly created software component.

Full Internal Developer Platform Notes

Software Engineering Insights

- **Baselines:** Users now have the ability to define a baseline and have a trend line for the custom table report.
- **Trellis Scores:** Users now have the ability to view the Trellis scores for different intervals as a 'Public Dashboard' user and 'Org Admin' user.
- **Pipeline Support Update:** Support for Azure DevOps Release pipelines.
- **Custom Reporting:** Ability to render the Executive Insights widget as a custom report using propels.
- **Widgets:** Drill-down support for single stat for Lead Time in Stages widget.

Platform

Here are some important improvements to our platform that should enhance your user experience:

- **Delegate Selection Updates:** Delegate selection logs now show the following: DelegateId, DelegateName, Hostname.
- **Go Library Upgrade:** Updated from version 1.20.4 to 1.20.5.
- **Harness AIDA Integration:** a. Introduced "Ask AIDA", an AI-driven chatbot for Harness Docs. b. Access AIDA by clicking the icon at the bottom-right of the screen.
- **Delegate Logs for Connectors:** View delegate logs while validating a connector that uses a delegate connection.
- **OAuth Improvements:** For accounts with Oauth enabled: Auto-acceptance of invites is now available, even if using password-based authentication.
- **Username Restrictions:** Usernames are limited to 256 characters.
- **List Tokens API Enhancements:** a. Ability to view all personal access tokens or service account tokens in an account. For those with user management permissions: List all personal access tokens, Filter tokens by user or show only active tokens. c. For those with service account management permissions: List all service account tokens, Filter tokens by service account or show only active tokens.

Full Platform Release Notes

Delegate

- **Delegate Connectors:** The Splunk connector has been enhanced to include support for Bearer Token.
- **Delegate Execution Logs:** Enhanced Execution Logs with added details: Duration, Task ID, Additional information useful for understanding and troubleshooting CV Steps, SRM Live monitoring, and SLI.
- **Delegate Manual Query mode:** In manual Query mode, the Datadog Metrics Health source now provides support for formulas.
- **Error Messages:** Health source provider error messages are now included in API responses, enhancing user experience and making debugging more efficient.
- **Azure API Update:** New API, getAzureKeyVaultClient, available: a. Fetches the list of Azure vaults. b. Speeds up the time for Harness to display a recently created Azure vault.

Self-Managed Enterprise Edition

- **New Database support:** Harness now supports external self-managed databases for high availability.

Full Self Managed Enterprise Release Notes

Early Access

Platform

- **Harness AI Development Assistant Launch (Beta Feature):** Harness has introduced the "Harness AI Development Assistant (AIDA)" to transform software delivery processes.
- **Purpose and Functionality:** a. Merges AI capabilities with established DevOps tools and practices. b. Streamlines and hastens the software development lifecycle. c. Enables teams to release high-quality applications swiftly and effectively. d. Features AI-driven predictive analytics, continuous verification, and advanced release orchestration.
- **Key Benefits of Harness AIDA:** a. Auto-recognition of failures in pipelines: The root cause analysis (RCA) option generates recommendations for step failures in pipelines. b. Asset governance: The asset governance feature assists you in drafting rules that are based on your requirements and aligned with your governance goals. c. Security: Harness AI identifies security vulnerabilities, describes them, and suggests remediation.

Continuous Delivery

- **Blue Green Deployments:** Added a new field in the release history for Blue Green deployments to differentiate between environments. You'll need to enable the CDS_BG_STAGE_SCALE_DOWN_STEP_NG to use this feature.
- **Scheduled automatic approvals have been added to manual approval steps:** This functionality is behind a feature flag, CDS_AUTO_APPROVAL. You can configure a manual approval step to automatically approve at a specific date and time.
- **Docker Artifact Digest Support:** Digest support was added for Nexus 3, Github, and Artifactory artifact sources. This feature is behind the feature flag CD_NG_DOCKER_ARTIFACT_DIGEST.

Continuous Integration

- **Using Output Variables as Environment Variables in Build Stages:** A new feature allows output variables from one step to be used directly as environment variables in subsequent steps within the same Build (CI) stage. **How to Activate:** Turn on the feature using the CI_OUTPUT_VARIABLES_AS_ENV flag. **Practical Example:** In a Build stage with three sequential steps, an output from the first step can be directly utilized in the second and third steps. **Easy Reference:** With the feature active, you can use an output variable just by its name. For instance, MY_VAR from an earlier step can be accessed as \$MY_VAR in later steps. **Without this feature:** You'd need to use a more complicated expression like <+steps.stepID.output.outputVariables.MY_VAR> to refer to the variable.
- **Cache Intelligence UI update:** You can enable Cache Intelligence in the Pipeline Studio's Visual editor. Previously, you could only enable Cache Intelligence through the YAML editor. a. The Enable Cache Intelligence UI field is behind the feature flag CI_CACHE_INTELLIGENCE b. For more information on this feature, go to the documentation on Output variables.

Cloud Cost Management

- **Propagate force cool down:** You can now propagate force cool down from primary rule to dependent rules. Earlier, when stopping a rule from the UI, you had to stop its dependent rules one by one. With this enhancement, you can propagate the stop operation to dependent rules as well.

Security Testing Orchestration

- **Dynamic target baselines:** You can now define dynamic target baselines using regular expressions. Dynamic baselines more accurately reflect the current "root" element in the context of a real-world software development life cycle. This feature is behind the Feature Flag STO_BASELINE_REGEX.

Full Early Access Release Notes

Continue the Journey

- Learn more hands-on with our increasing list of Tutorials.
- Further your knowledge with one of our Certifications.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/?88376de9_page=2

The Modern SoftwareÂ Delivery PlatformÂ®

Developer Experience Unleashed

How Harness supercharges your developer experience

Build fast and ship frequently

Next Generation CI/CD

4x Faster Builds

One-click deployments

Fully automated pipelines

AI/ML driven deployment verification

Continuous DeliveryÂ &Â GitOps

Continuous Integration

Code Repository

Infrastructure as CodeÂ Management

Integrate security at everyÂ step

Shift Left Security

Aggregate security scans in CI/CD pipelines

Vulnerability de-duplication & prioritization

AI/ML driven vulnerability remediation

SBOM orchestration & SLSA attestation

Security Testing
Orchestration

Software Supply
Chain Assurance

Improve resilience and quality

Continuous Resilience

Automated SLO management

Chaos engineering for resilience testing

Quality gates in pipelines

AI/ML driven change impact analysis

Chaos
Engineering

Service Reliability
Management

Continuous
Error Tracking

Release features confidently

Feature Management

Manage flags with GitOps experience

Build Feature Flag pipelines

Manage Flag Tech-Debt

Feature Flags

Accelerate developer onboarding

Platform Engineering

IDP for service catalog & scorecards

Rapid onboarding with golden paths

Built on Open Source - Backstage

Internal
DeveloperÂ Portal

Drive cost & process efficiency

Cost & Process Optimization

Cloud cost insights

Actionable intelligence across the SDLC

Automatically stop idle resources

OOTB DORA, planning, and execution metrics

Harness AIDA

Leverage AI across the software delivery lifecycle: code, build and test, secure, deploy, manage costs, and ensure resilience.

Engineering excellence platform for the enterprise

Hundreds of DevOps and engineering teams are powered by Harness to become elite performers in velocity, quality, efficiency, and governance.

Citi improves software delivery performance with Harness

Harness CD lets us release each change within minutes of a pull request being merged. Running all the necessary tests and scans during the pipeline gives us the confidence to do this.

Stefanos Piperoglou, Technical Program Manager, Citi

United Airlines Accelerates Deployments by 75% With Harness

By choosing Harness for CI and CD, we were able to give the governance policies to the developers and create the guardrails we needed. Harness gives us a platform rather than just a DevOps tool.

Ratna Devarapalli, Director of IT - Architecture, Platform Engineering & DevOps

Ancestry adds consistency and governance to cut downtime and systems onboarding effort

Harness now empowers Ancestry to implement new features once and then automatically extend those across every pipeline, representing an 80-to-1 reduction in developer effort.

Ken Angell, Principal Architect, Ancestry

Try Harness for free today

Source URL: <https://www.harness.io/blog/advanced-deployment-strategies>

Advanced Deployment Strategies

DevOps aims at automation and bridging the gap between Dev and Ops. Many organizations embrace DevOps practices to be ahead of their competition. It sets the stage for the development and operation teams with best practices for building and deploying applications/features.

When developers create applications and features, there is one thing that needs attention - How are they going to deploy it? The strategy they are going to use, etc. Answers to these things are important as they might directly impact customers. Some organizations might use a rolling deployment strategy, which is more of a default strategy that comes even along while deploying on Kubernetes. It is time; we need to understand what lies beyond just rolling deployment strategy and explore the best possible ways to deploy our applications/features.

Today, in this article, let us explore the different deployment strategies available and how to think beyond rolling deployments.

The Software Deployment Strategies

Software deployment strategies are crucial for efficiently and effectively releasing new or updated software applications. These strategies ensure smooth transitions, minimize downtime and mitigate risks. They provide organizations with various approaches to deploying software changes in production environments.

By adopting different deployment strategies, organizations can optimize their release processes. These strategies involve carefully managing the deployment of software changes to ensure reliability and user satisfaction. They allow continuous delivery while minimizing user impact and providing opportunities to monitor and roll back if issues arise.

Some deployment strategies involve incremental rollouts, where software changes are deployed gradually across the production environment, allowing organizations to monitor the impact of each release closely. Others involve deploying changes to a small subset of users or servers, allowing organizations to gather feedback and identify potential issues before a full-scale deployment.

Another strategy involves maintaining separate production environments, where one is active and serves live traffic while the other

remains inactive. This strategy allows for thoroughly testing software changes in the inactive environment before swapping the active and inactive environments, minimizing downtime and providing a quick rollback option if needed.

These deployment strategies offer flexibility and control, enabling organizations to optimize their software release processes based on their specific requirements and goals. Organizations can ensure successful software deployments by selecting the appropriate strategy while minimizing disruptions and risks.

Thinking Beyond Rolling Deployment

There is nothing unique about the Rolling deployment strategy. Even in Kubernetes, it comes along as a default strategy whether you mention it or not. But rolling deployment is not ideal when working with today's cloud-native applications and services. Firstly, modern cloud-native architectures often rely on distributed systems and microservices, where numerous interconnected components must be updated simultaneously. With a rolling deployment, updates are applied gradually, potentially causing compatibility issues or breaking the system's functionality if not carefully managed. Canary or blue-green deployments, on the other hand, provide more controlled mechanisms for updating and validating the entire ecosystem before directing traffic to the new version.

Moreover, Rolling deployments take a long time to complete. No environment isolation creates confusion; there is no clear separation or isolation between the environments where the new and old software versions are running. Also, one more thing to note is that the changes we rollout should be backward and forward compatible while using a rolling deployment strategy.

Canary Deployment Strategy

Canary deployment strategy is often used to avoid unreliable deployments. This strategy involves gradually rolling out new software versions or updates by targeting a small group of users or servers while most continue using the stable version. This approach allows for thorough testing and monitoring of the new version's performance and stability before a full rollout in a real-world environment.

By gradually exposing a small portion of users to the new version, any issues or bugs can be quickly identified and mitigated, minimizing the impact on the overall system. If problems are detected, the deployment can be halted or rolled back, ensuring that the majority of users are unaffected. This strategy acts as an early warning system, allowing quick feedback loops and reducing the risk of widespread failures.

Canary deployments allow for performance monitoring and comparison between the new and stable versions, providing insights into how the update affects system metrics like response time, error rates, and resource utilization. This information helps make informed decisions about whether to continue with full deployment or make further adjustments.

Canary Deployments using Harness

To do canary deployments on Harness, you need to have the following prerequisites.

- Harness CD free account
- A Kubernetes cluster access
- A sample application such as Guestbook. Fork it, and let's use it in our tutorial.

â

Let's sign in to our Harness CD module and create a project.

[Note: Make sure you have the Harness Delegate up and running to work with Harness CD.]

Next, create a pipeline and add the service, environment and execution steps as shown in the image below.

Make sure to select the **Canary** deployment strategy in the execution.

â

In the âServiceâ section, you need to add the manifest details.

â

Save everything and run the pipeline.

Blue-Green Deployment Strategy

A blue-green deployment strategy is often preferred to overcome unreliable deployments due to its inherent ability to mitigate risks and ensure a smooth transition between different versions of an application or service. This strategy involves maintaining two identical environments, referred to as the blue and green environments, with one serving as the production environment while the other remains inactive. By deploying new updates or changes to the inactive environment (green), the risk of disrupting the live system (blue) is minimized. This approach allows for thorough testing, troubleshooting, and validation of the green environment before routing traffic to it. If any issues or failures are encountered, the deployment can be easily rolled back by redirecting traffic to the stable blue environment.

Organizations can use the blue-green deployment strategy to ensure end-users are not negatively affected by faulty deployments. This approach ensures that only stable and thoroughly tested versions are made available to users, minimizing the risk of service disruption,

downtime, or errors. As a result, this strategy helps maintain high availability and reliability, ultimately leading to an improved user experience and higher customer satisfaction.

Blue-Green Deployments Using Harness

Follow everything from the previous Canary deployment set up until you select the deployment strategy step. In the execution strategy, you need to select âBlue Greenâ.

â

This is how your pipeline should look like when you add the Blue/Green deployment strategy.Â

â

Save and run the pipeline.

Software Deployment Best Practices

No matter which deployment path you choose, you need to have some best practices in place to make sure your engineering teams are well prepared for any unexpected events in case if they occur.Â

- **Define clear success criteria:** Before implementing a deployment, clearly define what success looks like. Determine the key metrics or indicators that will help you evaluate the performance and impact of the new feature or update.
- **Feature toggles:** Implement feature toggles or flags that allow you to enable or disable specific features or updates. This gives you the flexibility to control the release and easily roll back changes if necessary.
- **Automated testing:** Implement a robust automated testing strategy to ensure that the new changes are thoroughly tested before the deployment. This includes unit tests, integration tests, and any other relevant testing methodologies. Using Harness CI module, you can test your application using different testing frameworks and tools.Â
- **Rollback plan:** Have a well-defined rollback plan in place in case significant issues are detected during the deployment. This allows you to revert to the previous version quickly and minimize any negative impact on users or systems.
- **Documentation and templatize:** Document the entire deployment process, including configuration settings, monitoring parameters, and any troubleshooting steps. Communicate the process to your team members and stakeholders, ensuring everyone is aware of the deployment strategy and its goals. Create templates using Harness templates feature to make sure the pipeline settings are preserved and can be propagated further to the team members.Â
- **Post-deployment Evaluation:** Conduct a thorough post-deployment evaluation by collecting relevant logs and checking the health of the application/service by connecting your health source, such as Prometheus, DataDog, AppDynamics, etc. You can easily do this by using Harness Continuous Verification feature.Â

Rolling vs Canary vs Blue/Green: When to Choose What?

In general, rolling deployments are a good choice for applications that are not mission-critical and can tolerate some downtime. This strategy is suitable for small to medium-sized applications.Â Canary deployments are helpful when introducing new features or significant changes to an application. They are suitable for large and mission-critical applications that need to be monitored closely. Blue/Green deployments are ideal for applications that need to be highly reliable and have high availability requirements with zero downtime tolerance. They are suitable for critical systems and applications that demand rapid rollback capabilities.

Conclusion

As technology evolves and software development becomes more complex, embracing more advanced deployment strategies like canary and blue/green is becoming increasingly important, rather than solely relying on traditional rolling deployments. Both canary and blue/green deployments enhance overall system reliability and availability. These strategies empower teams to minimize risks, accelerate innovation, improve reliability, and scale easily. By adopting these approaches, organizations can stay ahead of the competition, deliver high-quality software efficiently, and provide an exceptional user experience in today's rapidly evolving digital landscape.

â

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/comparison-guide/turbonomic-vs-harness>

Comparison Guide

Harness

Cloud Cost Management

vs

IBM Turbonomic

Harness Cloud Cost Management vs IBM Turbonomic

Cloud Cost Management

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

IBM Turbonomic

IBM Turbonomic Application Resource Management for IBM Cloud Paks drives application performance while maximizing efficiency and maintaining compliance throughout the application lifecycle.

250,000

2008 (Turbonomic)

Acquired

Turbonomic was acquired by IBM in 2021

IBM Turbonomic is categorized as:
Application Resource Management

What is the difference between Harness Cloud Cost Management vs. IBM Turbonomic?

IBM Turbonomic is an application resource management tool that lacks the core cloud cost management features such as billing, reporting, budgeting, and forecasting that can be found in Harness Cloud Cost Management.

Turbonomic vs Harness: DevOps Tools Comparison

Updated

November 30, 2023

- SaaS
- Audience & Primary Users
- Cloud Support
- Time Granularity
- Tag Management Required
- Hybrid Cloud Visibility
- Cloud Account Visibility
- Cloud Region Visibility
- Application Visibility
- Microservice Visibility
- Environment Visibility
- Non-Cluster Visibility (Services)
- Cluster Visibility (Kubernetes/ECS)
- Root Cost Analysis (3 Metrics)
- Cloud Event Correlation (e.g. deploy)
- Cost Recommendations
- Alerting & Budgets
- Anomaly Detection
- Reporting
- Documentation
- Support
- Training
- Automated Idle Resource Management
- Cloud Cost Business Intelligence

<yes><yes>

Engineering & Finance

All

Hourly

<yes><yes> **Optional**

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Utilized, Idle, Unallocated

<yes><yes>

Engineering

All

Hourly

<no><no> **Required**

<no><no>

<yes><yes>

<yes><yes>

<no><no>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<no><no>

SaaS and Self-Hosted

Self-Hosted and SaaS

AWS, Azure, and GCP

AWS, Azure and GCP

<with><with>
Kubernetes Only

<yes><yes>

Percentage Cloud Spend

Per Workload / Node

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Coming soon

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

<with><with> Manual Schedule

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<with><with>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<yes><yes>
<yes><yes>
<no><no>
<yes><yes>
<yes><yes>
<yes><yes>
<no><no>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<with><with> Requires ServiceNow
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>

Summary:Â

If you're a large organization with a data center on-premises or are in a hybrid cloud environment, IBM Turbonomic seems like a good solution. It's not a cloud cost management tool in and of itself though, describing itself as an application resource management (ARM) tool. While it does have some interesting AI-powered features like Actions that can automatically reallocate resources, its main function is resource efficiency optimization, not cloud cost management. It falls short on cloud cost management functionality, lacking fundamental features such as budgeting/forecasting, cost anomaly detection, and even basic cloud cost reporting and billing.

If you're looking for rich multi-cloud and container cost visibility, automated cloud cost savings, and cost forecasting, Harness Cloud Cost Management is the clear winner. Harness understands the pressure on engineering and FinOps teams to optimize cloud efficiency, along with the requirement to paint a full picture of cloud costs that includes container costs. Harness is a powerful, easy to implement solution that gets customers more value out of the cloud.

Tag Management Required:

Tag management is essential to understanding your infrastructure costs, and having a strong tagging policy is the foundation for cloud asset management. With Harness CCM, there's no need for manual tagging of cloud resources to maintain excellent tag cleanliness. The Harness Platform automates the creation of cloud infrastructure throughout the CD development pipeline and can automatically provision resources and tag them at creation time.

Unfortunately, manual tagging and great tag health is a requirement to get the full value out of Turbonomic's resource management capabilities. The tool relies heavily on customers creating, applying, and maintaining good tagging policies to get granular control over resources and workflows.

Cost Anomaly Detection:

Harness allows customers to automatically flag and drill down to any anomalous cost spikes, avoiding end-of-month surprises. Leveraging machine learning that customizes itself to a customer's environment, including customizations for seasonality, customers can proactively detect and resolve cost spikes. **Turbonomic does not provide any insights into unexpected cost spikes.**

Resource Management:

One of the core benefits of using cloud cost management tools is the recommendations for optimizing the efficiency of cloud infrastructure resources. The challenge that all companies face is making sure that the right engineers are reviewing those recommendations and taking action. Harness goes beyond cost recommendations to actively manage cloud resource idle time effectively with Cloud Autostopping, without custom scripts or manual engineering effort. So engineers are free to focus on what matters most: revenue generating application development.

While Turbonomic does have some nice automation to push recommendation reviews into Jira or ServiceNow and take action on approved recommendations, it doesn't solve the problem that engineers must review those tickets, taking engineering time away from development. There is no support for managing idle resources, whether via schedule or otherwise.

Cloud Cost Management

Interested in seeing what's under the hood? Browse through the Harness Cloud Cost Management Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/blog/hierarchical-breakdown-of-organizational-cloud-costs>

Hierarchical Breakdown of Organizational Cloud Costs

Businesses leveraging public clouds often grapple with understanding and managing their cloud costs. Traditional methods and tools might offer insights, but in today's complex cloud environments, deeper visibility is essential.

Without a clear hierarchical breakdown, costs can easily spiral out of control without any accountability or ownership. Costs can become murky, teams can exceed budgets unknowingly, and companies can waste budgets on underused or redundant cloud resources. Therefore, getting a handle on these costs by understanding them at every tier of the organization is vital.

Harness Cloud Cost Management for Cost Visibility and Attribution

Harness Cloud Cost Management (CCM) offers two powerful cost attribution features - Cost Categories and Perspectives. These features offer a comprehensive view into cloud costs that is as rarely implemented as it is essential. Cost categories offer a way to group cloud costs in ways that are most meaningful for their business and act as the basis for creating and managing perspectives, which are cost views customized to the needs of your business that empower teams to track and manage costs directly.

Cost Categories

Users can group cloud costs in ways that are most meaningful for their business. This not only facilitates clearer budgeting but also provides an in-depth breakdown. Such insights offer clarity on which departments, projects, or applications are consuming the most resources and money.

It also ensures that costs across cloud providers and Kubernetes deployments can be included in the same cost categories to provide a

complete view of costs, without requiring manual report creation.

Perspectives

Harness CCM offers a flexible approach to viewing and allocating costs. It offers various perspectives that can be tailored to roles, departments, projects, or any other organizational structure. This ensures that everyone gets the most relevant and contextual view of their cost data.Â

Each perspective gives that team direct access to cost forecasting, relevant recommendations to optimize costs they are responsible for, the ability to set cloud budgets, and get direct alerts on anomalous cloud spends. All of this is in the context of the cloud costs that their Perspective is tracking to ensure they get the information they need to track and take action on their cloud costs.

Benefits of Hierarchical Cost Breakdown

- **Transparency** - Organizations gain a transparent view of how cloud spend maps to resources consumed, and can make data-driven decisions.
- **Accountability** - Departments or teams can be held accountable for their specific cloud expenses with more granular and accurate cost allocation.
- **Optimization** - Identifying which resources are not allocated to any team or department becomes straightforward, leading to cost savings. As they say, you can't improve what you don't measure.
- **Forecasting** - With a clear view of current costs per team or department, predicting future expenses across the organization becomes more accurate.

Real-World Application Example

Let us understand how you can leverage cost categories with a case study. Letâs assume we have an Organisation X, with 3 levels of hierarchy at which cloud costs are tracked; VP, Director and Manager. Multiple Managers roll up to a director and multiple directors roll up to a VP. Each manager is responsible for a specific team.

Cost categories are use Cost Buckets as the building block for creating the category. Each Cost Bucket is defined by a set of rules, which can be as simple as pulling from a team tag from a single cloud to as complex as multiple rules pulling data from across all clouds spanning across multiple different rules.Â This ensures that all costs per team, regardless of source, are included in their cost category.

Weâll start at the bottom of the hierarchy, at the Teams category level, and include all costs for each teams cost bucket.

Once the Teams have been defined, we can then created a hierarchical (nested) cost category for Director, which will include the Teams that report into each Director. Following that, we can repeat the process to create the VP Cost Category, defining the Directors that report into each VP.

Once the cost categories are created they can be used to define new perspectivesÂ and further breakdown costs in existing perspectives. This can give you high level visibility at multiple levels of the business, at the VP, Director and manager levels, all the way down to the individual teams.Â

These cost category and cost bucket definitions can also include attribution of costs for shared cloud services. Sharing of costs across cost buckets can be done using various strategies such as proportional cost sharing, sharing costs based on a fixed percentage or sharing costs equally across the different cost buckets.

Without a tool like Harness CCM, understanding which teamâs activity incurs what costs, including shared costs, can be challenging. However, with Cost Categories and Perspectives, each VP, Director, team, and the finance team, can get a clear and accurate breakdown. They can see costs associated with each department, assess the ROI, and allocate budgets more effectively for the future.

Conclusion

In today's multifaceted cloud environment, having a tool that offers a granular, hierarchical view of costs across your company isn't just a nice-to-haveâit's essential. Harness CCMâs features of Cost Categories and Perspectives provide this much-needed visibility, ensuring that businesses can use cloud resources effectively without busting their budgets.

Looking to get this kind of granular visibility into your company's cloud costs? Take control of your cloud expenditure and gain unparalleled insights with Harness CCM. Book a demo or sign up for free today.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/comparison-guide/apptio-cloudability-vs-harness>

Comparison Guide

Harness

Cloud Cost Management

vs

Apptio Cloudability

Harness Cloud Cost Management vs Apptio Cloudability

Cloud Cost Management

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

Apptio Cloudability

Apptio Cloudability is a cloud cost management and optimization tool that enables IT, finance, and business teams to optimize their costs and communicate the business value of the cloud.

1200

2011

Acquired

Cloudability was acquired by Apptio in 2019

Apptio Cloudability is categorized as:
Cloud Cost Management

What is the difference between Harness Cloud Cost Management vs. Apptio Cloudability?

Apptio Cloudability provides good cost visibility and reporting features, but lacks all-in-one multi-cloud cost visibility, or any of the advanced automated cloud cost savings features found in Harness Cloud Cost Management.

[Apptio \(Cloudability\) vs Harness: DevOps Tools Comparison | Harness](#)

Updated

November 30, 2023

- SaaS
- Audience & Primary Users
- Time Granularity
- Tag Management Required
- Hybrid Cloud Visibility
- Cloud Account Visibility
- Cloud Region Visibility
- Application Visibility
- Microservice Visibility
- Environment Visibility
- Non-Cluster Visibility (Services)
- Cluster Visibility (Kubernetes/ECS)
- Root Cost Analysis (3 Metrics)
- Cloud Event Correlation (e.g. deploy)
- Cost Recommendations
- Alerting & Budgets
- Anomaly Detection
- Reporting
- Documentation
- Support
- Training
- Automated Idle Resource Management
- Cloud Cost Business Intelligence

<yes><yes>

Engineering & Finance

Hourly

<yes><yes> **Optional**

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Utilized, Idle, Unallocated

<yes><yes>

Finance & FinOps

Daily

<no><no> **Required**

<no><no>

<yes><yes>

<yes><yes>

<with><with> **Requires Tagging**

<with><with> **Requires Tagging**

<with><with> **Requires Tagging**

<yes><yes>

<with><with> **Only AKS**

<no><no> **No Idle/Unallocated Cost Visibility**

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<no><no>

SaaS and Self-Hosted

SaaS

AWS, Azure, and GCP

AWS, Azure, GCP and OCI

<with><with>
Kubernetes Only

<no><no>

Percentage Cloud Spend

Percentage Cloud Spend

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Cluster Agent Required

<yes><yes>

<with><with> No single dashboard view

Coming soon

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<with><with>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<with><with>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes> via acquisition
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<yes><yes>

Detailed Feature Comparison

Harness Cloud Cost Management vs Apptio Cloudability

Summary:

Apptio Cloudability provides a great set of visibility and forecast tools focused for individual cloud providers, but lacks the all-in-one multi-cloud visibility modern organizations need and have come to expect. They also fall short on automating cloud cost savings features that make saving costs simply and easy. There's a great deal of manual heavy lifting required to build and maintain good tag hygiene, which is required to fuel their reporting features. They also lack native support for Kubernetes cost allocation, requiring an open source agent to be installed on every Kubernetes cluster in order to get the granular cluster visibility that Harness includes automatically.

If you're looking to maximize your cloud cost savings through intelligent automation, and get rich multi-cloud and container cost visibility, Harness Cloud Cost Management is the clear winner. Harness understands the pressure on engineering and FinOps teams to optimize cloud efficiency, along with the requirement to paint a full picture of multi-cloud costs that includes container costs. Harness is a powerful, easy to implement solution that gets customers more value out of the cloud.

Automated Cost Savings:

Getting engineers to take action on cloud cost recommendations shouldn't be your only path to cloud cost savings. Automation can unlock cost savings opportunities that simply can't be realized with manual or scripted efforts. Harness CCM has a multitude of automation tools that save our customers money, from idle cloud resource management (Cloud Autostopping) to spot instance orchestration, and more, that saves our customers up to 90% on cloud costs.

Apptio Cloudability acquired Cloudwiry to automate the RI/SP purchases, but it will be some time before this feature is integrated into the Cloudability portfolio (if ever). This acquisition aside, Apptio Cloudability has no native automated cost savings features available for their customers.

Tag Management Required:

Tag management is essential to understanding your infrastructure costs, and having a strong tagging policy is the foundation for cloud asset management. Unfortunately, manual tagging and great tag health is a requirement to get the full value of Apptio Cloudability, and get granular visibility into cloud assets and costs. Apptio Cloudability also won't update any tag changes to the cloud providers, the changes exist solely within Apptio Cloudability.Â

With Harness CCM, there's no need for manual tagging of cloud resources to maintain excellent tag cleanliness. The Harness Platform automates the creation of cloud infrastructure throughout the CD development pipeline and can automatically provision resources and tag them at creation time.

Time Granularity:

Businesses need to stay on top of shifting cloud costs so that they can catch (and stop) run-away cloud spend as it occurs. A rogue cluster can run up huge cloud bills if not detected quickly. That's why Harness CCM offers hourly granularity in its cost reporting to give an almost-real-time view into possible issues before they turn costly.

Apptio Cloudability only offers daily granularity into its cost reporting and analysis. For many customers, especially engineering teams, that simply isn't enough.

Cloud Cost Management

Interested in seeing what's under the hood? Browse through the Harness Cloud Cost Management Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/comparison-guide/aws-cost-management-vs-harness>

Comparison Guide

Harness

Cloud Cost Management

VS

AWS Cost Management

Harness Cloud Cost Management vs AWS Cost Management

Cloud Cost Management

500-1000

2016

\$425M

Harness is categorized as:
Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

AWS Cost Management

AWS continuously innovates and delivers the latest technologies across every solution area so you can meet high performance needs and scale at a lower cost. AWS offers a suite of management tools to monitor your application cost and identify modernizing and rightsizing opportunities.

45k+

2003

Public Company

Amazon Web Services (AWS) is the cloud technology division of Amazon, Inc.
AWS Cost Management is categorized as: Cloud Cost Management (CCM)

How Does AWS Cost Management Compare?

AWS Cost Management is a suite of products that work together to assist with cloud cost management and is âgood enoughâ for many AWS users. However, they canât support large complex deployments or help automate your cloud cost savings like Harness can.

AWS Cost Management vs Harness: DevOps Tools Comparison | Harness

Updated

November 30, 2023

- **SaaS**
- **Audience & Primary Users**
- **Cloud Support**
- **Time Granularity**
- **Tag Management Required**
- **Hybrid Cloud Visibility**
- **Cloud Account Visibility**
- **Cloud Region Visibility**
- **Application Visibility**
- **Microservice Visibility**
- **Environment Visibility**
- **Non-Cluster Visibility (Services)**
- **Cluster Visibility (Kubernetes/ECS)**
- **Root Cost Analysis (3 Metrics)**
- **Utilized, Idle, Unallocated**
- **Cloud Event Correlation (e.g. deploy)**
- **Cost Recommendations**
- **Cloud Event Correlation (e.g. deploy)**
- **Anomaly Detection**
- **Reporting**
- **Documentation**
- **Automated Idle Resource Management**
- **Cloud Cost Business Intelligence**

<yes><yes>

Engineering & Finance

All

Hourly

<yes><yes> **Optional**

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Utilized, Idle, Unallocated

<yes><yes>

Finance

<with><with> **AWS Only**

Hourly

<yes><yes> **Optional**

<with><with> **AWS Only**

<yes><yes>

<yes><yes>

<with><with> **Only With Tags, Only For AWS**

<with><with> **Only With Tags, Only For AWS**

<with><with> **Only With Tags, Only For AWS**

<yes><yes>

<with><with> **Only ECS, EKS**

<no><no>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<no><no>

SaaS and Self-Hosted

SaaS

AWS, Azure, and GCP

AWS Only

<with><with>

Kubernetes Only

<no><no>

Percentage Cloud Spend

Free

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<no><no>

Coming soon

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>
<yes><yes>
<yes><yes>
<no><no>
<no><no>
<no><no>
<yes><yes>
<yes><yes>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<with><with>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>

Detailed Feature Comparison

Summary

AWS Cost Management is a free suite of products that includes AWS Cost Explorer, AWS Cost Categories, AWS Budgets, AWS Trusted Advisor, and AWS Cost Anomaly Detection. Together these products help AWS users understand their cloud spend and provide basic support for rightsizing AWS EC2 instances. For many organizations with small cloud deployments, these tools will work well enough for their needs. While free, there are usage limits to many of the AWS Cost Management features, which even small to medium-sized cloud environments can easily exceed.Â

Larger organizations with multi-cloud environments across Azure and GCP, as well as cloud-native Kubernetes architectures, will be left needing to find other tools to understand and optimize those cloud costs. Even within a pure AWS environment, youâll need to leverage multiple AWS tools in order to get the functionality fully built into Harness CCM, and youâll still lack the automation that Harness CCM delivers to save you up to 70% on your overall cloud bills.

Multi-cloud Cost Visibility

With over 60% of organizations having some or all of their workloads spread across multiple cloud providers, itâs critical that cloud cost management tools have the ability to track and report on all multi-cloud costs for an organization. **As expected, AWS Cost Management tools only support AWS infrastructure.**

Harness provides granular cost visibility for AWS, Azure, and GCP, as well as Kubernetes deployments on these cloud providers as well as on-premises, all delivered in easy-to-consume reports and dashboards customized to your usersâ business needs.Â

Kubernetes Support

Cloud-native deployments depend on Kubernetes to deliver scalable and powerful services that can be deployed on shared infrastructure. The challenge, though, is getting the full picture of what portion of shared clusters are consumed by different teams or applications. AWS Cost Explorer can provide a single monolithic charge for your EKS/ECS cluster spend, but what it canât do is break down those shared costs into something usable, and it wonât give you any recommendations for optimizing those costs.

Harness not only gives you deep insights into shared cluster costs, but we also provide detailed rightsizing recommendations at the workload and node pool levels. We can also help automate your cluster autoscaling to take advantage of spot instances via our Cluster Orchestrator feature, saving up to 90% off on cluster costs.

Savings Plans and Reservations

AWS provides significant savings for their customers who pre-purchase their compute instances (reserved instances) or commit to an overall level of spend (savings plans). They even do a good job of providing RI purchase recommendations based on your cloud spend. However, what they donât do is give you any recommendations on savings plan purchases, nor help automate the management of these contracts for you.

Harness not only gives you detailed recommendations for both reserved instances as well as savings plans, we also automate the contract execution over time to relieve your teams of that burden. Simply set your coverage goals for both convertible RIs and savings plans, the payment terms that you want, and Harness takes over, making monthly purchases to keep your contracts in line with current usage and coverage goals.

Cost Anomaly Detection

One minor configuration error can result in massive bill shock if itâs not caught and addressed quickly. AWS Cost Anomaly Detection (yet another tool) does provide the ability to set up cost anomaly alerts across several different dimensions. However, the alerts are most likely not as granular as you need them to be, and they have hard limits on how many alerts can be configured per account. Thereâs also a challenge in that they require admins to do their homework to determine what their anomaly thresholds should be, which can lead to either missed cost anomalies or so many anomaly reports that true alerts are lost in the noise.

Harness cost anomaly detection uses multiple statistical analysis methods to get to the most accurate cost anomaly alerts, cutting out the noise that other tools create. Our alerts lead your engineers exactly to the source, and we can provide root cause analysis for actions taken upstream in your development pipeline that may have led to the cost spikes.

Cloud Cost Management

Interested in seeing what's under the hood? Browse through the Harness Cloud Cost Management Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Harnessing The Power of Secrets Management

While the whole DevOps practices talk about automation and speed, it is also important to ensure that security is taken care of. Most organizations that ignored security are now rethinking it and embracing the DevSecOps principles of shifting left and adding security checkpoints at every step of the CI/CD pipeline. Security is getting more attention as it gets the highest points when considering any tool. Hence, security professionals are in high demand to ensure the systems and processes work as expected without getting attacked by hackers and security vulnerabilities. The software delivery pipeline consists of a CI/CD pipeline that requires a lot of connections from third-party tools for a smooth software delivery workflow. These external tools and platforms, such as your Docker Hub, Artifactory, Cloud providers, Security testing tools, etc, are connected to your CI/CD pipeline through secrets.

Storing and managing these secrets is crucial so the credentials are not compromised. It is also one of the essential and basic traits when considering a CI/CD platform to understand how the secrets are managed securely.Â

Today, we will see how secrets are managed in the Harness platform.

What is Secrets Management?

Secrets management in the context of DevOps refers to securely storing and managing sensitive information such as passwords, API keys, database credentials, and other secrets used in the software development and deployment process. These secrets are typically required by various components of an application, including infrastructure configuration, code repositories, build scripts, deployment pipelines, and runtime environments.

The goal of secrets management is to protect sensitive information from unauthorized access, misuse, or exposure, while still providing controlled access to authorized users, processes, or systems. In a DevOps and CI/CD environment, secrets management becomes crucial because automation tools and processes often require access to these secrets to perform tasks such as building, testing, and deploying applications.

Where are secrets involved in a CI/CD pipeline in DevOps?

Secrets are integral to a CI/CD pipeline and can be present at every step, from code to deployment environment. In a typical CI/CD workflow, a developer pushes his/her code to a source control management (SCM) tool like GitHub or BitBucket, and then a continuous integration (CI) tool gets triggered, where it builds and tests the code. Next, the artifacts get created and pushed to a repository, where an artifact repository connector is required. Then the code gets deployed to production, and you need to connect the cloud provider or wherever you are deploying the code (the target environment). Finally, the monitoring tool gets connected to the CI/CD pipeline for observability purposes to keep the application running as expected.Â

In a typical CI/CD workflow/pipeline, these types of connectors are involved where managing secrets is very important.Â

- Code repository connections - GitHub, GitLab, and BitBucket.
- Continuous Integration servers and Continuous Delivery platforms â Jenkins, Bamboo, JFrog, TeamCity, Drone, Harness, GitLab CI/CD, CircleCI, etc.
- Database connections - MongoDB, Redis, ElasticSearch etc
- Cloud Provider connections - AWS, GCP, Azure, Kubernetes (as keys, configuration files, and other information).
- Container Registry Connections - DockerHub, JFrog Artifactory, Sonatype Nexus, etc
- Infrastructure â (SSH keys, load balancer credentials).
- Collaboration providers â Slack, SMTP, Jira, ServiceNow.
- Verification providers â AppDynamics, New Relic, Prometheus, Splunk, etc

Challenges of Managing Secrets in a DevOps Pipeline

Managing secrets in a DevOps pipeline poses significant challenges in today's technology landscape. One of the primary concerns revolves around maintaining the security and integrity of sensitive information, such as API keys, passwords, and encryption keys, throughout the development and deployment processes. As DevOps emphasizes rapid and continuous delivery, secrets management becomes crucial to prevent unauthorized access and data breaches. Additionally, the distributed nature of DevOps, with multiple teams and various tools involved, adds complexity to secret distribution, rotation, and revocation. Ensuring proper access controls and permissions across different environments, including development, staging, and production, further compounds the challenges.Â

â

â

â

â

â

â

â

Addressing these challenges requires a combination of robust security practices, effective access controls, automation, and secure secret management solutions.Â

How are Secrets Managed in Harness?

You can manage your secrets in Harness using a key management service (KMS) or a third-party secret manager. Google Cloud Key Management Service is the default Secret Manager in Harness and is named Harness Secret Manager Google KMS.

The Key Management Service (Google Cloud KMS or AWS KMS) only stores the key. Harness uses envelope encryption to encrypt and decrypt secrets. The encrypted secret and the encrypted Data Encryption Key (used for envelope encryption) are stored in the Harness database.Â

Using Third-Party Secret Managersâ

You can also use third-party Secret Managers, for example, HashiCorp Vault, Azure Key Vault, and AWS Secrets Manager.

These Secret Managers store the key, perform encryption and decryption, and also store the secrets (encrypted key pair). Neither the keys nor the secrets are stored in the Harness database. A reference to the secret is stored in the Harness database.

Letâs see how to add a Google KMS.Â

Go to your project and click on âConnectorsâÂ

â

â

Click on the âNew Connectorâ, and you will be presented with all the possible secret managers present.Â

â

Click on the âGCP KMSâ and continue. Mention a name and start adding the details.Â

Go to your GCP console and click on âSecurityâ and then âKey Managementâ.

Create a key ring and open the Actions menu (â®), then click Copy Resource Name.

[Each KMS [key management system] is a little different. Google KMS follows this pattern for folks to retrieve the key]

A reference to the key is now on your clipboard.

Paste the reference into an editor.

You can now copy and paste its substrings into each of the Harness Secret Managerâs Details settings as shown below.

â

Next, you need to attach the GCP KMS Credentials File.Â

You can follow this guide and connect your GCP KMS to Harness.Â

Similarly, you can addÂ

- AWS Secrets Manager
- AWS KMS Secrets Manager
- Azure Key Vault Secrets
- HashiCorp Vault Secrets Manager

You can also add encrypted test secrets, file secrets or SSH keys to reference in your CI/CD pipeline. Harness includes a built-in Secrets Management feature that enables you to store encrypted secrets, such as access keys, and use them in your Harness connectors and pipelines.

â

This is how we add our GitHub Personal Access Token, Docker Hub credentials, Database connection string, etc.

â

One thing to note here is that you need to make sure that your Delegate is up and running.

Harness Delegate is a service you run in your local network or VPC to connect your artifacts, infrastructure, collaboration, verification, and other providers, with Harness Manager. The first time you connect Harness to a third-party resource, Harness Delegate is installed in

your target infrastructure, for example, a Kubernetes cluster. Know more about Harness Delegate here.[Â](#)

Docker Registry Secret Connector Example

When you want to connect your Docker Hub to Harness, this is the step you need to take.

Click on the new connector and then select **âDocker Registryâ** from the list.

Mention a name and continue.

Next, you will add the Docker Registry URL and authenticate with your Docker Hub credentials as shown in the screenshot below.[Â](#)

If you see, the password is using an encrypted text secret that nobody can see.[Â](#)

Finally, you will make sure to connect this with Harness Delegate.

You need to make sure that the connection between your Docker Hub and Delegate is successful.[Â](#)

This is how secrets are used and managed at Harness. You can also add and store SSH keys securely in Harness.[Â](#)

Take a deep dive at Harness secret management overview.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/comparison-guide/kubecost-vs-harness>

Comparison Guide

Harness

Cloud Cost Management

VS

Kubecost

Harness Cloud Cost Management vs Kubecost

Cloud Cost Management

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

Kubecost

Kubecost provides real-time cost visibility and insights for teams using Kubernetes, helping you continuously reduce your cloud costs.

11-50

2018

\$30.5M

KubeCost raised \$30.5M in a Series A round in 2022

Kubecost is categorized as:

Kubernetes Cost Management

What is the difference between Harness Cloud Cost Management vs. Kubecost?

Kubecost provides excellent cost visibility and optimization for Kubernetes environments, but falls short in providing support for non-cluster services. Whether that's visibility, optimization, or forecasting, Harness beats Kubernetes.

Kubecost vs Harness: DevOps Tools Comparison

Updated

November 30, 2023

- SaaS
- Audience & Primary Users
- Cloud Support
- Time Granularity
- Tag Management Required
- Hybrid Cloud Visibility
- Cloud Account Visibility
- Cloud Region Visibility
- Application Visibility
- Microservice Visibility
- Environment Visibility
- Non-Cluster Visibility (Services)
- Cluster Visibility (Kubernetes/ECS)
- Root Cost Analysis (3 Metrics)
- Cloud Event Correlation (e.g. deploy)
- Cost Recommendations
- Alerting & Budgets
- Anomaly Detection
- Reporting
- Documentation
- Support

- **Training**
- **Automated Idle Resource Management**
- **Cloud Cost Business Intelligence**

<yes><yes>

Engineering & Finance

All

Hourly

<yes><yes> **Optional**

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Utilized, Idle, Unallocated

<yes><yes>

Engineering

All, With Caveats

Hourly

<yes><yes> **Optional**

<yes><yes> **With Caveats**

<yes><yes>

<yes><yes>

Utilized, Idle, Unallocated

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

SaaS and Self-Hosted

Self Managed (installed in customer cloud environment) and SaaS

AWS, Azure, and GCP

AWS, Azure and GCP

<with><with>

Kubernetes Only

<with><with> Kubernetes Only

Percentage Cloud Spend

Per-node pricing

<yes><yes>

<with><with> K8s Only

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<with><with> K8s Only

Coming soon

<no><no>

<yes><yes>

<with><with> K8s Only

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<with><with> K8s Only

<yes><yes>

<yes><yes>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<with><with> CLI Based
<yes><yes>
<yes><yes>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>

<yes><yes>

<yes><yes>

<with><with>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

Summary:

At first glance, Kubecost seems to do it all â and while it does excel in multiple ways, itâs important to remember that Kubecost is primarily for Kubernetes costs (as implied by the name). As always, we love giving credit where itâs due: Kubecost monitors costs in Kubernetes fantastically well â but it stops there. For any costs outside of Kubernetes, you may find yourself needing another serviceâ

...or, alternatively, you could go with a product that does both well, like Harness Cloud Cost Management. You probably use too many tools to manage cloud costs already. Why not bring that all into a single pane of glass on the Harness platform?

Full Multi-cloud Cost Visibility:

You wouldnât drive your car with only half your windshield clean, so why run your cloud with only partial cost visibility?Â With Kubecost, you only get cost information for your Kubernetes clusters, which accounts for less than 50% of the cloud infrastructure for most organizations.

Harness provides a premier Kubernetes cost management solution (along with every other kind of cloud cost) on all major cloud providers, including single, multi-, or hybrid cloud environments. We give you deep business context into your costs with Cost Perspectives, providing you with a single view of all containerized and non-containerized workloads by teams, applications, environments, or whichever view makes the most sense for your business.Â

Automated Costs Savings:

One of the core benefits of using cloud cost management tools is the recommendations for optimizing the efficiency of cloud infrastructure resources. The challenge that all companies face is making sure that the right engineers are reviewing those recommendations and taking action. Itâs much better to create new cost savings opportunities through automation.

Harness goes beyond cost recommendations to actively manage cloud resource idle time effectively with Cloud AutoStopping, without custom scripts or manual engineering effort. Engineers are then free to focus on what matters most: revenue generating application development. We also automate Kubernetes cluster management to orchestrate Kubernetes cluster autoscaling with Spot Instance Orchestration. You can automatically respond to spot instance interruptions, closing the gap in spot instance availability.

Kubecost doesnât have any automated cost savings tools available for its customers, and has only recently released their automatic right-sizing feature in alpha or beta.

Cloud Cost Management

Interested in seeing what's under the hood? Browse through the Harness Cloud Cost Management Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/comparison-guide/gcp-billing-vs-harness>

Comparison Guide

Harness

Cloud Cost Management

VS

Google Cost Management

Harness Cloud Cost Management vs Google Cost Management

Cloud Cost Management

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

Google Cost Management

Cloud Billing is a collection of tools that help you track and understand your Google Cloud spending, pay your bill, and optimize your costs.

37k+ (GCP Team)

2008 (GCP Launch)

Public Company

Google Cloud Billing is categorized as:

Cloud Cost Management (CCM)

â

Google Cloud Platform (GCP) is the cloud technology division of Google

How Does Google Cost Management Compare?

Google Cost Management is a suite of products that work together to give cost visibility and savings recommendations, and is âgood enoughâ for the basics, but lacks any support for cost savings automation or multi-cloud cost management that Harness provides.

Google Cost Management vs Harness: DevOps Tools Comparison

Updated

November 30, 2023

- SaaS
- Audience & Primary Users
- Cloud Support
- Time Granularity
- Tag Management Required
- Hybrid Cloud Visibility
- Cloud Account Visibility
- Cloud Region Visibility
- Application Visibility
- Microservice Visibility
- Environment Visibility
- Non-Cluster Visibility (Services)
- Cluster Visibility (Kubernetes/ECS)

- Root Cost Analysis (3 Metrics)
- Cloud Event Correlation (e.g. deploy)
- Cost Recommendations
- Alerting & Budgets
- Anomaly Detection
- Reporting
- Documentation
- Automated Idle Resource Management
- Cloud Cost Business Intelligence

<yes><yes>

Engineering & Finance

All

Hourly

<yes><yes> **Optional**

<yes><yes>

Utilized, Idle, Unallocated

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Finance

<with><with> **GCP Only**

Hourly

<yes><yes> **Optional**

<with><with> **GCP Only**

<yes><yes>

<yes><yes>

<with><with> **Only With Tagging, Only for GCP**

<with><with> **Only With Tagging, Only for GCP**

<with><with> **Only With Tagging, Only for GCP**

<yes><yes>

<with><with> **Only GKE**

<no><no>

<no><no>

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<no><no>

<no><no>

SaaS and Self-Hosted

SaaS

AWS, Azure, and GCP

GCP Only

<with><with>

Kubernetes Only

<no><no>

Percentage Cloud Spend

Free

<yes><yes>

<with><with>

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<no><no>

Coming soon

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<with><with>

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<with><with> Budgets Only

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>

Summary

Google Cost Management is a suite of products including Google Cloud Billing, Recommenders, and the newly released FinOps Hub (Sept 2023) that together provide a good set of visibility, savings, and forecast tools for GCP. As a free tool available to GCP customers, it provides the basics that finance teams and budget owners want to know. However, they fall short in providing support for multi-cloud setups and container orchestration outside of GKE, which is where the world is headed. To fully leverage the capabilities of Google Cloud Billing and get a deep understanding of costs, thereâs an inordinate amount of tagging required, and customers find themselves lacking support for day-to-day operations, as with services such as anomaly detection. Google Cloud Billing is built for point-in-time views into costs, not for the daily toil that goes into efficient cloud cost management.

If youâre looking for rich multi-cloud and container cost visibility, automated cost savings, and cloud governance, Harness Cloud Cost Management is the clear winner. Harness delivers automated cost savings features such as Cloud AutoStopping and Commitment Orchestration, creating cost savings opportunities that donât exist with manual efforts. Our Cloud Asset Governance gives our customers the ability to proactively control and govern cloud costs over time, allowing our customers to get more from their cloud.

Multi-cloud Cost Visibility

With over 60% of organizations having some or all of their workloads spread across multiple cloud providers, itâs critical that cloud cost management tools have the ability to track and report on all multi-cloud costs for an organization. **As expected, Google Cost Management tools only support GCP infrastructure.**

Harness provides granular cost visibility for AWS, Azure, and GCP, as well as Kubernetes deployments on these cloud providers as well as on-premises, all delivered in easy-to-consume reports and dashboards customized to your usersâ business needs.Â

Automated Cost Savings:Â

Getting engineers to take action on cloud cost recommendations shouldnât be your only path to cloud cost savings. Automation can unlock cost savings opportunities that simply canât be realized with manual or scripted efforts. Harness CCM has a multitude of automation tools that save our customers money, from idle cloud resource management (Cloud Autostopping) to spot instance orchestration, and Commitment Orchestration, which save our customers up to 90% on cloud costs when used together

The only savings automation that Google (in general, not in Google Cloud Billing) provides is an option to auto-renew Committed Use Discounts (CUDs). There is no intelligence built-in to understand what *should* be renewed, and once a CUD is renewed, it can not be canceled. Customers could be stuck with contracts that no longer meet their needs, leading to cloud cost waste.Â

Cost Anomaly Monitoring

Harness allows customers to automatically flag and drill down to any anomalous cost spikes, avoiding end-of-month surprises. Leveraging machine learning that customizes itself to a customerâs environment, including customizations for seasonality, customers can proactively detect and resolve cost spikes. **Google Cloud Billing does not provide any insights into unexpected cost spikes.**

Cloud Cost Management

Interested in seeing what's under the hood? Browse through the Harness Cloud Cost Management Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Comparison Guide

Harness

Cloud Cost Management

VS

Spot by NetApp

Harness Cloud Cost Management vs Spot by NetApp

Cloud Cost Management

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

Spot by NetApp

Spot by NetApp automates and continuously optimizes cloud compute infrastructure, helping companies reduce their compute costs by up to 90% while ensuring scalability and availability.

11000

2015

Acquired by NetAppâ

Spot.io was acquired by NetApp in 2020

Spot by NetApp is categorized as:

Cloud Cost Management

What is the difference between Harness Cloud Cost Management vs. Spot by NetApp?

Spot by NetApp focuses on cost savings through spot instance orchestration and commitment management. However, Spot lacks the deep visibility into cloud costs in your business context, unlike Harness that provides automated cost savings AND excellent cost visibility to our customers.

Spot by NetApp vs Harness: DevOps Tools Comparison

Updated

November 30, 2023

- SaaS
- Audience & Primary Users
- Cloud Support
- Time Granularity
- Tag Management Required

- Hybrid Cloud Visibility
- Cloud Account Visibility
- Cloud Region Visibility
- Application Visibility
- Microservice Visibility
- Environment Visibility
- Non-Cluster Visibility (Services)
- Cluster Visibility (Kubernetes/ECS)
- Root Cost Analysis (3 Metrics)
- Cloud Event Correlation (e.g. deploy)
- Cost Recommendations
- Alerting & Budgets
- Anomaly Detection
- Reporting
- Documentation
- Support
- Training
- Automated Idle Resource Management
- Cloud Cost Business Intelligence

<yes><yes>

Engineering & Finance

All

Hourly

<yes><yes> **Optional**

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Utilized, Idle, Unallocated

<yes><yes>

Engineering & Finance

All

Hourly

<no><no> **Required**

<yes><yes>

<yes><yes>

<yes><yes>

<with><with> **Application Visibility**

<with><with> **Requires Tagging**

<with><with> **Requires Tagging**

<yes><yes>

<with><with> **Requires Tagging**

<no><no>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<no><no>

SaaS and Self-Hosted

SaaS

AWS, Azure, and GCP

AWS, Azure, GCP

<with><with>

Kubernetes Only

<no><no>

Percentage Cloud Spend

Percentage of Savings

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<with><with>

Coming soon

<no><no>

<yes><yes>
<yes><yes>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<with><with>
<yes><yes>
<with><with> Manual Schedule
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<no><no>
<yes><yes>
<yes><yes>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<no><no>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<with><with>

<yes><yes>

<with><with>

<yes><yes>

<with><with>

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

Detailed Feature Comparison

Harness Cloud Cost Management vs. Spot by NetApp

Summary:

Spot by NetApp approached the cloud cost management space from the angle of active cloud management, meaning they honed in on the savings component of cloud cost management while providing less support for visibility and forecasting use cases. While Spot by NetApp has evolved and expanded to include support for more âclassicâ cloud cost management use cases, they do so in the context of cost savings, which may not make sense as an immediate use case for all teams in the cloud. Many teams in the cloud want to see costs in the context of their applications, microservices, environments, or by higher-level groupings such as team or business unit, and Spot by NetApp does not provide this support. Instead of by logical groupings in the context of the business, it provides this visibility at a cluster or workload level.

For teams that want both automated cloud cost savings as well as rich multi-cloud and container cost visibility, savings, and forecasting, Harness is the clear winner. Harness understands the pressure on engineering and financial management teams, and the requirement to paint a full picture of cloud costs that includes container costs. Our Cloud AutoStoppingâ¢, combined with Spot Orchestration, automates and creates cost savings opportunities that canât be found through manual scheduling efforts.

Cost Visibility:

Understanding your cloud costs in the perspective that you do business is fundamental to taking control of and optimizing your cloud spend. With multi-cloud deployments built on scalable Kubernetes clusters on the rise, having a single view of all of your costs by department, workload, environment, or any aspect that fits your business needs ensures that your teams have the information they need to manage their cloud spend.Â

Spot by NetApp falls short on providing costs in their users business contexts, instead providing only account/sub-account level granularity of cloud costs. They donât provide unified views of multi-cloud costs, so your teams, applications, and workloads donât have full visibility of their costs out of the box.Â

Harness Cloud Cost Management excels at giving your teams the granular cost visibility they need to succeed. From Cost Perspectives that are customizable views of costs in your business context and Cost Categories which allow you to do dynamic bucketing of cloud costs across all your multi-cloud environments, to custom business intelligence dashboards which give you powerful insights into your cloud spend, Harness Cloud Cost Management gives your teams the information they need to take control of their cloud spend.

Idle Resource Management:

One of the biggest sources of cloud waste is idle cloud resources, especially in test/dev environments. Your engineering teams don't work 24x7, so why would you want their cloud instances to? Shutting down idle cloud resources while they aren't in use can deliver major cost savings, but infrastructure teams struggle to automate this to reduce friction with engineering productivity.

Spot by NetApp does provide schedule based instance shut down, but it's little better than a manually maintained Lambda script. While this can save on cloud costs, it leaves a lot of cost savings on the table, especially for test resources that are not used consistently during working hours. Any deviations needed from the predetermined schedule need manual administrative attention to execute.

Harness Cloud Autostopping automatically detects and halts idle cloud resources when they aren't in use, and brings them back up again once an application or engineer accesses them. Our customers configure the Cloud Autostopping policy once, and we do the rest, freeing up DevOps teams to focus on more value added activities than managing manual shut-down schedules. Harness customers save up to 80% on the cloud costs with Cloud Autostopping.

Cloud Cost Management

Interested in seeing what's under the hood? Browse through the Harness Cloud Cost Management Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/blog/optimizing-facebook-s-proxygen-project-build-time-with-harness-ci-a-case-study>

Optimizing Facebook's Proxygen Project Build Time with Harness CI: A Case Study

Introduction

Open-source projects are a crucial aspect of software development, but their increasing size and complexity can result in longer build times. This can make it difficult to meet software release deadlines and carry out effective testing.

In this article, we will build Facebook's Proxygen project on Harness CI module and draw a comparison of the build time with Github Actions (Proxygen's Default CI Tool)

Proxygen: Facebook's C++ HTTP Libraries

Facebook's Proxygen project is a collection of C++ HTTP libraries including an easy-to-use HTTP server. In addition to HTTP/1.1, Proxygen (rhymes with "oxygen") supports SPDY/3 and SPDY/3.1. Proxygen is not designed to replace Apache or nginx â those projects focus on building extremely flexible HTTP servers written in C that offer good performance but almost overwhelming amounts of configurability. Instead, the focus is on building a high performance C++ HTTP framework with sensible defaults that includes both server and client code and that's easy to integrate into existing applications.

Setting up Proxygen on Harness CI

To set up the Proxygen project on Harness CI, fork the facebook/proxygen repository into your GitHub account and sign up for Harness CI if you haven't already. Once you're in the Continuous Integration module, follow these steps to create your pipeline:

- Click on "Create Pipeline" and give your pipeline a name. You can also provide a description and tags if desired.
- Choose the "Remote" option for setting up your pipeline and provide your Git connector.
- To create a new connector, go to "Connectors" and click on "+ New Connector."
- Select "GitHub" as the connector type and name it "proxygen" For the URL type, select "Repository" and provide the GitHub repository URL for your forked repository.
- Enter your GitHub username and a personal access token (PAT) as the secret value.
- Click "Save" to allow the repository to be fetched, then click on it and select "Apply Selected."
- Turn on API access and ensure the secret token is created.
- Click on "Connect through Harness Platform" and select your repository (proxygen) and the Git branch will be automatically fetched.
- Provide your YAML path as ".harness/{PIPELINE_NAME}.yml", where {PIPELINE_NAME} is the name of your pipeline.
- Finally, select "Start" to initiate the pipeline.

It is important to note that the root folder ".harness" is required for the YAML path. For more information on connectors, refer to the Harness CI documentation.

Here's the YAML to build the proxygen project:

Please modify the connectors in the pipeline YAML to use the ones you created, Â go to the Triggers tab of the newly created pipeline to

confirm the creation of two triggers, including a Pull Request trigger and a Push trigger. In this case, the Pull Request trigger should be enabled.

To compare the build time on GitHub Actions, enable the GitHub Actions workflow from the Actions tab on GitHub (the cloned repository already has the actions workflow file).

Finally, create a Pull Request with a few source or test file changes. This will trigger the Harness CI pipeline, as well as the GitHub Actions workflow if it's enabled.

Testing & Validating the Pipeline & Build Time

Here are the execution times for a same Pull Request triggered build on GitHub Actions and Harness CI.

Understanding the Pipeline Configuration file

The pipeline can consist of several stages, each of which is composed of one or more steps. In this case we have a single stage pipeline with the stage name as "Build" and has 5 steps: "Fetch Deps", "Build Deps", "Build Proxygen", "Copy Artifacts" and "Test Proxygen".

The "Fetch Deps" step fetches dependencies needed for building the project using the "getdeps.py" script. The "Build Deps" step builds the dependencies, and the "Build Proxygen" step builds the project itself.

The next step i.e "Copy artifacts" copies the built artifacts to a final installation location. The final step is called "Test Proxygen" and runs the tests for the project.

The pipeline is defined to run on a Linux operating system and uses the AMD64 architecture on Harness Cloud infrastructure

This pipeline uses a matrix strategy, which allows for parallel execution of steps with different dependencies. In this pipeline, the matrix strategy is used to define a set of dependencies required for building the "proxygen" project, and to execute the "Fetch Deps" and "Build Deps" steps in parallel for each dependency.

The matrix strategy is implemented over the dependency list, which in this case includes dependencies such as "ninja", "cmake", "zlib", etc.. For each dependency in the list, a parallel execution of the "Fetch Deps" and "Build Deps" steps is initiated. This means that if there are 10 dependencies in the list, then there will be 20 parallel executions of the "Fetch Deps" and "Build Deps" steps, 10 for each step.

By using the matrix strategy the pipeline can make efficient use of resources by executing multiple steps in parallel which can significantly reduce the overall time required to complete the pipeline. This is particularly useful in cases where there are multiple configurations or dependencies that need to be built, as it allows for a more streamlined and efficient execution of the pipeline.

Conclusion

Harness CI is built for speed and offers various parallelism strategies as well as high speed Cloud infrastructure to dramatically improve the build times. Harness also offers several other build and test optimizations to shorten build times (and cost) while also improving developer productivity.

Sign up for free and try it yourself!

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

[Documentation](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/case-studies/sensormatic-optimizes-retail-operations-with-harness-sei>

Sensormatic optimizes retail operations with Harness SEI

Overview

Sensormatic Solutions is the leading global retail solutions portfolio of Johnson Controls. It optimizes retail operations at scale, while enabling smart and connected shopper engagement. Our intelligent digital operating platform, Sensormatic IQ, combines the full Sensormatic Solutions portfolio, including third-party data to deliver unmatched insights into shopper experience, inventory intelligence, loss prevention and operational excellence with advanced technologies like AI and Machine Learning. The platform empowers retailers to act confidently with prescriptive and predictive data-driven insights and outcomes, when making key business decisions. Sensormatic Solutions prioritizes responsible retail practices, leveraging innovative and sustainable technology to meet retailers' evolving needs.

One of the key objectives for Sensormatic Solutions was to enhance the overall quality of the product and establish appropriate key performance indicators (KPIs) and metrics to monitor the organization's progress in terms of quality and efficiency improvements.

â

The ChallengeÂ

Sensormatic Solutions employs a diverse range of tools and processes across its engineering teams. In early 2021, an internal dashboard solution was initially developed and proved effective for a specific team. However, scalability challenges were encountered when attempting to extend its usage across multiple teams and varied tool sets. This, combined with a lack of integrated insights, starting from ticket creation through coding and deployment, prevented a comprehensive view of the Software Development Life Cycle (SDLC) and hindered the teamâs identification of bottlenecks.

The primary objectives of the Sensormatic Solutions engineering teams revolved around enhancing their delivery efficiency, quality, and cost-effectiveness. The organization sought comprehensive insights into its current quality status, process bottlenecks, change failure rates, and overall process hygiene. This necessitated the integration of insights from various tools utilized in its SDLC.

Furthermore, the teams were in the process of transitioning towards a unified, agile framework and desired a benchmark to assess its agile maturity, as well as a distinct framework for identifying process gaps, and effectively tracking adherence, to ensure successful agile implementation.

â

The Solution

Sensormatic Solutions utilized the SEI (Software Engineering Insights) to gain valuable and unique insights into agile methodologies, quality enhancement, and efficiency optimization. By leveraging this resource, it was able to standardize and normalize these insights across various teams, enabling data-driven improvements across the organization.

â

The Approach

DORA Benchmarking

Sensormatic Solutions recognized the importance of DevOps Research and Assessment (DORA) insights in achieving standardized Key Performance Indicators (KPIs). The organization prioritized obtaining insights from DORA, considering them essential elements for optimizing its software development processes.

Sensormatic Solutions strategically made DORA insights accessible at both the team and executive levels. This involved consolidating

data from various teams and integrating Jira and Bitbucket with Harness SEI. The seamless integration allowed Sensormatic Solutions to gain out-of-the-box DORA insights, empowering all teams to comprehensively track four key metrics crucial for their DevOps journey:

- **Lead Time for Change:** The duration from the initiation of a change request to its implementation.
- **Deployment Frequency:** The frequency of deploying changes to production, a critical measure of agility.
- **Mean Time to Recover (MTTR):** The average time taken to recover from failures, a crucial metric for system resilience.
- **Change Failure Rate:** The percentage of changes that fail, impacting system stability.

Agile Standardization and Maturity

As Sensormatic Solutions embarked on bolstering its agile delivery process, a set of foundational metrics was introduced to guide teams towards standardization and maturity. These metrics, tracked monthly, aimed to elevate the organization's agile practices and ensure a consistent, high-quality delivery process.

â

Hygiene Score for Process Conformance:

The hygiene score became a pivotal metric, assessing process conformance and maturity. Teams were scored based on their adherence to fundamental agile tenets, including:

- **Acceptance Criteria for Stories:** Ensuring that user stories had well-defined acceptance criteria, fostering clarity and understanding.
- **Story Pointing:** Accurately assigning story points to tasks, facilitating better planning and resource allocation.
- **Story Right Sizing:** Ensuring that stories were appropriately sized, preventing overcommitment, and optimizing delivery timelines.
- **Ticket Assignment:** Clearly defining and assigning tasks to team members, enhancing accountability and collaboration.
- **Work Allocation:** A refined process has been introduced to bolster work allocation reporting, providing valuable insights into feature development, technical debt management, and bug resolution. This systematic approach ensures alignment between allocated and actual percentages, serving as a critical checkpoint to maintain team accountability and project accuracy.

This quantified approach provided teams with a clear goal and a checklist, empowering them to actively work towards achieving and maintaining agile process maturity.

â

Scope Creep Measurement and Threshold:

Recognizing the impact of ad hoc requests on engineering planning and delivery, Sensormatic Solutions implemented a metric to measure and control scope creep.

A predefined threshold was set to ensure that ad hoc requests remained within acceptable limits. This strategic threshold aimed at maintaining consistency and safeguarding the quality of the delivery process.

By actively monitoring and limiting scope creep, Sensormatic Solutions aimed to foster a more controlled and predictable development environment, minimizing disruptions and ensuring the integrity of project timelines.

â

Commit to Done:

Completing committed work would be a key indicator of a team's comprehensive understanding of their tasks and accurate estimation of capacity.

Teams were encouraged to signal the completion of committed tasks, indicating a thorough understanding of the work and an ability to estimate their capacity realistically. This metric aimed to enhance delivery consistency by ensuring teams fulfilled their commitments, contributing to a culture of accountability and reliability in the agile delivery process.

â

Quality Scorecard

The team, recognizing the nuanced nature of software quality, delved into a meticulous exploration of multiple data points to construct a comprehensive quality scorecard. This scorecard comprised two sets of quality signals, capturing both leading and lagging indicators to provide a holistic view of the software development process.

â

Leading Indicators: Predictors of Quality Pre-Deployment

- **Initiative Hygiene or Process Hygiene:** Assessment of the overall hygiene in the development process, ensuring that agile and development practices align with established standards.
- **Coding Hygiene:** Evaluation of the codebase for adherence to best practices, readability, and maintainability, fostering a foundation for robust software.
- **Code Smells and Vulnerabilities:** Identification and measurement of undesirable code patterns (code smells) and potential security vulnerabilities, mitigating risks early in the development lifecycle.
- **Unit Test Coverage:** Measurement of the percentage of code covered by unit tests, ensuring comprehensive testing and early detection of potential issues.

- **Dev to QA Ratio for Features:** Evaluation of the balance between development and quality assurance efforts, promoting an efficient workflow and minimizing bottlenecks.

â

Lagging Indicators: Quality Implications Post-Deployment

- **Defect Escape Rate:** Calculation of the percentage of defects that escape detection during testing and manifest in the production environment, providing insights into the effectiveness of testing processes.
- **Defect Escapes per Story Point:** Measurement of the number of defects discovered in the production environment relative to the complexity of the implemented features, offering a refined understanding of the impact of defects on delivered functionality.
- **Change Failure (DORA):** Utilization of DORA (DevOps Research and Assessment) metrics as a lagging indicator, assessing the success or failure of changes post-deployment.

SEI Integration for Streamlined Insights and Reporting:

Leveraging Harness SEI, the team seamlessly integrated these quality signals, enabling the automatic generation of insights "out-of-the-box." This integration empowered Sensormatic Solutions to not only provide teams with valuable insights immediately, but also to ScoreCard each team based on these metrics. The scorecard, complete with Key Performance Indicators (KPIs), served as a tangible target for teams, fostering accountability and providing leadership with an executive dashboard for holistic quality assessment across all teams.

â

Results

Improved Coding Hygiene

Facing a challenge with Peer Review approvals, Sensormatic Solutions employed the power of Harness SEI for a transformative solution. Initially, 35% of Peer Reviews lacked formal approval, posing risks to code quality and architecture. With SEI's implementation, teams focused on establishing a robust Peer Review approval process.

Within months, Peer Review hygiene surged from 35% to an impressive 95%, while unapproved Peer reviews dwindled to just 5%. This turnaround not only mitigated the risk of merging suboptimal code, but also allowed early detection and resolution of architectural issues. The heightened attention to Peer Reviews instilled a proactive quality assurance culture, aligning with Sensormatic Solutions' commitment to delivering best-in-class software products.

Agile Process Standardization

Implementing a standardized set of Agile Hygiene Criteria proved pivotal for Sensormatic Solutions, catalyzing a significant evolution in their agile processes. As the engineering teams embarked on this journey, the majority initially scored in the low 30s out of 100. However, in a matter of months, there was a remarkable surge, with most teams achieving scores in the 80s.

This leap in Agile Hygiene scores was indicative of teams successfully honing their agile process skillsets. Notable achievements included:

- **Acceptance Criteria Enhancement:** Streamlined communication between Development and QA teams by reducing unnecessary back-and-forth. Well-defined acceptance criteria led to a shared understanding, minimizing ambiguities, and optimizing workflow.â
- **Consistent Story Points:** The establishment of consistent story points facilitated uniformity in planning and delivery. Teams experienced enhanced predictability and efficiency in their agile workflows.â
- **Done to Commit Ratio Exceeding 75%:** Adherence to process hygiene translated into a remarkable improvement in the Done to Commit ratio across all teams. This surge to over 75% indicated accelerated and more consistent product shipments.

The disciplined focus on Agile Hygiene Criteria not only standardized processes but also catalyzed a cultural shift, fostering faster and more reliable product deliveries. Sensormatic Solutions' journey exemplifies how strategic process standardization can elevate agile practices, leading to enhanced team efficiency and consistent project success.

â

Lead Time Efficiency Gains

During DORA benchmarking at Sensormatic Solutions, inefficiencies surfaced in the SDLC process, specifically in the testing phase. Notably, testing time per feature emerged as a critical bottleneck. To address this, the team implemented a strategic approach, combining efforts in coding hygiene with a dedicated focus on automating testing processes.

The outcome was a substantial 30% reduction in testing time, resulting in a corresponding lead time gain for shipping features. Crucially, this efficiency enhancement occurred without compromising product quality. The streamlined testing process not only accelerated feature

deployment but also underscored Sensormatic Solutions' commitment to delivering high-quality software.

Quality Benchmarking

Armed with well-defined leading and lagging indicators for quality, Sensormatic Solutions implemented a robust quality benchmarking initiative. The organization successfully introduced a standardized Quality ScoreCard, meticulously assessing each team on a scale of 1-100 across various quality dimensions. This comprehensive evaluation extended to both leading indicators, reflecting pre-deployment processes, and lagging indicators, gauging post-deployment outcomes.

The standardized executive reporting, facilitated by the quality scorecard, empowered leadership with a clear overview of each team's performance. This not only streamlined decision-making processes but also enabled teams to pinpoint specific areas for immediate improvement based on their scores. The scorecard served as a dynamic tool for adopting targeted quality initiatives, fostering a culture of continuous improvement across the organization.

â

Investment Analysis

With the introduction of a cohesive software delivery process, Sensormatic Solutions achieved a profound understanding of its investment priorities, categorized into growth, sustainability, and efficiency. This newfound clarity empowered the team to enact necessary changes based on actionable insights derived from the process.

Harnessing the capabilities of the SEI platform, Sensormatic Solutions delved into the intricacies of each module within the software delivery process, encompassing workshop, discovery, development, testing, and deployment phases. The platform played a pivotal role in gauging the punctuality of deliverables, rendering teams accountable and instilling a sense of ownership across the entire development lifecycle.

The data-driven approach facilitated not only the celebration of successes,Â it focused attention to areas requiring improvement. Sensormatic Solutions' strategic utilization of the SEI platform underscores its commitment to efficiency, adaptability, and continuous improvement in the realm of software delivery.

â

Conclusion: Elevating Software Delivery Excellence

Sensormatic Solutions' journey with Harness SEI exemplifies a commitment to excellence in software delivery. From enhancing agile processes to streamlining testing, and fostering coding best practices, the transformative impact is evident.

The organization's adept use of SEI insights for investment prioritization, coupled with a unified software delivery approach, demonstrates a forward-thinking stance on growth, sustainability, and efficiency that will positively impact its employees and customers going forward.

Teams, armed with punctuality metrics and a sense of ownership, are celebrating their successes and continuing to tackle areas for improvement with haste and precision.Â

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps

engineer efficiency.

Advanced Achieves Cloud Cost Governance Excellence, Saves 33% on Cloud Costs with Harness CCM

Advanced was struggling with cost overruns, and lack of adoption of the previous cost management tool. After implementing Harness Cloud Cost Management, Advanced has seen 33% annualized cost savings to-date.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/products/security-testing-orchestration/features>

A Look At Harness Security Testing Orchestration

See how Harness STO's extensive features pave the way for developers, devops, and security practitioners to ship secure code at high velocity

Detailed, intuitive dashboards and reports

STO's intelligent scanner analysis engine helps developers categorize new and existing vulnerabilities, analyze and deduplicate results across multiple security scanners, and prioritize remediation of the most critical security findings.

Intelligent deduplication & issue prioritization

STO's intelligent scanner analysis engine helps developers categorize new and existing vulnerabilities, analyze and deduplicate results across multiple security scanners, and prioritize remediation of the most critical security findings.

Seamless AppSec scanner & workflow integrations

Harness Security Testing Orchestration makes it easy for users to consume their security scanners of choice via both native and custom integrations. Integrate with 40 commercial and open-source scanners natively. In addition, users can integrate with workflow or issue-tracking systems either in the pipeline or against identified vulnerabilities.

Collaborative exemption management for developers and security practitioners

Security exemptions are an important consideration for a secure software development lifecycle. Harness Security Testing Orchestration allows security stakeholders to grant and manage exemptions for vulnerabilities and other issues surfaced by security scans which may not be actionable, or would otherwise bottleneck CI/CD processes.

Governance policies based on OPA

Harness Security Testing Orchestration empowers teams to enforce governance as part of the CI/CD pipeline with customizable policies based on the Open Policy Agent OPA. This provides flexibility to define governance policies as needed across the organization and ensure that the code being deployed meets the organization's security standards or compliance requirements.

Enterprise-grade audit trails and RBAC

Harness STO generates highly-detailed audit trails, dramatically reducing audit processes from several days to just a few hours. Harness also offers fine-grained RBAC, allowing you to tailor your permissions system to meet your organization's needs.

Trusted by DevOps and Developers

Hundreds of DevOps and engineering teams are powered by Harness to become elite performers in velocity, quality, efficiency, and governance.

deluxe

Using Harness Security Testing Orchestration for a single pipeline, Deluxe identified 170 issues from a scanning vendor, narrowed to nine prioritized problems post-deduplication. The team highlighted a 95% noise reduction, allowing efficient focus on top issues.

Learn more about

Harness Security Testing Orchestration

Product Documentation

Learn how to connect SEI with your existing tech stack and get insights. How to remove bottlenecks and improve planning and sprint hygiene

Product Updates

See our latest feature releases, product improvements and announcements

Blogs

See our latest feature releases, product improvements and announcements

Case Studies

Sign up for a free 14 day trial and take your software development to the next level

Source URL: <https://www.harness.io/case-studies/h2o-ai-saves-35-000-in-first-30-days-with-cloud-asset-governance-from-harness-cloud-cost-management>

H2O.ai Saves \$35,000 in First 30 Days with Cloud Asset Governance from Harness Cloud Cost Management

Harness Impact

- Saved \$35,000 in first 30 days
- Eliminated up to 15 hours per week of manual cost reporting
- Detected and deleted unused Amazon Elastic Block Storage (EBS) volumes

About

Recognized as a global visionary and thought leader in automated machine learning (autoML), time series forecasting, and responsible AI, H2O.ai is the trusted AI partner to more than 20,000 organizations around the world. Its platform, the H2O AI Cloud, enables businesses, government entities, nonprofits and academic institutions to accelerate responsible innovation and push the boundaries of what's possible with artificial intelligence.

Challenge: Improve insights into resource utilization and attribution

As a company that thrives on democratizing AI and leveraging open source solutions, H2O.ai equally values its ability to build its cloud generative AI solutions in its customers' cloud environments and to open up H2O.ai's own cloud environments for testing. That's a great service for customers, says Michal Malohlava, H2O.ai VP of Engineering. But it also meant that H2O.ai needed to gain greater visibility into cloud costs, and track usage and budgets by business unit.

Finding answers to questions about cloud costs wasn't as simple and efficient as H2O.ai wanted it to be. The large and complex cloud environments, spread across AWS, Azure, and Google Cloud Platform (GCP), complicated the task of attributing costs to the correct business unit. In addition, generative AI adoption was exploding across all sectors of technology, increasing the cloud budget for H2O.ai.

To serve customers, H2O.ai was also using costly resources such as GPUs to run AI and machine models, and wanted to view costs and resource utilization in real time. But AWS Cost Explorer and the native tools in Azure and GCP couldn't break down those costs accurately, leaving most of the bill attributed to engineering.

Solution: Cloud Asset Governance to identify and eliminate waste

With a goal to ensure that every cloud resource was in active use, H2O.ai adopted Harness Cloud Cost Management (CCM). Malohlava has deployed CCM in stages, beginning with simple cost tracking and automated notifications about cost anomalies, and later addressing governance and idle resources with CCM Cloud Asset Governance.

With CCM Cloud Asset Governance, Malohlava can dig deeper into inefficient use of cloud resources, proactively manage these resources, and ensure that all cloud resources match corporate standards and policies. Harness CCM enables H2O.ai to label machines and assign owners to them, which helps stakeholders better attribute costs to the proper owners.

Harness CCM also looks for trends in overspending or overestimating the necessary cloud resources. For example, costs for Amazon EBS volumes made up about \$2.5 million—half of the company's annual cloud operating costs—because teams were overestimating the

size of the volumes needed. Harness CCM and its Cloud Asset Governance feature helped Malohlava detect monthly spend on EBS volumes of \$32,000 per month, of which 60% was unused and then delete the unused volumes.

Result: \$35,000 cost savings in first 30 days

To date, H2O.ai has saved \$87,000 in the first 4 months of using Harness CCM by leveraging the automated governance policies in Cloud Asset Governance to automatically find and eliminate cloud waste. H2O.ai is also using Cloud AutoStopping™ to dynamically shut down idle VMs and containers on all root cloud accounts that are connected to Harness, and then run them on fully orchestrated Amazon EC2 Spot instances.

Malohlava saves time as well as money. Before H2O.ai began using Harness, he spent about 15 hours per week creating and managing cloud cost reports. With Harness, Malohlava can automatically generate the reports and set alerts for budget overruns.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform®

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/advanced-achieves-cloud-cost-governance-excellence-saves-33-on-cloud-costs-with-harness-ccm>

Advanced Achieves Cloud Cost Governance Excellence, Saves 33% on Cloud Costs with Harness CCM

Advanced was struggling with cost overruns, and lack of adoption of the previous cost management tool. After implementing Harness Cloud Cost Management, Advanced has seen 33% annualized cost savings to-date.

Harness Impact

- 33% annualized cost savings and growing
- Proactive cloud cost governance
- Improved understanding of Cost of Goods Sold (COGS)
- Reduced budget overruns with proactive cost anomaly detection

About

Advanced is a leading provider of sector-specific software that powers the world of work, effortlessly getting the job done and giving their customers the freedom to focus on thriving for their customers and their people. Headquartered in the UK, Advanced employs

2,000 people.

As a trusted partner to their customers, Advanced encourages their customers to think differently and adapt to the changing needs of their business. Powered by their cloud-first software, they streamline processes, boost productivity, make insightful decisions, and provide their people with the right tools to excel.

Challenge: Lack of cost management tool adoption, 15% overruns on budgets

With a company-wide initiative to automate processes, the development team at Advanced sought out Harness Continuous Delivery (CD) to empower their 700+ engineers to streamline software delivery. This resulted in an average deployment time reduction of 88%.

Similarly, for improving cloud costs and governance, Jay Patel, Director of Platform Engineering-DevOps at Advanced, evaluated the effectiveness of their existing cloud cost management tool. She shared that, âWe had what we felt was good cloud governance. We followed the sun with how we started and stopped service: starting them up at 8am India time, finishing them at 7pm UK time, and not running them on the weekends. This turning on and off of machines required manual scripting and effort.â

Unfortunately, the biggest drawback was that their existing cloud cost management tool suffered from a lack of adoption since it wasn't integrated with their continuous integration (CI) and CD tools. The team only looked at it when they went to buy Reserved Instance (RI) contracts, but did not use it beyond that feature. This rendered potential right-sizing optimizations and cost savings out of reach.

Additionally, product owners weren't paying attention to the set budgets because the cloud cost management tool in place wasn't intuitive to them. They were averaging up to 15% overruns on budgets, making changes without understanding their impact, and exceeding the threshold for cost of goods sold (COGS). With over 40 products on AWS alone, Advanced needed to understand the COGS vs operations costs as well as forecast cloud spend accurately to ensure cost efficiency.

The search for a better tool was driven by the need for clear insights into resource wastage and unexpected cost fluctuations, the ability to enable teams to take ownership of their own product costs, and the ability to monitor long-term savings.

Solution: Proactive cost anomaly detection, visibility across services

In May 2022, the DevOps team implemented Harness Cloud Cost Management (CCM), making the commitment to save at least 10% on annual cloud costs beyond any savings they had been getting with scheduled shutdowns. Patel commented, âThe process of implementing Harness CCM was fairly direct. We connected to all of the accounts from the one master AWS account, and then we had visibility into costs immediately.â

The ease of onboarding Harness CCM allowed the 80-person DevOps team to focus on implementing recommendations and the three-person governance team to accelerate adoption. They used anomaly detection alerts so that product owners are notified of potential overspend. They brought costs from shared services into the Perspectives view in Harness CCM, giving them visibility across all services. Furthermore, they leveraged Cost Categories to show shared costs without having to worry about tagging health.

Result: 33% annualized cost savings with recommendations and Cloud AutoStoppingâ€

The DevOps team started seeing cost improvements within the same month of implementation. Between recommendations, improvement in cost visibility and Cloud AutoStoppingâ€, Advanced has seen 33% annualized cost savings to-date.

In fact, by eliminating the follow-the-sun model and ensuing manual scripting, Cloud AutoStopping actively manages cloud resource idle time effectively, shutting down idle resources when not in use and dynamically running on spot instances with no impact to end users. Patel remarked that âWith the Harness Cloud Asset Governance feature, we are now automating cost governance, driving half of the savings from the part that we thought was already well-governed. That governance-as-code approach includes real-time enforcement and auto-remediation, and quite frankly achieved even greater savings.â Moreover, Cloud AutoStopping has also encouraged developers to increase their focus on cost management, as it is now incorporated into the culture.

Additionally, instead of waiting to address a cost anomaly when the cloud bill arrives days or weeks after the fact, the team can now catch it before it overruns the budget. This let Advanced adopt a proactive approach to managing cloud costs, transforming it from an implementation project to a business-as-usual activity. Patel shared that, âHarness CCM is catching a lot more overruns, including those that are smaller â not to mention enabling active reviews of costs during every sprint meeting.â

Advanced has also created a set of standard service catalogs with known cost architectures based on the learnings from Harness CCM. Combined with asset governance policies, the team actively prevents cases where engineers create non-compliant cloud instances, such as CI/CD pipelines that aren't cost compliant.

With a stronger handle on COGS, Advanced can make strategic pricing decisions. Not only have they discontinued products that are too expensive to operate, but they have also approached new application architecture more efficiently.

Patel points out, âAt Advanced, we felt like we had a good handle on cloud cost governance. Then, we implemented Harness Cloud Cost Management. And what we actually discovered was that we were just okay. Harness CCM took us to excellence, and it has made huge savings for us!â

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/comparison-guide/github-actions-vs-harness>

Comparison Guide

Harness

Continuous Delivery & GitOps

VS

GitHub Actions

Harness Continuous Delivery & GitOps vs GitHub Actions

Continuous Delivery & GitOps

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

GitHub Actions

GitHub Actions makes it easy to automate all your software workflows, now with world-class CI/CD. Build, test, and deploy your code

right from GitHub.

501+

2018

Acquired

GitHub Actions is categorized as:
Continuous Delivery

What is the difference between Harness CD Vs. Github Actions?

GitHub Actions vs Harness: DevOps Tools Comparison

Updated

November 30, 2023

- **SaaS & On-Premises**
- **No Scripting Required**
- **Ease of Use**
- **Cloud-Native App Support**
- **Traditional App Support**
- **Canary Deployments**
- **Infrastructure Provisioners**
- **GitOps (Pipelines as Code)**
- **Continuous Verification ↗**
- **Change Management Jira/SNOW**
- **Role-Based Access Control**
- **Secrets Management**
- **Audit Trails**
- **Accelerate Metrics & Reporting**

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

CloudFormation and Terraform

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<with><with>

<yes><yes>

<yes><yes>

<yes><yes> **With Caveats**

CloudFormation and Terraform

```
<yes><yes>
<no><no>
<with><with>
<no><no>
<yes><yes>
<yes><yes>
<no><no>
<yes><yes>
<yes><yes>
<yes><yes>
<no><no>
<yes><yes>
<with><with>
<yes><yes>
<no><no>
```

SaaS & On-Premises:

GitHub Actions used to be SaaS-only, but as of May 13th (just 11 short days ago at the time of this writing), a version was made available for customers on GitHub Enterprise Server 3.0 or later. From our understanding, there are some minor limitations, but most actions are available on the on-prem version. Harness provides SaaS and on-prem versions of the product, with feature parity between both.

No Scripting Required:

GitHub Actions does unfortunately require a decent amount of scripting. The good part is that if you do have to write an action, you can publish it as a container image, so you can write it in whatever language you want. There are also lots of pre-written actions out there on the actions marketplace, including integrations to plenty of tools. While scripting isn't the bane of our existence, it is time that could be spent shipping code. It is toil. It's something that needs maintenance by your brightest engineers. Harness is much more intuitive and takes away the burden of scripting.

Ease of Use:

You can get up and running with GitHub Actions fairly quickly. Assuming you have a GitHub repo already, you can add your first workflow into the repo and your action should run. Managing secrets is painless: you have the ability to set your secrets at the organization level as well as at the repo level, which is relatively flexible. One of the larger sticking points is that the way you pass data around your workflow between jobs can be a bit tricky at first. There's a learning curve. When trying to pass data between actions and jobs, you'll find yourself needing to use cryptic action syntax or direct .env file insertion to chain each job's state together. With Harness, there's no scripting needed and configurations are passed to pipelines securely and in a pragmatic way. Like most CI/CD providers, GitHub Actions has third-party actions that you can use as semi plug-and-play functionality. You'll find that it's hit or miss whether or not a task has already been solved by a publicly-available GitHub Action â otherwise you'll need to write your own. Given that, you'll find yourself needing to code a lot when working with GitHub Actions. In comparison, Harness offers containerized plugins featuring a vast amount of integrations â no maintenance, no writing â only simple, scalable, self-service extensibility. Lastly, because GitHub Actions is a newer offering, it still doesn't have any sort of good reporting and oversight on what's happening outside of a given repo. There's no high-level overview of watching executions with a timeline or native deployment tracking, which is something Harness gives you. With GitHub Actions, you'll be building everything yourself â besides where you get lucky and find a pre-written action.

Canary Deployments:

An Azure user put together a helpful tutorial on how to do canary deployments with GitHub Actions and Azure using Linkerd. It is possible, but as the user noted, âAn end-to-end solution, fully automated and based on metrics, in a more complex production application is quite challenging. [â] While I think the GitHub Action works well, I am not in favor of driving all this from GitHub, Azure DevOps, and similar solutions. There's just not enough control.â The adage âJust because you can doesn't mean you shouldâ remains true here. Harness provides guided Canary deployments out of the box â no coding required, only some minor config.

Infrastructure Provisioners:

HashiCorp created an action to set up and configure the Terraform CLI in your GitHub Actions workflow. There is also an action for CloudFormation. Harness offers both infrastructure provisioners â with a simpler setup and configuration process.

Continuous Verificationâ:

Continuous Verification is the process of monitoring your app for abnormalities after a deployment. For example, Continuous Verification could catch a latency issue or 5xx errors and automatically roll back your app to the previous version. The idea is to catch errors as quickly as possible â ideally, before customers notice â and make a seamless transition back to the prior version. We found some basic deployment status actions out there, but those only tell you if a deployment passed/failed, not how it's doing post-deployment. It seems that to have complete visibility into deployments, you would need to write your own actions to achieve this. There are integrations into some observability tools like Lightstep, so it could be done â manually. Harness, however, provides Continuous Verification out of the box, effectively reducing risk and reputational damage from downtime. Harness supports many vendors, including Prometheus, Datadog, AppDynamics, New Relic, StackDriver, CloudWatch, and custom monitoring and observability tools.

Change Management Jira/SNOW:

Atlassian has created an action to integrate Jira into your workflow. From the actions available, though, it seems more for recordkeeping purposes than an approval mechanism. The SNOW action is much more comprehensive, allowing for publishing and rollback of apps. Harness offers full integrations for Jira and SNOW, insofar as they can be used for recordkeeping purposes and as approval mechanisms.

Secrets Management:

GitHub Actions provides different ways of scoping secrets. For one, secrets can be set up at a global organization-wide level. Instead of having to update secrets across a number of Git repositories, they can be handled in a centralized place. Additionally, you can restrict whether secrets are exposed to forked repositories or not. Onto the downsides: GitHub Actions doesn't have granular access control mechanisms. Also, secrets in GitHub Actions are natively handled for CI/CD â third parties, like HashiCorp Vault, cannot be leveraged. While Harness does offer its own secrets management solution, it also integrates with third party vendors like HashiCorp Vault, Amazon Secrets Manager, Google Secret Manager, AWS Key Management Service, Google Cloud Secret Manager, CyberArk, and Azure Key Vault.

Accelerate Metrics & Reporting:

There are four key metrics when it comes to software development: Lead Time (the average amount of time it takes from the time code is checked in to the version control system to the point in time where it is deployed to production), Deployment Frequency (the number of times deploys to production occur in a time period), Mean Time to Restore (MTTR: how long it takes to resolve or rollback an issue in production), and Change Failure Rate (what percentage of changes to production fail). These metrics are paramount in truly

understanding performance. GitHub Actions does not provide native Accelerate metrics dashboards. Harness offers a beautiful dashboard specifically for these metrics and allows you to set alerts as needed â for example, you could set an alert to notify you if the Change Failure Rate goes above 1%.

***Please note: Our competitors, just like us, release updates to their products on a regular cadence. We keep these pages updated to the best of our ability, but there are bound to be discrepancies. For the most up-to-date information on competitor features, browsing the competitorâs new release pages and communities are your best bet.**

Try Harness For Free

Continuous Delivery & GitOps

Interested in seeing what's under the hood? Browse through the Harness Continuous Delivery & GitOps Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/comparison-guide/spinnaker-vs-harness>

Comparison Guide

Harness

Continuous Delivery & GitOps

VS

Spinnaker

Harness Continuous Delivery & GitOps vs Spinnaker

Continuous Delivery & GitOps

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

Spinnaker

Spinnaker is an open source, multi-cloud continuous delivery platform for releasing software changes with high velocity and confidence.

â

2015

â

Spinnaker is categorized as:
Continuous Delivery

What is the difference between Harness CD Vs. Spinnaker?

Updated

November 30, 2023

- SaaS & On-Premises
- No Scripting Required
- Ease of Use
- Cloud-Native App Support
- Traditional App Support
- Canary Deployments
- Infrastructure Provisioners
- GitOps (Pipelines as Code)
- Continuous Verification
- Change Management Jira/SNOW
- Role-Based Access Control
- Secrets Management
- Audit Trails
- **Accelerate Metrics & Reporting**

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

CloudFormation and Terraform

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

â

<yes><yes>

On-Premises Only

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

CloudFormation and Terraform

Terraform

```
<yes><yes>
<yes><yes>
<yes><yes>
<with><with>
<yes><yes>
<with><with>
<yes><yes>
<yes><yes>
<yes><yes>
<with><with>
<yes><yes>
<with><with>
<yes><yes>
<no><no>
```

SaaS & On-Premises:

Spinnaker is only offered on-prem â no SaaS option (though, Armory does have a beta in progress to offer a SaaS version of Spinnaker). Harness delivers both, so no matter your use case, we're ready and able to help.

Ease of Use:

Spinnaker is a nightmare to set up and configure, even for organizations with relatively simple setups. The difficulty increases exponentially for organizations with complex CD processes. We'll happily quote Armory admitting, âIt often takes 2-4 full-time engineers to install, operate and configure Spinnaker on-prem.â There are also gaps in the documentation, so finding answers to questions can be a burden on engineers, who often are left to figure it out by themselves. Prior to July 2020, plugins didn't exist for Spinnaker. Armory, the enterprise-grade version of Spinnaker, created a plugin framework for Spinnaker â which is good progress in terms of opportunities for extensibility. However, there is a very limited list of available plugins. Other than this repo of example plugins, creators are expected to host plugins in their own repos and choose to make them publicly available or not, so there is no main easy-to-find central plugin index. Alternatively, the Harness platform is designed with simplicity in mind. Engineers can get simple pipelines running in 5 minutes, encouraging adoption and ensuring an easy transition/migration.

Traditional App Support:

Spinnaker provides no support for traditional apps in Java or .NET, among other technologies. It was designed to be cloud-native only. Harness supports many traditional apps, including custom creations.

Infrastructure Provisioners:

For a very long time, Spinnaker didn't offer a Terraform integration. They finally went GA with their Terraform integration last year. Their integration is roughly the same as how most users leverage the Terraform process: download the Terraform configs and run the commands. Harnessâ Terraform integration is more robust and gives users advanced templating, which makes it easier to manage and scale. Additionally, Harness easily pairs the execution of Terraform to the delivery pipeline, especially for ephemeral environments. Lastly, we also offer a CloudFormation integration, if that's your provider of choice. All other infrastructure provisioners are supported via shell script.

Continuous Verification:

Continuous Verification is the process of monitoring your app for abnormalities after a deployment. For example, Continuous Verification could catch a latency issue or 5xx errors and automatically roll back your app to the previous version. The idea is to catch errors as quickly as possible â ideally, before customers notice â and make a seamless transition back to the prior version. There are tools that can be added to Spinnaker to provide deployment verification. Harness, however, provides Continuous Verification out of the box, effectively reducing risk and reputational damage from downtime. As for vendor integrations, Harness supports Prometheus, Datadog, AppDynamics, New Relic, StackDriver, CloudWatch, and custom monitoring and observability tools.

Change Management Jira/SNOW:

Spinnaker supports Jira change management. Harness offers both Jira and ServiceNow integrations.

Secrets Management:

Spinnaker doesn't offer built-in secrets management, but they do support HashiCorp Vault, Google Cloud Storage, AWS S3, and the AWS Secrets Manager. Harness, on the other hand, offers proprietary, integrated secrets management. No third parties are required, but all of the major secrets managers are supported.

Audit Trails:

Armory, the enterprise-grade version of Spinnaker, does offer audit trails, but the open-source Spinnaker itself does not. Harness provides audit trails on every pipeline, workflow, step, execution, and change. It's all audited by Harness so you have a complete trail of all user activity.

Accelerate Metrics & Reporting:

There are four key metrics when it comes to software development: Lead Time (the average amount of time it takes from the time code is checked in to the version control system to the point in time where it is deployed to production), Deployment Frequency (the number of times deploys to production occur in a time period), Mean Time to Restore (MTTR: how long it takes to resolve or rollback an error in production), and Change Failure Rate (what percentage of changes to production fail). These metrics are paramount in truly understanding performance. Spinnaker does not offer any Accelerate metrics tracking or reporting. Harness offers a beautiful dashboard specifically for these metrics and allows you to set alerts as needed - for example, you could set an alert to notify you if the Change Failure Rate goes above 1%.

**Please note: Our competitors, just like us, release updates to their products on a regular cadence. We keep these pages updated to the best of our ability, but there are bound to be discrepancies. For the most up-to-date information on competitor features, browsing the competitor's new release pages and communities are your best bet.*

Try Harness For Free

Continuous Delivery & GitOps

Interested in seeing what's under the hood? Browse through the Harness Continuous Delivery & GitOps Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/burst-sms-accelerates-application-modernization-with-harness>

Burst SMS accelerates application modernization with Harness

Burst SMS has accelerated its application modernization journey with Harness, achieving cost savings, improved reporting, efficient deployments, and positioning itself for future growth.

Introduction

Burst SMS is a leading technology company at the forefront of the SMS marketing industry. Their SMS gateway enables customers to increase sales, retain more customers, and streamline operations. Recently, Burst SMS has found success expanding their APAC connectivity, achieving a new milestone - with over 20 billion conversations for over 35,000 businesses.

However, at the core of their business was a monolithic application written in PHP. As the company grew, it became clear that to continue scaling the business and improve agility, they needed to modernize their application.

The Challenge: Modernizing a Monolithic Application

The decision to modernize their application was not without its challenges. After a failed attempt at a complete rewrite, Burst SMS decided to employ the strangler pattern, rewriting portions of the application as services while leaving the core in place. This approach, while effective, introduced new complexities. With more moving pieces, cloud costs grew and became harder to predict, manage, and understand.

Simultaneously, Burst SMS was undertaking a Herculean task of consolidating from seven brands into one, requiring surgical precision to ensure a seamless transition for their customers.

Harness: DevOps for Application Modernization

In these challenges, Burst SMS turned to a familiar ally: Harness. Having previously wielded Harness to reduce system downtime from one hour to mere minutes, Burst SMS expanded its usage of Harness to navigate this modernization journey.

Harness was now the backbone for building and deploying 30 services, not just monolith deploys. Burst SMS also adopted the Harness Cloud Cost Management (CCM) module, a solution designed to provide visibility and control over cloud costs.

Harness Impact: Happy Engineers, Cost Savings, Improved Reporting, and Efficient Deployments

The implementation of the CCM module yielded swift and substantial benefits. Through its automation capabilities, it facilitated the automatic shutdown of test environments outside of operational hours. This led to a 66% reduction in environment-related costs, contributing to notable financial savings. Furthermore, the CCM module greatly simplified reporting processes. In the pre-CCM period, the arrival of the monthly cloud bill would trigger a time-consuming investigation process, followed by a lengthy meeting to discuss the findings. However, with CCM, anomalous charges were detected within a single day. Additionally, the system seamlessly generates regular reports, offering clear explanations of any modifications made. Consequently, the once-dreaded and time-consuming monthly meetings were no longer necessary, replaced by an automated and efficient process that streamlines reporting and enhances cost-efficiency.

Moving to Harness's build infrastructure for Continuous Integration (CI) also delivered significant cost savings. Previously, Burst SMS had to maintain their own cluster for builds and related activities. The cloud costs alone were \$110,000 per year. By moving to Harness CI, builds accelerated and those infrastructure costs were cut by 76%, saving over \$80,000 annually.Â

Harness also brought improvements to Burst SMS's deployment processes. By using team tags alongside the native Slack integration, failure notifications were sent directly to the impacted teams. This reduced noise and improved responsiveness, ensuring errors were dealt with quickly.

One of the most significant improvements came from adopting the blue/green deployment technology built into Harness Continuous Delivery (CD). According to Thomas Wilson, SRE Manager at Burst SMS, "My CEO was next to me when an update brought our whole service down. He was shocked at how calm I was. Using Harness, we had everything back up in a couple of minutes. No harm done." Before Harness, outages could take two hours. Now, with blue/green deployments, that's down to under five minutes, a 96% improvement.Â

Conclusion: Harness Empowers Burst SMS's Future Growth

Harness has been instrumental in Burst SMS's application modernization journey. It has helped manage complexity, reduce costs, and improve deployment processes. As a result, Burst SMS has been able to deliver a customer experience that "just works," reducing stress for the engineering team and positioning the company for future growth and success. With Harness, Burst SMS is well-equipped to face the future, ready to take on new challenges, and seize new opportunities.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

Introduction

Harness played a pivotal role in the transformation of Vivun, the world's leading PreSales software provider. It has enabled Vivun to evolve DevOps strategy to embody efficiency, speed, and innovation in its software delivery processes.

Milestone Achievements:

- Advanced from monthly to on-demand release cadence
- 300% improvement in DevOps engineer efficiency
- Consolidated DevOps toolchain

About Vivun

As the global leader in PreSales software, Vivun's AI-driven platform is the go-to solution for PreSales Operations, Demo Automation, and bridging the gap between Sales and Product teams. By leveraging Vivun, revenue leaders can drive growth efficiently by scaling their technical selling efforts, de-risking their deals, and ensuring every R&D dollar is well spent.

Challenge

Before adopting Harness, Vivun's DevOps landscape was fragmented. Jon Call, their Engineering Manager for SRE, recalls, "We wanted a more cohesive DevOps strategy."

Vivun's monthly release cadence was a significant event, with developers and top executives joining calls to approve proposed changes verbally and manually test them in production. Releases were risky, stressful, and a lot of work for everyone.

Previously, the company was using another vendor for builds and deployments. However, the configuration files with that solution had ballooned to unmanageable sizes. The makeshift infrastructure bridging the solution to Vivun's internal environment was brittle and cumbersome.

The Transition

Recognizing these inefficiencies, Vivun embarked on a transformative journey. They began by establishing a dedicated SRE team to reduce production toil and enhance their DevOps delivery. In parallel, they expanded their QA team, with a focus on automating testing.

Initially, developers would send changes to the QA team and wait for the results. Working together, the SRE and QA teams ensured that tests were orchestrated by the pipeline and automated seamlessly. Jon highlighted the shift in their approach, stating, "Now nobody waits for tests, and testers don't run tests—they build them." Because the pipeline provides a common execution interface, Vivun has been able to change testing frameworks as needed without impacting the developers.

At first, Vivun used Jenkins for builds and Harness for deployments. However, as they delved deeper into the capabilities of Harness, they realized the potential of consolidating their processes. The transition to Harness CI for builds simplified their operations immensely. Jon compared their experience with the Jenkins Templating Engine plugin, stating, "Templates moved the needle, but Harness takes reusability miles further by making it easier to build templates at the right levels of abstraction, and making it easier for developers to consume them."

Results

Harness's adaptability, combined with its out-of-the-box solutions, has been a revelation for Vivun. Jon proudly states, "Today, we ship on demand, not on monthly calls." Harness's pipeline structure's clarity and efficiency have marked a considerable improvement over their previous experience.

One of the standout results has been the efficiency in resource allocation. Previously, one DevOps engineer was needed for every 20 to 30 developers. Now, a single person can support 50 to 75 engineers. Jon adds, "Harness is like having extra team members. Without Harness, there's no way my team could support the company's updated core platform and the many products we offer."

Developers are also more effective with self-service. Using templates, they can now set up a robust deployment in mere minutes. Furthermore, when something goes wrong, the intuitive UI of Harness empowers Vivun's developers to debug swiftly and independently. Instead of hunting around for an error in Jenkins, they pop into Harness and see the error lit up red in their pipeline.

The Future with Harness

Vivun's journey with Harness is far from over. They are gearing up to integrate Harness Feature Flags, aiming to replace an in-house tool.

Moreover, they are keen to leverage Harness for MLOps use cases for their newer Large Language Model (LLM). While data science workflows have some unique wrinkles, the core structure of testing versioned artifacts and either promoting or rejecting them is a great fit.Â

Conclusion

Harness has been the linchpin of Vivun's DevOps transformation, enabling them to scale advanced DevOps practices with unparalleled efficiency. With Harness by their side, Vivun is poised to continue its trajectory of innovation and excellence in the Sales software domain.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Advanced Achieves Cloud Cost Governance Excellence, Saves 33% on Cloud Costs with Harness CCM

Advanced was struggling with cost overruns, and lack of adoption of the previous cost management tool. After implementing Harness Cloud Cost Management, Advanced has seen 33% annualized cost savings to-date.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/tyler-technologies-reaches-annualized-cost-savings-over-1-million-with-harness-cloud-cost-management>

Tyler Technologies reaches \$1.2M annualized cost savings with Harness Cloud Cost Management

Learn how Tyler Technologies reached \$1.2M annualized cost savings with Harness Cloud Cost Management

About Tyler Technologies

Tyler Technologies is the largest SaaS vendor in the United States solely devoted to the public sector. The companyâs Enterprise Permitting & Licensing solution is used by government agencies to automate and streamline community development and business management operations. Overall, Tylerâs products empower the public sector to create smarter, safer, and stronger communities.

Challenges: Skyrocketing costs, ad hoc deployment strategy, and limited opportunities for automation

When you visit the website for your local government to get a license, make a payment, or respond to a summons, chances are you're using software developed by Tyler Technologies.

The leading U.S. software provider to the public sector, Tyler Technologies manages over 40,000 client deployments in nearly 13,000 locations across the country. Its cloud footprint is enormous, and as a significant one of the highest cost input to the business, it's a footprint that Tyler is constantly trying to reduce.Â

Each Tyler Technologies Enterprise Permitting & Licensing customer has a set of non-production environments, on non-production infrastructure, organized into âpodsâ and used to test Tyler software before deploying into production environments. This is a critical feature for Tyler, given the regulatory nature of its permitting and licensing services for government agencies: New features and bug fixes must be fully tested and vetted before pushing to production.

AWS, Tyler's primary cloud provider, offers tools such as CloudWatch to help monitor and control costs, but the tools lacked the necessary automation to manage Tyler's unique use case effectively.Â

âWe originally used CloudWatch alarms to power down non-production infrastructure outside of working hours,â explains Chris Camire, Senior Manager of Technical Services at Tyler. âBut this wasn't a long-term solution. Our clients often work late hours and need their non-production environments available on demand.â

The flip side of this challenge was that some cloud environments weren't used daily, or even weekly, but could be needed on a moment's notice.Â

âAs our client base grew, we continued to build additional pods to meet demand so that clients had secure, reliable environments to test in,â Camire says. âWhat we didn't do was create a strategy for how we organized our clients across that cloud infrastructure. We had clients from different time zones, of all sizes, and with varying use patterns all sharing the same cloud deployments.âÂ

Solution: Cloud AutoStopping to improve visibility into and management of idle resources

The lack of manageability with existing tools led Tyler to explore more full-featured cloud cost management solutions, including Harness Cloud AutoStoppingâ¢, part of Harness Cloud Cost Management (CCM), for active management of idle cloud costs.Â

âCloud AutoStopping opened up new possibilities for cloud cost management,â Camire says. âWe saw how reorganizing our deployments by geography, function, and use patterns could unlock game-changing savings.â

Tyler created a framework for reorganizing the cloud for cost efficiency â in their case, reorganizing the entire cloud estate by client time zone and client activity.Â

âWe started with time zones,â Camire explains. âBecause our client base is public agencies with regionalized offices, we grouped clients that begin and end their workdays at roughly the same time.âÂ

When Cloud AutoStopping detects an idle state for all the clients in a group, it can automatically power down that instance until it detects activity again. âThe cloud instance can re-start quickly, so that our clients don't even know the difference,â Camire says.Â

Results: Annualized savings of \$1.2M after just six months

Organizing cloud instances around time zones was just the start. Using the Harness CloudAutostopping console, Camire and his colleagues gained new visibility into client usage â specifically, just how infrequently some clients were using their pod environments.Â

âWe identified clients that were idle for much longer periods, measured in weeks or even months, grouped those clients together, and set Cloud AutoStopping to shut them down accordingly,â Camire says.Â

Grouping clients together by time-based criteria has produced amazing results. âRight away we were saving \$15,000 to \$20,000 a month, and in only six months, we saw our first savings topping \$100,000 a month,â Camire says.Â

Following the initial successful results, Camire and his team are working on establishing 100% coverage for Cloud AutoStopping rules, planning to replicate the cost savings throughout the cloud estate. Says Camire, âWe'll start exploring other Harness CCM features more in depth, like Cloud Asset Governance and implementing recommendations.â

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/comparison-guide/concourseci-vs-drone>

Comparison Guide

Harness

Continuous Integration

VS

Concourse CI

Harness Continuous Integration vs Concourse CI

Continuous Integration

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

Concourse CI

Concourse CI is an open source platform. CLI is a command line tool that you need to use to set up a pipeline on Concourse.

â

2014

â

Concourse CI is categorized as:

Continuous Integration

â

What is the difference between Harness DevOps Tools Vs. Concourse CI?

Concourse CI vs Drone: DevOps Tools Comparison

Updated

November 30, 2023

- Open Source Version
- GitHub Stars
- Self-Service (Simple)
- No Scripting Required
- Container & Cloud-Native
- Traditional App Support
- GitOps (Pipelines as Code)
- Any Source Code Manager
- Containerized Pipelines
- Containerized Plugins
- Secrets Management
- Command Line Interface
- Scalability (Required Infra)
- Admin & Maintenance
- Total Cost of Ownership
- Pricing

Free & Paid

24800

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Vault/KMS/3rd

<yes><yes>

Lightweight

<yes><yes> **.25 FTE**

<yes><yes>

<yes><yes> **Per User**

Free

5500

<with><with>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Lightweight

<yes><yes> **.25 FTE**

<yes><yes>

<yes><yes> **Free**

Free & Paid

Free

24800

5500

<yes><yes>

<with><with>

<yes><yes>

Lightweight

Lightweight

<yes><yes> **.25 FTE**

<yes><yes> **.25 FTE**

<yes><yes>

<yes><yes>

<yes><yes> Per User

<yes><yes> Free

Open source vs. Open core:

Concourse was created by two engineers in 2014 after trying various CI tools that always fell short. They proceeded to open-source the project on GitHub, where it sits at 5.5k stars at the time of this writing, and is constantly being updated by the creators and contributors. Drone is also open-source. There is, however, also a paid version of Drone that provides access to enterprise support and more integrations and features. Additional features include secrets management options, autoscaling, custom plugins, and more.

Self-Service (Simple):

The good: Concourse CI is a very light, easy-to-start CI tool with an intuitive UI. Though, as with every tool, there is a learning curve, and documentation can be a little light when running into problems. It is container-based and their small selection of resources, which are basically the equivalent of a plugin, help extend the platform. It's also easy to scale up and down. The bad: building pipelines on Concourse CI is time-consuming, because you have to do every single step yourself (no templates â every step has to be configured manually). It also lacks features that other CI tools already have, such as flexibility (pipelines are a bit rigid and basic, poor conditional flow), thereâs not much information about past runs in the UI, plugins are quite limited, and there are shortcomings with docker if you employ multiple libraries. The ugly: workers are a constant source of frustration that sometimes don't work as intended. They worked so poorly that it is now being deprecated in favor of Prototypes (should be available in Q2). Lastly, at the time of this writing, Concourse CI has 750 open issues on GitHub â so there are definitely lots of kinks to work out. Drone is built upon three pillars that enable engineers to build and test code quickly and accurately: simple, scalable, self-service. Drone installs in under 5 minutes, scales on demand, and all plugins run in containers on their latest version. This means less person hours spent by engineers maintaining the tool, and more time on what matters: getting that code to artifact.

GitOps:

Concourse CI added GitOps capabilities a few months ago. They have a tutorial on how to implement GitOps for your pipelines. Drone comes with built-in GitOps functionality â and has since 2013!

Containerized Plugins:

A great feature of both Concourse CI and Drone is that everything is run in a container. A massive plus of containerized plugins is that the plugins are maintained to their latest version and create no dependency chains. They require no updating. Concourse CI offers roughly 80 plugins (called resources, as stated above) at the time of this writing. Drone offers 150 plugins, thereby dramatically increasing the extensibility of the tool.

Secrets Management:

Concourse CIâs default is to encrypt secrets. They also offer support for HashiCorp Vault, CredHub, and AWS Secret Manager. Drone offers encryption on its open-source version. Meanwhile, the enterprise version offers these alternatives: encrypted, native, or externally, through third-party providers such as AWS Secret Manager, Kubernetes Secrets, and HashiCorp Vault. No matter how you want your secrets to be handled, Drone can rise to the occasion.

Pricing:

Concourse CI has no pricing associated with it as it is open-source. Drone is also free and available for download. It also has an enterprise version that is extremely feature-rich, but does have pricing attached to it. To familiarize yourself with enterprise pricing, please contact sales.

**Please note: Our competitors, just like us, release updates to their products on a regular cadence. We keep these pages updated to the best of our ability, but there are bound to be discrepancies. For the most up-to-date information on competitor features, browsing the competitorâs new release pages and communities are your best bet.*

Try Harness For Free

Continuous Integration

Interested in seeing what's under the hood? Browse through the Harness Continuous Integration Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/sheerid-accelerates-deployments-by-35x-eliminates-burden-on-developers-by-using-harness>

SheerID Accelerates Deployments by 35X, Eliminates Burden

On Developers by Using Harness

Using Harness, SheerID accelerated deployment velocity, increased developer productivity allowing more time to focus on innovation and successfully implemented canary releases during its transition to a continuous deployment cycle.

âHarness Impact

- Accelerated deployments by 35X, from 1 per week to 7 per day
- Cut deployment effort by 35-40 hours per week

About SheerID

SheerID collaborates with leading B2C brands in retail, media, telco, consumer apps, travel and hospitality and software. Their solution enables brands to instantly verify eligible consumers in high-value communities like students, teachers and the military for gated, personalized offers. SheerID has conducted over 135 million verifications for more than 300 brands as of April 2023, preventing several billion dollars in fraud within the last year.Â

â

Challenge: Even With Kubernetes, Deployments Were Too Complex

SheerID is a fast-moving technology company, constantly looking for ways to move even faster. However, their development process faced a bottleneck due to a monolithic application that limited their team to bi-monthly deployments. Deployments involved a manual process conducted by a central release team, requiring nearly 17 hours of preparation. Furthermore, scaling up was challenging, as spinning up an Amazon Elastic Compute Cloud (Amazon EC2) instance took approximately 15 minutes. This slowed growth and hindered the company's ability to expand into new regions and industries. Additionally, this hampered a move to support a multi-cloud deployment strategy across both Amazon Web Services (AWS) and Google Cloud Platform (GCP).

â

â

Creating a continuous deployment process was the ultimate goal, but SheerID had to move in phases to get there. They started by breaking the monolithic application into services and utilizing Jenkins and Kubernetes for their services deployments. This allowed SheerID to successfully transition from a centralized release team to a self-service model owned by their development teams. The shift empowered their teams to take ownership of the deployment process and reduced the deployment time, enabling SheerID to push releases weekly.

â

âWe had four development teams and an additional ops team releasing code and microservices using several Jenkins instances. While it helped to reduce the deployment time, it also created its own challenges with usability, complexity, instability, secrets management, and more. Developers had to log in to many Jenkins instances for their deployments, occasionally missing environments in deployment targets, resulting in inconsistencies. There was too much overhead and it was eating into developers' time,â Ellis added. He knew they had to do more to take time and effort out of deployments so that developers could focus on features.

â

â

Solution: Pipeline Automation, Canaries, and Visibility, All In OneÂ

SheerID started looking for continuous deployment solutions. The team explored CircleCI as an option, but found it came with a substantial cost. After a brief demo of the Harness platform, SheerID was convinced of the value of investing in a software delivery platform.

â

â

Ellis elaborated on the capabilities that have accelerated deployments at SheerID:Â

Within six months, SheerID was automating deployments using the Harness platform.Â

â

Results: Continuous Deployment With Harness

By using Harness to automate deployments and eliminate most of the Jenkins burden from developers, SheerID has drastically reduced deployment times while also drastically increasing deployment frequency by 35X.Â

â

â

âI think back to how it was before Harness,â Ellis explained. âDevelopers had to log into a bunch of Jenkins, click through, run tests, check logs every day. That eats up a lot of time. Harness gave a lot of time back to our teams. Itâs like we hired an entire developer without hiring an entire developer.â

â

Harness is also saving time and effort for SheerID with automated canary releases. This increased the teamâs confidence in deployments, so if a problem arises, developers simply roll it back and flag it for the engineering team.

â

SheerID has realized time savings and improved deployment quality using the Harness platform. In combination with a multi-cloud deployment strategy, SheerIDâs teams can focus on what matters mostâ deploying new features quickly with confidence and expanding to new regions and industries.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform^{Â®}

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/skillsoft-harness-cd-reduce-frustrations-workloads-increasing-velocity>

Skillsoft Moves to Harness CD to Reduce Frustrations and Workloads While Increasing Velocity

As Skillsoftâs software platform expanded to more than 100 microservices, the engineering team looked to Harness to standardize deployments, enable scalability.

- Minimized operations teamâs deployment workload
- 200 developers across 17 teams running seamless deployments
- Reduced time to deploy by over 50%, deployments went from taking over 3 hours to under 45 minutes for 100+ microservices
- Ended weekend deployments and accompanying team frustration by gaining the ability to deploy during business hours with 0 downtime

About

Skillsoft (NYSE: SKIL), a leading platform for transformative learning experiences partners with enterprise organizations and serves a global community of learners to prepare today's employees for tomorrow's economy. Skillsoft customers gain access to multimodal learning experiences that build skills and grow a more capable, adaptive, and engaged workforce.

Challenges

Skillsoft nurtured an entrepreneurial environment within its business that proved beneficial as the company grew. But, as its software platform expanded to include about 100 microservices, each team of engineers created a unique delivery process to fit their own needs â adding complexity that made standardization more difficult at scale. The operations team would then take the code and manage the deployments. Many manual steps and inconsistencies invited errors, which then added delays for reviews. In some cases, engineers would hand-deliver configuration changes between environments.

âIt was not very efficient, and we couldn't get the economies of scale we wanted,â recalled Anil DâSilva, Senior Director of DevOps at Skillsoft. âWe had to standardize in a way that let engineering deploy from their development environment right into production with as little friction as possible. We had just moved to AWS as our production environment, so we wanted to be on the cutting-edge for deploying software as fast and as safely as we could.â

Skillsoft wanted delivery and deployment standardization, self-service, and speed with two-week lead times and 90-minute deployments as the goals to help accelerate time-to-market for new features and products. The company was also seeking certification for FedRAMP, the U.S. government program for adoption of cloud services, so deployment consistency and compliance were critical.

âWe couldn't continue delivering all microservices at the same time,â added DâSilva. âThat's not continuous delivery (CD). So we stepped back and asked, âwhy can't we just have engineering be responsible for the complete deployment cycle?ââ

Adding to the urgency was team burnout and frustration due to deployments occurring every other Saturday. That cadence forced over 40 people across operations and engineering to work or be on-call during weekends along with at least three hours of downtime that also impacted customers. Skillsoft finally made the decision to move away from its manual, homegrown, Jenkins-based deployment process.

Solution

The challenges, frustrations, and friction from its inconsistent deployment processes became a high-visibility challenge that pushed Skillsoft to find a CD solution. DâSilva evaluated Argo CD and Spinnaker before selecting Harness.Â

âChoosing Harness gave us a lot of potential, but we first had to fix our process,â said DâSilva. âWe wanted Harness to make our lives easier. We started from scratch with a standardized YAML that we turned into Helm templates in order to make the deployment process self-service and repeatable. We took our time so we could take full advantage of Harness.â

Skillsoft's engineering team is arranged into 17 squads, each contributing microservices to the platform. With Harness, squads now run self-service deployments, taking ownership of the entire process and nearly eliminating deployment efforts for operations. Creating common templates in Harness also helped Skillsoft use a common secrets management solution, which was critical for its FedRAMP certification and other initiatives.Â

âWe barely had to train anyone on Harness,â DâSilva said. âThere's a big green flag for successful deployments, and if something goes wrong, they can just read the Harness log. Next, we plan to implement quality gates in Harness to ensure services hit their service level agreements (SLAs).â

âResults

With the standardized, frictionless process for development teams to automate otherwise manual work with Harness CD, Skillsoft now has the deployment velocity and consistency it long desired and has given Engineering complete ownership of the delivery process. That adds up to a lot of time saved for Skillsoft's 17 teams and 200 developers.

âWith Harness, we can rollout changes to 100 microservices in less than 45 minutes â that used to take more than three hours,â added DâSilva. âWe don't even think about it anymore. Engineering can do it by themselves, too, eliminating the effort operations used to take on. And, operations was happy to give up that tedious work. Harness is saving us a huge amount of time.â

Using Harness to standardize CD has also given Skillsoft the confidence to begin moving deployments to business hours rather than weekends. Harness also provides more visibility into metrics and trends as Skillsoft works to hit broader deployment goals.Â

âNow, anyone can deploy 24/7 and I don't have to be awake 24/7,â DâSilva concluded. âHarness is simple. I don't want to be worried about individual deployments going out. And, as we continue to meet the goals for what a CI/CD pipeline should do for Skillsoft, Harness is a big part of that progress.â

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/deluxe-corporation-modernizes-deployments-automates-devsecops-processes>

Deluxe Corporation Modernizes Deployments and Automates DevSecOps Processes with Harness

When Deluxe was struggling with reduced time-to-market for new features, they turned to harness to automate CI/CD processes and shift application security left.

Harness Impact[®]

- Lowered overhead costs associated with building and maintaining their continuous integration/continuous development (CI/CD) pipelines by 72% for modern applications.
- Decreased new pipeline build times from 2 weeks to a few hours for modern applications by reusing the templates and connections.
- Reduced time to onboard deployments from 2 months to a couple of days for modern applications on already provisioned environments.[®]
- Decreased code security issues noise by 95% with STO.

About Deluxe Corporation

Deluxe is a leading provider of payments and technology products, services, and platform solutions for businesses. Its solutions help businesses pay, get paid, and grow through a powerful scale that supports millions of small businesses, thousands of vital financial institutions, and hundreds of the worldâs largest consumer brands, while processing approximately \$3 trillion in annual payment volume. The company has grown over the years through multiple acquisitions, resulting in a wide and deep tech stack that includes multiple development languages, frameworks, and databases.

Challenge: Reduced Time-To-Market for New Features

Deluxe grappled with the integration and standardization of technology and roles across its portfolio of on-premises and cloud-based products and services. The company also experienced slow quarterly releases, which resulted in delays and missed deadlines. They had no standardization or templatization for deployment pipelines, and they relied on manual processes and heavy scripting, resulting in slowed-down processes. Furthermore, they needed greater compliance, security, and governance.

Pankaj Gupta, Executive Director of Product and Software Architecture, observed that âthere were no consistent or single architectural patterns or shareable technology patterns. Additionally, there were various technologies, and everyone was continuously changing them without thinking about a unified cloud strategy. This all hindered our go-to-market strategy.â

The gap created by these challenges manifested as reduced time-to-market for new features, impacting customer acquisition and retention.[®]

Solution: Automating CI/CD Processes and Shifting Application Security Left

To solve these challenges, Deluxe decided to standardize on cloud and started looking for a solution that would offer the flexibility and capabilities needed to achieve its goals. Harness was the solution that provided the company with the required automation capabilities to speed up the go-to-market strategy for its products and functionalities.

Moreover, Deluxe recognized the need to modernize its manual Jenkins environment as a key initiative to reduce time-to-market for new features. According to Andrew MacLean, Director of Software Delivery and DevOps, "When your customer depends on you for a solution, you better make sure it's there and it works."

The current toolset required an automated solution for its CI/CD process via an innovative and modern approach from Harness. This would enable a higher degree of automation and facilitate more frequent deployments. Furthermore, they could make sure of greater application security by orchestrating throughout the CI/CD process and much earlier in the software development lifecycle (SDLC) via the Harness Security Testing Orchestration (STO) module.

Results: Building Pipelines Faster, Reducing Security Scanner Noise by 95%

Harness significantly reduced the time spent on deployment activities — lowering overhead costs associated with building and maintaining their CI/CD pipelines by 72% for modern applications.

Already, Deluxe is seeing pipeline onboarding time decrease significantly, from two weeks to a few hours for modern applications by reusing the templates and connections. As Deluxe templateizes things further, the goal is to reduce it even more. In addition, they're seeing time to onboard deployments decrease from two months to a couple of days for modern applications on already provisioned environments. MacLean commented, "We are building new pipelines in less than two hours and onboarding deployment in a few days. Harness has enabled our developers to deploy safely and faster than ever."

Leveraging Harness STO for one pipeline alone, the team at Deluxe found 170 issues from one scanning vendor and deduplicated them down to just nine prioritized issues. Gupta remarked, "Harness STO automated the deduplication of outputs from a security scanning vendor, reducing the noise for my team by 95%. Now, we can focus on the top issues and tackle them quickly."

Overall, Deluxe could deliver features faster, improve customer retention, and enhance the developer experience with Harness. They also created a collaborative DevSecOps culture, as well as improved security, compliance, and governance. Gupta confirmed this, saying, "Harness not only gave us the ability to automate our CI/CD pipelines but also brought consistency and standardization across all applications and teams."

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/comparison-guide/jenkinsx-vs-drone>

Comparison Guide

Harness

Continuous Integration

VS

Jenkins X

Harness Continuous Integration vs Jenkins X

Continuous Integration

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

Jenkins X

Jenkins X provides pipeline automation, built-in GitOps, and preview environments to help teams collaborate and accelerate their software delivery at any scale.

1-50

2018

â

Jenkins X is categorized as:

Continuous Integration

â

What is the difference between Harness DevOps Tools Vs. Jenkins X?

Jenkins X vs Drone: DevOps Tools Comparison

Updated

November 30, 2023

- **Open Source Version**
- **GitHub Stars**
- **Self-Service (Simple)**
- **No Scripting Required**
- **Container & Cloud-Native**
- **Traditional App Support**
- **GitOps (Pipelines as Code)**
- **Any Source Code Manager**
- **Containerized Pipelines**
- **Containerized Plugins**
- **Secrets Management**
- **Command Line Interface**
- **Scalability (Required Infra)**
- **Admin & Maintenance**
- **Total Cost of Ownership**
- **Pricing**

Free & Paid

24800

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Vault/KMS/3rd

<yes><yes>

Lightweight

<yes><yes> **.25 FTE**

<yes><yes>

<yes><yes> **Per User**

Free

3500

<with><with>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<with><with>

<yes><yes>

Lightweight

<yes><yes> **.25 FTE**

<yes><yes>

<yes><yes> **Free**

Free & Paid

Free

24800

3500

<yes><yes>

<with><with>

<yes><yes>
<no><no>

Vault/KMS/3rd

<with><with>
<yes><yes>
<yes><yes>

Lightweight

Lightweight

<yes><yes> .25 FTE

<yes><yes> .25 FTE

<yes><yes>

<yes><yes>

<yes><yes> Per User

<yes><yes> Free

Open source vs. Open core:

Jenkins X is an open-source solution, and as such, only offers an on-prem version. Drone is also open-source. There is, however, also a paid version of Drone that provides access to enterprise support and more integrations and features. Additional features include secrets management options, autoscaling, custom plugins, and more.

Self-Service (Simple):

Jenkins X was created as a cloud-native version of Jenkins, and with that comes Kubernetes , pipeline automation, GitOps, Terraform, secrets management, and more. Some may wonder what the benefits of Kubernetes are, and weâd point you to this: not only can Jenkins X run and scale on Kubernetes itself, it can help you deploy your applications to Kubernetes with a relatively simple setup. Each job is typically run in its own container, run ad-hoc, so it will get placed anywhere within the cluster as youâve described in your node affinities. Underlying kubelets can scale horizontally if you have Cluster Autoscaler or something similar thatâs augmenting your clusterâs elasticity. All this to say, compared to the OG Jenkins, this is a treasure trove of modern features. Less toil, fewer setup issues, no scripting. Fewer person hours on maintenance. Fewer plugins. Easy scaling. But â letâs switch the focus from Jenkins X vs. Jenkins, and look at what weâre all really after: Jenkins X vs. Drone. From a general standpoint, Jenkins X could use further improvement. The toolâs UI is complex, it doesnât offer good error messaging (ie: why did my build fail?), and lacks integrations. Jenkins X has only a few plugins to its name as well, most of them in this GitHub repo, and while admin and maintenance time/effort is much smaller than on Jenkins (.25 FTEs vs. 2-5 FTEs), itâs still not a completely simple platform. Drone is built upon three pillars that enable engineers to build and test code quickly and accurately: simple, scalable, self-service. Drone installs in under 5 minutes, scales on demand, and all plugins run in containers on their latest version. This means less person hours spent by engineers maintaining the tool, and more time on what matters: getting that code to artifact.

Containerized Plugins:

A great feature of Drone is that everything is run in a container. A massive plus of containerized plugins is that the plugins are maintained to their latest version and create no dependency chains. They require no updating. Drone offers 150 plugins, thereby dramatically increasing the extensibility of the tool. Jenkins X, however, still involves managing dependency versions manually. That is to say, it does not pull dependencies from immutable and versioned container images like Drone. This gives Drone an edge by reducing the maintenance overhead upon the DevOps engineers managing the system.

Secrets Management:

Jenkins X does not offer native secrets management capabilities, but it comes pretty close. It leverages Kubernetes External Secrets, an open-source tool created by GoDaddy engineers, which integrates with third-party secret management tools, such as HashiCorp Vault, AWS Secrets Manager, Azure Key Vault, and Google Secret Manager. Drone offers encryption in its open-source version. Meanwhile, the enterprise version offers these alternatives: encrypted, native, or externally, through third-party providers such as HashiCorp Vault, AWS Secret Manager, and Kubernetes Secrets. No matter how you want your secrets to be handled, Drone can rise to the occasion.

Pricing:

Jenkins X has no pricing associated with it as it is open-source. Drone is also free and available for download. It also has an enterprise version that is extremely feature-rich, but does have pricing attached to it. To familiarize yourself with enterprise pricing, please contact sales.

**Please note: Our competitors, just like us, release updates to their products on a regular cadence. We keep these pages updated to the best of our ability, but there are bound to be discrepancies. For the most up-to-date information on competitor features, browsing the competitor's new release pages and communities are your best bet.*

Try Harness For Free

Continuous Integration

Interested in seeing what's under the hood? Browse through the Harness Continuous Integration Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/zeroflucs-reduces-infrastructure-cost-feature-flag-optimization>

ZeroFlucs Reduces Infrastructure Cost by 60% with Feature Flag-Triggered Optimization

ZeroFlucs increased efficiency and reliability for customers with Harness Feature Flags.

- Over 1,400 releases per month; 10+ releases per day per developer
- 98.8% of deployments are successful - not requiring a rollback
- 3-minute average build and deploy pipeline including testing, benchmarking, and Kubernetes rollout of over 100 services
- 50-60% drop in infrastructure cost within 1 week

About ZeroFlucs

ZeroFlucs, a startup located in Brisbane, Queensland, provides technology that enables bookmakers and wagering service providers to offer same-game betting. In a highly price-sensitive, risk-averse, and regulated market, the ZeroFlucs technology is enabling customers to double their margins. By running low-cost simulations with ZeroFlucs, customers are able to quantify the probability of a bet succeeding and use it to set the price for the transaction.

Challenge: Minimizing Cost to Customers to Survive in a Price-Sensitive Industry

Steve Gray, CEO and Founder of ZeroFlucs, knew that keeping the time taken to generate predictions low was critically important to his business â both to maximize profits for bookmakers, as well as provide engaging products for end users. âNumbers are our game. We use numbers in everything we do,â Gray said. The more simulations performed of a given match, the more accurate the pricing offered to customers becomes, but with each simulation, comes a cost. The challenge then was to minimize the cost while maximizing the accuracy of the analysis.

In his own words, automation and efficiency became the focus of the company and its culture right from the start. He wanted to âcome out strong out of the gate with automation and a continuous integration and delivery (CI/CD) culture. All the successful teams I ever

worked in had that culture first. They've been able to ship software continuously, reliably, and safely.

The more the ZeroFlucs team could focus on being automation-first, the better they'd be able to meet their customer needs. The amount of processing required in their system directly impacts the cost their customers pay, especially when low bet simulation costs were a primary value proposition. In a price-sensitive and risk-averse market, it was important to show a demonstrable improvement in margins to get customers to even look at the product of such a young company.

Driving the cost per simulation as low as possible was the top priority for ZeroFlucs. According to Gray, "coming in as a new product that's more expensive is a death knell in our industry. Cost and margins are absolutely the biggest things."

Solution: Leveraging Feature Flags to Run Betting Simulations as Efficiently as Possible

The first step was to build a massively parallel system that could run full-match simulations as quickly as possible. However, the team had yet to analyze the optimum configuration that would minimize the cost of running simulations, which would in turn enable them to run more simulations and boost the accuracy of the probabilities they output. What backend configuration would make the most sense to minimize cost and maximize performance?

As it would turn out, the answer would be feature flags. Gray and his team had 12 configuration points such as the balance between storage and RAM usage and the number of servers to use to compute in parallel and about four options at each configuration point, netting out to over 16 million possible permutations of configuration.

Initially, the team was testing these configurations manually. And for each configuration, the application had to be redeployed, which took 5-10 minutes each time. Even with an automated script, this would have taken the ZeroFlucs team months to test every alternative in sequence. This extended time would directly impact their ability to provide value to their customers and maintain the viability of the company.

Gray then had the idea to leverage feature flags. Instead of sequentially deploying and testing one permutation at a time, they could put each of the configurations behind a feature flag and write a script that would test all those permutations live while in production.

The team was already familiar with LaunchDarkly, but it was cost-prohibitive for the sports betting market and also took away from the value ZeroFlucs offered to the market. When they found out that Harness, which they already used for CI/CD, offered the Feature Flags solution, it was a no-brainer to work with Harness again.

"Feature Flags was a very easy and economical way to continue leveraging that investment with Harness," Gray said. "It's part of a holistic solution."

Results: Passing on 50-60% Reduction in Costs to the Customer

Steve and the team were already leveraging Harness CI/CD for their deployments. After using Drone until it could not be scaled further, ZeroFlucs turned to Harness CI/CD. With 30 services, the effort associated with the upkeep of the scripts and pipelines took resources away from building the software. So, the team upgraded from Drone to Harness CI/CD to build and streamline those templates, resulting in:

- Over 100 services that are maintained by 10 deployment templates
- Over 1,400 releases per month
- 98.8% of deployments are successful - not requiring a rollback
- Three-minute average build and deploy pipeline including testing, benchmarking, and Kubernetes deployment of the new versions
- 2x the number of services in less than 9 months

With their success using Harness, it only made sense to extend their use of the Harness platform and implement Feature Flags. After all, this was a mission-critical part of the puzzle to minimize cost and maximize performance.

Once Feature Flags was implemented and the necessary configurations were set, it became effortless for ZeroFlucs to test large numbers of permutations and find the most optimal configuration between cost and performance. Since implementing feature flags, ZeroFlucs has optimized their system for where they're at, and they're confident they can continue to optimize as they scale using the same feature flags. Today, they're evaluating flags over one billion times per day.

Each developer was already releasing more than 10 times a day with Harness CI/CD, and when they expanded into Harness Feature Flags, ZeroFlucs gained the following benefits:

- From 5-10 mins per configuration test to 5-10 seconds
- Optimized their infrastructure within 1 week instead of months
- 50-60% drop in infrastructure cost
- 50,000 simulations run per second due to the reduced cost profile
- No performance impact - running over 1 billion flag evaluations per day
- 30-50% price advantage over competition

"With Harness, we have decreased infrastructure cost by 60% with no performance impact as we run over 1 billion flag evaluations per day. This allows us to maintain a price advantage over competitors!" - Steve Gray, CEO and Founder, ZeroFlucs

According to Gray, they've only barely scratched the surface of what they can do with Harness. In fact, Gray and his team continue to use Harness Feature Flags in creative ways. As one example, the ZeroFlucs team was struggling to show off the intelligence in their system in a live demo to prospective customers because the technology worked so fast â normally, that's a good thing. By strategically placing feature flags at key inflection points in the product, they were able to artificially slow down their demo so that prospects could see the technology at work and be properly wowed by what they were seeing.

By leveraging Harness CI/CD and Harness Feature Flags, ZeroFlucs has created a cost and performance-efficient system that will continue to be competitive as the business continues to mature. In a price-sensitive industry, being able to beat out the competition by 30-50% in price is sure to drive the growth ZeroFlucs is looking for.Â

âWe have never gotten an objection on price,â says Steve.

The Race for First Place

ZeroFlucs is just getting started leveraging Harness. In the near future, they're looking to implement Feature Flags in more traditional ways, including using it to increase the velocity of their software delivery process.Â

Gray's vision is âto get human hands out of the release process as much as possible. Ideally, we want it all to go through a predefined and automated process.â With this automation-first culture, ZeroFlucs wants to leverage Harness to automate the rollout of new features and changes to their algorithm. They want to be able to create a release template that will ensure any changes meet their strict release criteria.Â

While a lot of the work to date has been focused on optimisation of raw costs, another key area is managing the risks associated with changing a live, customer-facing system. By using Harness Feature Flags to incrementally roll out changes to users and evaluate performance versus the current solution, as ZeroFlucs evolves its models and algorithms they'll be able to minimize the risk of regressions and ensure that customers get a consistent, profitable experience.

With Harness CI/CD speeding up and automating deployments and Harness Feature Flags optimizing their infrastructure, Gray and ZeroFlucs are turning their eyes to customer acquisition and how to leverage more of the Harness platform. By continuing to automate and create efficiencies, ZeroFlucs is ready to take on the sports betting market and completely disrupt the status quo.

âThe biggest benefit of Harness is that essentially everything is automated and it's simple,â said Gray. âIt's a cohesive package of tools, and all of it is usable, available, and approachable. My 12-year-old can use Harness to trigger a build pipeline!â

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Ulta Beauty Speeds Time-to-Market By Easing and Accelerating Deployments

Ulta Beauty sought to improve guest experience by developing its digital store of the future. Harness CD helped achieve this, increasing deployment volumes 50x.

- Faster time-to-market for key project
- 50x increase in deployment volumes
- Time saved for developers and DevOps teams

Ulta Beauty is the largest U.S. beauty retailer and the premier beauty destination for cosmetics, fragrance, skin care, hair care products, and salon services. The company offers a complete beauty experience with more than 25,000 products from over 600 well-established and emerging beauty brands sold online and at its 1,300+ stores across all 50 states. Ulta Beauty also has more than 39 million loyalty members and over 40,000 employees, helping drive nearly \$9 billion in annual sales.Â

Challenges

Ulta Beauty wanted to improve its guest experience by developing its digital store of the future to stand out among the crowd and better cater to todayâs growing consumer demands. This was a fundamental shift for the company as building the new experience with speed and reliability in mind involved moving its ecommerce platform to Google Cloud from an on-premise data center. However, friction across the delivery process began to impede the transition and grow the scope for developers.

âContinuous delivery (CD) was hugely important to us,â recalled Michael Alderson, Principal Cloud Architect at Ulta Beauty. âOur goal was to get out of the way of our developers and take work off their plates. We needed to build a delivery process that was both seamless and transparent for them.â

Ulta Beauty also quickly outgrew its existing delivery tool. Alderson pointed to maintenance and troubleshooting processes that were âless than idealâ and other issues that could occur when the tool was left to run for extended periods of time. It all hastened the companyâs decision to make a change.

âWe want to spend time using the business rules, not maintaining the platform,â said Alderson. âThis was our opportunity to empower developers to spin up or down a new environment without worrying about the details.â

When exploring new CD solutions, Ulta Beauty knew they needed a tool that would allow them to continue using its Helm charts to manage Kubernetes, employ environment variables to port CI data into its CD process, and integrate tightly with tools already in place like Jira.Â

Solution

Realizing its current deployment tooling couldnât support its needs, Ulta Beauty considered new solutions for CD and, ultimately, landed on Harness Continuous Delivery & GitOps as the best choice.

âOnce we saw Harness, it was immediately evident this was the direction we needed to go in,â said Alderson. âHarness was SaaS-driven, had more security than we could deploy in-house, and provided transparency for troubleshooting. It was also easy to integrate with Jira, so we could pass information to developers or deploy an app to different landscapes just by changing the ticket status.â

Using Harness, Ulta Beauty organized its Helm charts and created reusable variables to templatize the information throughout its CI/CD pipeline, making deployments more repeatable and less manual in nature. Combined with the Jira integration and streamlining more of the overall delivery process, Ulta Beauty was able to increase deployment velocity with Harness.Â

âHarness lets developers spin up a new environment in just a few minutes,â added Alderson. âThey donât have to worry about the inner workings of the deployment. Itâs empowering them to do what they need to do without any hurdles.âÂ

Harness helped Ulta Beauty overcome its key challenges â troubleshooting, maintenance, organization, and integration â while expediting the delivery process for developers. Better yet, the company could do it all with a Harness rollout that took âmaybe an hour,â according to Alderson.

âWeâve just found benefit after benefit with Harness,â Alderson said. âIn under a month, we were in full production, able to do canaries, get ephemeral environments spun up, organize and optimize our Helm templates, and connect with Jira. Every strength Harness has lined up with what we needed.â

Results

Ulta Beauty has now expanded Harness to CD processes beyond just its digital store of the future to seven more projects across multiple teams. Developers can deploy faster with more confidence and consistency, without getting bogged down by repetitive tasks. When an issue does arise, Harness provides clear visibility to uncover the root cause. For DevOps teams, the ability to provision infrastructure on-demand, verify deployment health automatically with Continuous Verification, automate rollbacks, and standardize around pipeline templates with Harness saved countless hours of work. Enforcing pipeline governance policies and relying on audit trails to keep compliant also played an integral role in scaling their CD processes effectively.Â

âTime-to-market for our eCommerce platform was a huge benefit of Harness,â Alderson said. âWe saved months of time. Months. Harness really came through in a big way for that project.â

Faster, more streamlined processes with Harness have also drastically increased the total volume of deployments. When developers used to rely on local sandbox environments before Harness, Alderson estimated Ulta Beauty would deploy only about 100 environments per month. Now, itâs well over 5,000 per month with over 15,000 instances deployed at any given time. According to Alderson, âHarness helped us increase our deployment volumes by 50 times.â

âWeâre more agile and can quickly give developers the environments they need,â said Alderson. âTeams are on a two-week delivery cycle and eventually want to move to daily. Harness will allow us to get there.â

Alderson added that the speed enabled by Harness extends across projects and teams, saying it saves âtons of timeâ for DevOps and developers alike. People can use that time for more strategic tasks, key projects, or simply logging off in time for dinner.

âThe big benefit of Harness comes down to one thing: quality of life,â concluded Alderson.â

To learn more about Harness Continuous Delivery & GitOps, visit: <https://www.harness.io/products/continuous-delivery>

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/synopsys-boosts-cloud-visibility-cost-control-harness-cloud-cost-management>

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

About

Synopsys, Inc. (Nasdaq: SNPS) is the Silicon to Software[®] partner for innovative companies developing the electronic products and software applications we rely on every day. As an S&P 500 company, Synopsys has a long history of being a global leader in electronic design automation (EDA) and semiconductor IP and offers the industry's broadest portfolio of application security testing tools and services. Whether you're a system-on-chip (SoC) designer creating advanced semiconductors, or a software developer writing more secure, high-quality code, Synopsys has the solutions needed to deliver innovative products. Learn more at www.synopsys.com.

Challenge: Increasing Cloud Infrastructure to Support Growth

Over the years, Synopsys has broadened its technology portfolio to serve customers across a variety of markets. As it continued to scale product lines through mergers and acquisitions (M&A), the company was also increasing its cloud infrastructure to support that growth, specifically with Google Cloud Platform (GCP) and Amazon Web Service (AWS).⁶

M&A expansion also resulted in a drastic cloud spend increase, amplifying an ongoing issue at Synopsys: lack of visibility into additional cloud infrastructure resourcing, particularly with Kubernetes clusters and workloads. In addition, teams from these acquisitions had their own tools and ways of managing and visualizing cloud costs, resulting in inconsistent data measurement across organizations.⁷

Synopsys tried resolving these challenges in the past through a traditional cloud cost management product. The results were mixed, and didn't provide a comprehensive solution that the team needed. The prior product was designed for FinOps teams, offering some visibility for cloud resources and recommendations for cloud-native services through open-source tools. However, it lacked granular recommendations for cost transparency and optimization that the CloudOps team needed.

The ideal solution for Synopsys would unify cloud management of the entire application lifecycle – from deployment to cost optimization. This would result in their CloudOps team managing resources efficiently, and their FinOps team achieving cost control. It would also make it easier for developers to act on cost recommendations with added context that was historically missing from cloud cost tools.

Solution: Simplified Cloud Cost Visibility and Optimization at Scale

Synopsys chose the Harness Software Delivery Platform to solve this challenge, leveraging Harness Cloud Cost Management (CCM). It was the only single, integrated solution that met the needs of both the CloudOps and FinOps teams, delivering performance efficiency for engineering and minimizing ongoing cloud costs for FinOps goals.

"The big advantage was that we could enable developers to deploy the applications and manage costs from the same tool," said Jim D'Agostino, Senior DevOps Engineer, Synopsys. This ultimately resulted in costs being more proactively managed as part of the software delivery process instead of being treated as an afterthought optimization that compounded cost inefficiencies.

In addition to the technical features that made significant improvements, Synopsys also valued the Harness customer success and engineering teams' support, helping to customize views and recommendations for their business requirements. As Colleen Kelleher, IT Manager, Synopsys put it, "Having a cloud cost management tool and a support team that is aligned with our business goals is key to the success of all our teams here at Synopsys."⁸

Results: Increased Cloud Visibility and Cost Control

Synopsys leverages three key capabilities of Harness CCM to simplify their visibility needs and inform decision-making.⁹

1. Custom Business Intelligence (BI) Dashboards

The BI dashboards in Harness Cloud Cost Management offer customizable views that eliminate the effort needed for finance users and engineers to understand resource inventory as well as track cost trends across their multi-cloud architecture. Harness Cloud Cost Management can also set and send alerts when user-defined thresholds are exceeded.

The flexibility of these BI dashboards allows organizations to personalize the experience and provide clear visibility into what's most important to them. For example, Synopsys customized dashboards to find and investigate suspicious activity with cloud resources. They're able to address issues as they occur, resolving them quickly and minimizing their impact on budgets.¹⁰

2. Kubernetes Cost Visibility and Optimization¹¹

Because the developers have direct access to Harness Cloud Cost Management, the team enabled one of its unique features: deep visibility and granular recommendations into Kubernetes workloads and clusters.

Developers can view costs by workloads, clusters, namespaces, or pods. They can also tailor recommendation calculations to include or exclude certain instance types, add CPU/Memory buffers, or even specify the minimum CPU/memory requirements. Metric granularity into workloads and clusters levels is crucial for developers to make the right decisions for their applications, instead of relying on generic recommendations.¹²

3. Cost Anomaly Detection Across Cloud Providers

Synopsys uses the Cost Anomaly feature in Harness Cloud Cost Management to identify resources across GCP and AWS that show cost spikes deviating from the usual spend – all in one place. This helps them track potential waste and unexpected charges on a regular basis, alerting affected teams so they can immediately take action, avoiding the usual cost overrun fire drill at the end of the month.¹³

The team at Synopsys uses this capability extensively with the budgeting and forecasting feature. Harness Cloud Cost Management shows their spend as the month goes on, and it projects what costs will be at the end of the period. Now, the team can proactively manage budgets across cloud providers and keep them under control.

Next Up: More Cloud Cost Reductions

As Synopsys continues to count on Harness for visibility and to optimize their cloud accounts, the IT and CloudOps teams plan to automate idle resource management with Harness Cloud Cost Management's Intelligent Cloud AutoStopping. This feature automatically stops/starts idle resources, which can reduce costs by up to 75% across their pre-production environments.

Synopsys set out to take on two major challenges with their growing portfolio of company acquisitions: manage the high cost of cloud infrastructure expansion and unify the different cloud cost data measurement tools across companies. With key Harness Cloud Cost Management features and customized BI dashboards, the Synopsys CloudOps and FinOps teams now gain unmatched visibility and cost control for cloud management. That's a win-win for Synopsys and the new companies acquired through M&A.

Harness Cloud Cost Management offers best-in-class cost visibility, recommendations, and automation for both FinOps and DevOps users. Request a demo to see the savings your team can achieve.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

Advanced Achieves Cloud Cost Governance Excellence, Saves 33% on Cloud Costs with Harness CCM

Advanced was struggling with cost overruns, and lack of adoption of the previous cost management tool. After implementing Harness Cloud Cost Management, Advanced has seen 33% annualized cost savings to-date.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/ancestry-consistency-governance-downtime-systems-onboarding-cicd>

Ancestry Adds Consistency and Governance to Cut Downtime and Systems Onboarding Effort

To hit 99.9% uptime, Ancestry's deployment processes had to change. Find out how Harness helped build a robust, consistent, and governed CI/CD process.

Harness Impact

- Centralized governance to drive deployment consistency and best practices
- 78% reduction in systems onboarding effort
- 50% decrease in deployment caused production outages

About

For more than 30 years, Ancestry has been using cutting-edge technologies, ongoing innovation in DNA science, digitizing historical records, and automated research to help customers create a more complete family portrait. It has invested over \$300 million to build the world's largest collection of family history records and continuously launch new products and services for journeys of personal discovery. Now, with over 10,000 terabytes of data, \$1 billion in revenue, and 1,400 employees, Ancestry has become a global leader in family history and consumer genomics.

Challenges

Ancestry built its massive business by helping people uncover information about their family histories. But running such a popular online business at scale for more than three million paying subscribers and over 125 million family trees across a growing portfolio of brands, products, and services requires an equally massive development effort. As Ancestry ramped up its development team with new hires while maintaining 30-plus years of development history, its deployment processes became inconsistent and difficult to govern. That put undue strain on the company's architecture team, which was charged with standardizing and adding best practices to its deployment processes.Â

âWe've been growing our engineering base, onboarding new developers, and creating new teams for new products,â explained Ken Angell, Principal Architect at Ancestry. âAncestry has also been around for a long time and our development processes were left up to the developers. With everyone focused on new products, we didn't have the time to guide teams on processes and best practices.âÂ

With most of its developers already familiar with Jenkins, Ancestry managed delivery and integration using Jenkins as a CI solution extended to CD. But every team had a different process for building and shipping code. Without any consolidated governance, the company had little consistency in how products were developed, when code was pushed live, if quality checks were in place, or if a canary was deployed before going live. It left Ancestry with 80 to 100 distinct Jenkins instances, requiring additional cost to be stood up and maintained.

âWe were acting like we had an advanced CI/CD culture with the capability to deploy multiple times a day,â said Russ Barnett, Chief Architect at Ancestry. âThe truth is, we didn't have the infrastructure to support that for all of our teams.â

âIt was up to us, the architecture team, to drive standards and best practices, but we had no controls in place to do so,â added Angell. âIn an effort to deliver valuable features to Ancestry's customers, teams lacked the tools to deliver that value quickly without occasionally causing temporary disruptions in service. As a company, we wanted to deliver value quickly and maintain the highest quality service.â

With outages hitting the company's bottom line, executives pushed Angell's team to strive for three-nines availability, or just eight hours and 46 minutes of downtime per year. But, to hit 99.9% uptime, its deployment processes had to change.Â

âWe had the standard processes and best practices already documented, but we had no ability to govern it,â Angell said.

Ancestry decided to leverage the expertise of the company's software engineers to identify best practices and spread those throughout Ancestry. However, doing this manually and team-by-team, per CI pipeline, became unmanageable and Angell had no assurance that any requested change had actually been implemented. For DevOps, it was also a time- and energy-intensive process since each change required constant follow-up that overstretched the team's bandwidth.

Solution

Realizing the development time for a custom solution would be measured in years, not months, Ancestry stepped back to examine its requirements. The team started looking for a third-party solution that would be much faster to deploy and, looking forward, could manage both CI and CD on the same platform. Any solution would also need to standardize deployment and integration processes without hampering developers' ability to quickly ideate, innovate, and maintain new products.

âWe considered each vendor's governance capabilities, security model, breadth of coverage of different deployment types, and ability to configure with code,â Angell explained. âWhen it came down to it, Harness met our requirements and was one of few with an Elastic Container Service (ECS) process out of the box.â

Partnering with different stakeholders (Security, Ops, Platform, etc.) allowed Angell to get a better view of the challenges and potential solutions. Once it selected Harness, Ancestry was able to onboard 350 systems in the first year and an 85% reduction in effort when compared to Jenkins. Ancestry also expected full adoption across all systems by the end of year two.

Results

Harness gave Ancestry a new standard for pipelines that were easy to adopt, change, and govern. That oversight helped to quickly ensure canary deployments became the default Ancestry deployment strategy. Even more, Ancestry's centralized testing tool's adoption rate rose to 97%.

âOnce teams are using a Harness pipeline, it's easy to add additional mandatory steps,â said Barnett. âOur teams were initially skeptical of getting rid of their Jenkins pipelines for Harness, but now they're big fans of the Harness user interface and the ease it brings to the deployment process.â

Its prior Jenkins approach required slow, manual changes to each of its 80 pipelines, but Harness lets Ancestry now make a single change that applies to all pipelines. The increase in pipeline governance and consistent usage of best practices enabled by Harness also resulted in a 50% decrease in overall production outages and significantly reduced outages related to deployment issues. Harness further helped Ancestry increase developer productivity with a 3x increase in deployment velocity.

âHarness now empowers Ancestry to implement new features once and then automatically extend those across every pipeline, representing an 80-to-1 reduction in developer effort.â - Ken Angell, Principal Architect at Ancestry.

Now, as Ancestry continues to bolster its CD governance and compliance efforts, it's looking to expand its use of Harness up the DevOps process into CI. Ancestry is also taking advantage of Harness options for cloud hosting, zero touch maintenance, and automatic backups, all of which give Angell and team more time to build a robust, consistent, and governed CI/CD process.

âIâm going to focus on improving availability and our ability to govern change management in production environments,â concluded Angell. âWeâre letting Harness take care of the infrastructure.â

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/uwm-deploys-in-minutes-instead-of-hours>

UWM Deploys in Minutes Instead of Hours

UWM leverages Harness for self-service Kubernetes deployments and infrastructure creation, reducing deployment time from hours to minutes!

About

UWM is dedicated to making dreams come true for their team members, their broker partners, and homeowners nationwide. Everything they do is designed to support their partnersâ businesses and grow the broker channel as a whole. Theyâre a winning team working together to deliver the fastest turn times in the industry, develop groundbreaking technology, and wowing their brokers with friendly, personal service. Relationships come first. Thatâs why theyâre one of the fastest-growing companies in metro Detroit and why more brokers choose UWM than any other lender.

From Scratch: Containers and Kubernetes

UWMâs executives wanted to bring more new features to customers with less downtime risk. On paper, this was a great decision. In reality, the companyâs DevOps team had to figure out how to go from legacy monolithic applications to containers and Kubernetes. Containerizing UWMâs applications was the only way theyâd be able to create a scalable and repeatable deployment process, but containerizing applications and deploying to Kubernetes was complex.

The team had two options. They could either build everything from scratch using Jenkins and custom groovy scripts, or they could purchase a solution.

The DevOps team estimated it would take several months to code the pipelines in Jenkins and would require dedicated hours every week to maintain the tool.

âIn a past position I tried to set up a simple-to-use integration in Jenkins and it took months. We needed a tool that would take minutes.â

After scoping the build, executives decided it was infeasible to custom script a Jenkins deployment solution. Instead, UWM turned to Harness for modern software delivery.

Faster, Safer, and Scalable

UWM took roughly one day to set up the necessary integrations in Harness. The new deployment methodology is repeatable, takes a couple of minutes, and occurs on demand.

After an image is built and published to Artifactory, a Harness pipeline is triggered to deploy the application to the first integration environment. Harness then creates a card in Jira representing the new build. A Jira user can then go into their Kanban board and move the build card to different stages, each time the card is moved a Harness workflow is triggered for that environment. The entire process is self-service for developers.Â

This has sparked a shift left revolution at UWM. Developers are buying into a new deployment methodology that gives them more control over releases.

âKubernetes is a complex tool that can be intimidating for developers to dive into. Harness abstracts away that complexity to allow anyone to deploy to Kubernetes.â

The UWM team also found a unique way to leverage Harness to automate application setup.

Whenever a dev team wanted a new microservice, they had to submit a request for the DevOps team to spin up pipelines, environments, and repos. This platform configuration took 2-3 hours per microservice with roughly 14 requests coming in every week. To tackle this issue, the DevOps team created an application that leverages Harness to automatically spin up all the application configurations for a development team. The entire process happens instantaneously as soon as a developer submits a request.

Harness has ensured UWM will be able to scale as they continue deploying more microservices.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/comparison-guide/flexera-vs-harness-cloud-cost-management>

Comparison Guide

Harness

Cloud Cost Management

VS

Flexera

Harness Cloud Cost Management vs Flexera

Cloud Cost Management

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

Flexera

Flexera's Cloud Cost Optimization gives a comprehensive set of cloud cost optimization capabilities designed to easily reduce costs across your entire cloud environment.

1300

2008

Privately Equity Owned

Flexera is privately owned by equity investment firm Thomas Bravo, which acquired Flexera for \$2.85B in Dec 2020.

Flexera is categorized as: IT Asset Management

What is the difference between Harness Cloud Cost Management vs. Flexera?

Flexera's strength in IT asset management gives them some interesting cloud cost management features, but they lack support for Kubernetes and cost savings automation that Harness customers rely on to save them even more on their cloud costs.

Flexera vs Harness: Cloud Cost Management

Updated

November 23, 2023

SaaS and Self-Hosted

SaaS and On-Premises

AWS, Azure, and GCP

AWS, Azure, GCP, OCI+SaaS Apps

<with><with>

Kubernetes Only

<no><no>

Percentage Cloud Spend

Percentage of Cloud Spend

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<with><with>
From Kubecost

<yes><yes>

<yes><yes>

Coming soon

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<with><with>

<yes><yes>

<with><with> Manual Schedule

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<with><with> From Kubecost

<yes><yes>

<with><with> From Kubecost

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>

Summary:

Flexera is one of the leading IT Asset Management companies, giving their users a complete view of their IT infrastructure, including their cloud inventory. It was a natural extension of their portfolio to provide reporting on cloud costs in addition to the cost of SaaS and on-premise IT applications. However, the depth that Flexera has applied to IT asset management (ITAM) and governance has only partially made its way into their cloud support. They lack any support for Kubernetes or spot instance orchestration, which means their customers are missing out on more cost savings.

Anyone that deploys Kubernetes in their cloud infrastructure knows that shared cluster costs can quickly spiral out of control if they aren't proactively managed, and that's where Harness wins. We give our customers rich, granular insights into their Kubernetes costs, provide right-sizing recommendations at all levels of the cluster, and proactively manage cluster scaling and costs with Cluster Orchestrator, saving our customers anywhere from 70-90% on their Kubernetes cluster costs.

Anomaly Detection:

Misconfigured scripts, test clusters left running, run away Lambda functions—all of these things can cause your cloud bill to spiral out of control unless you have intelligent cost anomaly monitoring and alerting in place. Not having any automated cost monitoring in place is bad, but having too many distractions in cost anomaly alerting can cause important alerts to go unnoticed.

Flexera's cost anomaly alerts is a static reporting module that cloud administrators have to manually configure to receive alerts (brush up on your standard deviations). The inputs as to what constitutes an anomaly are a best guess effort from the administrator attempting to find a one-size-fits-most anomaly threshold. It's also a global alerting policy that can't be customized to alert only affected users because they don't support custom tags in the reports. Everyone gets only email alerts, so no support for Slack or other integrations. For large enterprises, this will be a lot of alerts that lack the data needed for prioritization.

Harness cost anomaly detection leverages artificial intelligence and machine learning (AI/ML) to dynamically detect and alert on cost anomalies as they occur, using a variety of statistical methods to find true anomalies and filter out the noise. We report anomalies to the affected users by Cost Perspective, so only the folks that need to take action are getting alerted. We deliver actionable alerts without a lot of noise.

Kubernetes Support:

Modern SaaS infrastructure relies on containers and Kubernetes for easy scalability and flexible deployment models. However, shared cost allocation is a challenge unless you have the tools in place to dig into costs by workload/node/pod/label, etc. Harness CCM provides detailed cost analysis for Kubernetes clusters at a granular level that can be tailored for your business. We also deliver automated cluster cost management features such as Cluster Orchestrator for EKS that can save our customers up to 90% on cloud costs. **Flexera has no support for Kubernetes at all.**

Idle Resource Management:

One of the biggest sources of cloud waste is idle cloud resources, especially in test/dev environments. Your engineering teams don't work 24/7, so why would you want their cloud instances to? Shutting down idle cloud resources while they aren't in use can deliver major cost savings, but infrastructure teams struggle to automate this to reduce friction with engineering productivity.

Flexera does provide schedule based instance shut down, but it's little better than a manually maintained Lambda script. While this can save on cloud costs, it leaves a lot of cost savings on the table, especially for test resources that are not used consistently during working hours. Any deviations needed from the predetermined schedule need manual administrative attention to execute.

Harness Cloud Autostopping automatically detects and halts idle cloud resources when they aren't in use, and brings them back up again once an application or engineer accesses them. Our customers configure the Cloud Autostopping policy once, and we do the rest, freeing up DevOps teams to focus on more value added activities than managing manual shut-down schedules. Harness customers save up to 80% on the cloud costs with Cloud Autostopping.

Cloud Cost Management

Interested in seeing what's under the hood? Browse through the Harness Cloud Cost Management Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/soulcycle-ci-cd-harness>

Soulcycle Reduced CI/CD Time & Effort by 80%

Learn how Soulcycle evaluated open-source and Harness for CI/CD and reduced their deployment effort by 80%.

About SoulCycle

SoulCycle is a 45-minute indoor cycling class that features high-intensity cardio, muscle-sculpting strength training, and rhythm-based choreography. Its customers consider SoulCycle classes to be much more than just a workout; they see them as a powerful mind-body experiences. With more than 82 studios (and counting!) their revolutionary indoor cycling class is available across the U.S. and Canada.

CI/CD Before Harness

SoulCycle performs daily deployments across 20+ services and growing.

Deployments in the past typically took developers 1 hour each and 3 QA engineers 3 to 6 hours to verify. Both the deployment and verification processes were done manually, making it difficult to measure success of deployments beyond the site staying up.

DevOps teams previously held the keys to deployments, and wanted to shift that responsibility to the developers so they could be more autonomous and deploy without dependencies.

"Developers need to be able to build and ship their applications to the right places at the right time," said Mark Sost, Engineering and Product Leader at SoulCycle Inc.

The Compelling Event

SoulCycle decided to migrate from a legacy PHP application to a cloud-native, microservice architecture so they could independently scale and roll out new applications.

Evaluating Harness and Open-Source

Prior to Harness, SoulCycle evaluated an open-source alternative for a month, managing its components & configuration on their own. The open-source solution lacked configuration-as-code, documentation, and support. In short, it created too much management overhead. SoulCycle would have needed 3 people (1 lead engineer, 1 senior DevOps and 1 QA) to manage and operate this solution for Continuous Delivery (CD).

In contrast, Harness offered âContinuous Delivery as-a-Serviceâ and took care of the previous management overhead by streamlining the deployment process. Harness also had full support for configuration-as-code with automated deployment, verification, and rollback working out-of-the-box.

In addition, feature and support requests were quickly turned around by Harness engineering, and the dedicated customer success team made on-boarding and deployment easy, simple, and painless.

Harness successfully integrated with SoulCycleâs technology and tools consisting of:

- Go, Node.JS, Python, Postgres
- Google Cloud Platform (GCP), GKE , Kubernetes, Docker for Container Orchestration
- Cloud SQL, Big Query, PostgreSQL, MySQL
- Travis CI, Cloud Builder and GCR for Continuous Integration (CI)
- StackDriver and Datadog for monitoring

Key Harness Benefits

Harness provided the following benefits for SoulCycle:

- Automation of deployments across 5+ Kubernetes clusters and 100+ pods
- Configuration-as-code (YAML)
- Reusable pipeline templates/patterns across dev teams and applications
- Support for Blue/Green and Canary deployments
- Out-of-the-box integration with Datadog
- Auto-verification of deployments using ML
- Auto-rollback for failed deployments
- Custom integration hooks for customer-built toolsets
- Customer Success team, on-boarding, support line and velocity of feature requests

Reduced Deployment Time & Effort by 80%

SoulCycle has been able to reduce deployment time and effort by as much as 80%.

Deployments now take 1 developer 10-15 mins instead of 60 minutes. QA and verification time for deployments has also been reduced by 80%, and deployment verification now takes 3 QA engineers a single hour instead of 3 QA engineers spending 3-6 hours each.

In addition, SoulCycle avoided 3 FTE resources to manage the open-source CD alternative, worth well over \$100k of fully-loaded cost.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform®

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/xactlys-single-pane-of-glass-ci>

Xactly's Journey from Shadow IT to a Single Pane of Glass CI

With Harness CI, every build from commit to completion takes the same amount of time for Xactly. There is no queueing or build failures. This single consistent performance capability has helped with productivity immensely.

Harness Impact

- Developer experience went from âimpossible to debugâ to full traceability for deployments through CI pipelines.
- Removed an EC2-based Jenkins farm amounting to \$575000/year

About Xactly

Xactly is the company behind the popular revenue operations products such as Xactly Incent, Xactly Forecasting, and Xactly Operational Sales Management - part of its Intelligent Revenue Platform. Its users benefit from the platformâs precise compensation plans, better incentives, and data-informed insights to give them more confidence in their pipeline. Itâs the kind of product most sales organizations love to have for many reasons, not the least of them being that it makes sense of compensation plans.Â

Xactly's Mission to Blend Modern Software and Data Science

To meet the expectations of excellence, the software Xactly delivers this value through has to meet the highest standards. Revenue is not something anyone wants to mess with. It not only has to be precise, but also go a step beyond it to provide a bigger picture to all incumbents. This experience takes the shape of Xactlyâs Intelligent Revenue Platform, and Alby Graham, director of engineering, is one of the leaders behind its success.Â

Graham oversees two fundamental pieces of Xactly software delivery efforts: on one hand, his team manages the platform that is designed for rapid application delivery; this same framework, as they like to call it, also manages a series of reference applications that feed data into the Incentive Compensation Management (ICM) solution. This last piece of the puzzle is the one responsible for providing AI and ML capabilities to the platform that, in turn, provide insights to Xactlyâs customers â basically, a combination of pure software development and advanced data science in one single software delivery effort. For these operations to run smoothly, Alby has around 60 engineers and data scientists reporting to him.Â

Xactlyâs CTO doesnât want to waste resources. Xactlyâs technology leadership knows developing software will require strong investment, but they also know that if run efficiently, it can have a bigger impact on the business. Shadow IT, wild use of public cloud, and other consequences of perverse incentives that come from anarchic software delivery pipelines can be cut down without hindering the developer experience and software quality. In fact, some constraints will not only make the CTO happy, but the end user as well. Letâs dive into Xactly's journey to modern CI.

Monolithic Builds Becomes A Tale of Continuous Integration and Fishing

Alby tells the following anecdote about one of his best performing engineers: a production release was programmed to be delivered. However, that developer ran the build and Jenkins crashed. The developer let Alby know that he was going to go fishing since he knew that Jenkins wouldnât be back up until the next day. Alby was completely ok with this, because itâs true. Jenkins wasnât going to be up and running until the very next day.Â

This is the story of many Jenkins shops. Jenkins is adopted and run, letâs say in the public cloud. The companyâs software development teams grow and this initial CI implementation keeps working. The growing teams start committing more and more code to both the platform and the ML model repos. The Jenkins instanceâs performance starts lagging, the build queue starts growing, and builds pile up.Â

But letâs get back to Xactlyâs particular Jenkins instance. It crashed or halted at least two to three times a day, rendering that pipeline idle. While the pipeline was running, however, queues would be around six or seven builds long at any given time during working hours. Effectively, someone would make a commit and only hours later would hit the target environment because of Jenkins.Â

It was also not infrequent that the build âdied in the middle.â At one point, there were 35 builds in the queue because a build died in the middle, and Albyâs team had to wait an hour to restart it.Â

Fixing Build Speed for the Win

The initial solution was to split that Jenkins bottleneck into other Jenkins instances. That's when the second variable of the Jenkins problem kicked in: the Jenkins sprawl. The event that triggered the Jenkins build process was commit to a central repo. The problem is that up to four or five Jenkins instances picked the same commit each time, generating multiple builds and deploys in parallel. There were several problems to this new approach to CI. One of the most important ones was traceability or auditability: it was effectively impossible to track back, let's call it *build 123* or *release JE 12*, to a specific Jenkins instance and its build pipeline. A second variable to this problem comes from answering the question: how much can you rely on tests if you don't know where they are running? What's the test coverage? Which build pipeline ran which tests? All those questions eventually became unanswerable - until Graham onboarded Harness Continuous Delivery (CD).

Once this scenario was fixed and only one Jenkins machine would pick up one commit each time only, it seemed that the situation was under control. Xactly already had Harness CD in use, which was listening for new Docker image tags that the Jenkins build system was updating after each build that produced a new artifact, and it would update the image and its tag pushed to Docker Hub. But again, the image tag is not the only parameter that a software delivery process should rely on. One cannot make the whole production environment rely only on an image tag whose source is fuzzy, at best.

Developer Productivity and Single Pane of Glass: No Trade-offs

Naturally, managing such a scenario became a burden. Costs were growing and most of them came from truly unproductive tasks that were, in effect, hindering development to the business critical applications: the platform and the intelligence within it. A single Jenkins instance running in an AWS EC2 instance can cost \$36,000 or more per year. Imagine 16 Jenkins instances running wild on EC2 compute layers, the cost becomes astronomical just to run CI only.

Financial costs aside, wasting 40 developers' precious time because a build system crashed and no one can get the most recent code for hours is a big deal. Developers lose trust in the delivery platform and are incentivized not to commit so frequently because they anticipate that problems will arise. That hinders developer confidence, which is one of the main drivers behind the company's success. To get rid of this repeating scenario, a plan was put in place to progressively sunset all Jenkins instances and provide a better solution.

Pay when it's due: fast and easy like Xactly

Xactly's approach to software development was that if a tool is purchased to support the delivery pipeline, it has to solve the problem with a deep feature and capability set. Spinnaker was, for this very reason, evaluated but not considered: a broader but somehow shallower feature set would not solve the problem with CI. Since Xactly was already a Harness CD customer and was familiar with what Harness is able to do, exploring Harness CI Enterprise was a natural step in the same direction. "Do one thing and do it well," Alby said.

Stability and time from commit to deployment are the two main areas in which Xactly has seen an improvement. For one, Harness CI has not crashed a single time since it was implemented, and that's a tremendous gain compared to the previous state with Jenkins. For another, removing an EC2-based Jenkins farm resulted in savings of \$575,000 per year.

Go Fishing With No Worries in the Back of Your Mind

Ultimately, every build from commit to completion takes the same amount of time. There is no queuing or build failures. This single consistent performance capability has helped with productivity immensely. Developers can see the impact of their changes in their alpha environment in a matter of minutes, compared to the multiple hours it would take with Jenkins. This incentivizes developers to contribute more and to try new things. Speed in CI means that more tests, even manual tests, can be run without hindering the process. Having CI be part of a bigger platform provides developers with shorter feedback loops, which in turn sparks creativity and promptness among them.

Looping back to Alby's anecdote about one of his best developers going fishing rather than waiting for the Jenkins instance to get back up, this is what he had to say after Harness CI ran for only a couple of months in Xactly's pipeline: "I don't hear people complain anymore about builds. I can't tell you how much better that makes me feel. Those things are hard to quantify, but so much more impactful than the dollars."

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/comparison-guide/gitlab-vs-harness-ci>

Comparison Guide

Harness

Continuous Integration

VS

GitLab CI

Harness Continuous Integration vs GitLab CI

Continuous Integration

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

GitLab CI

GitLab CI works to integrate code provided by your team in a shared repository. Developers share the new code in a merge request (MR).

1000+

2014

434.2m

Gitlab is categorized as:

Continuous Integration
Continuous Delivery
Feature Flags

â

â

â

What is the difference between Harness CI Vs. GitLab CI?

GitLab CI vs Harness CI: DevOps Tools Comparison

Updated

November 30, 2023

- Open Source Version
- GitHub Stars
- Self-Service (Simple)
- No Scripting Required
- Container & Cloud-Native
- Traditional App Support
- GitOps (Pipelines as Code)
- Any Source Code Manager
- Containerized Pipelines
- Containerized Plugins
- Secrets Management
- Command Line Interface
- Scalability (Required Infra)
- Admin & Maintenance
- Total Cost of Ownership
- Pricing

Free & Paid

24800

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Vault/KMS/3rd

<yes><yes>

Lightweight

<yes><yes> .25 FTE

<yes><yes>

<yes><yes> Per User

Free & Paid

â

<with><with>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<with><with> **2 FTE**

<yes><yes>

<yes><yes> **Per Minutes OR Per User**

Free & Paid

Free & Paid

24800

â

<yes><yes>

<with><with>

<yes><yes>

Vault/KMS/3rd

<yes><yes>

<yes><yes>

<yes><yes>

Lightweight

<yes><yes>

<yes><yes> .25 FTE

<with><with> 2 FTE

<yes><yes>

<yes><yes>

<yes><yes> Per User

<yes><yes> Per Minutes OR Per User

Testing and Verification

GitLab CI comes with a set of test suites out-of-the-box. They are pretty standard and apply to any language. Harness CI goes beyond that. Test Intelligence is one of Harness CI's most advanced features, and it is unique in the CI market. Test Intelligence dramatically reduces build times. It does so by running incremental builds which allows Harness to run relevant tests and bypass unnecessary tests. With Test intelligence on, Harness CI customers have seen build times shrink by 40%. These are just a few of the gains an advanced testing feature like Test Intelligence can provide to a modern CI system compared to GitLab CI.

Open source vs. Open core:

GitLab is not an open-source product. It offers a free plan with 400 CI/CD minutes â possibly good for personal projects and small businesses â but otherwise, expect to pay a hefty bill for your CI needs. When Harness acquired Drone, it committed to keeping it open-source forever. Harness recently reaffirmed its investment in the open-source solution with a massive release where a sleeker interface, new visual pipeline builder, governance and security features, and real-time debugging tools were added. While this feature-rich version is free, there is also a paid version of Drone that provides access to enterprise support and more integrations and features yet. Additional features include secrets management options, autoscaling, custom plugins, and more.

Self-Service (Simple):

A plus to using GitLab is that it was, initially, a source code management tool / Git repository. As such, converting to their CI/CD platform would have advantages when it comes to easy integration. However, if GitLab is your SCM tool of choice, rest assured that Drone easily integrates with it as well. When it comes to self-service in GitLab, some features are buggy and the overall system can be quite slow. Documentation is lacking for more complex setups. The UI is clean, but not intuitive â it definitely has a learning curve and needs improvements in order to be less confusing. Lastly, CI can be hard to integrate with automatic and manual tests users have created in the past with their prior CI tool. Drone offers an easy âget startedâ experience where you can be up and running in 5 minutes. Drone also benefits from roughly 150 containerized plugins, profoundly extending the functionality of the tool. Drone scales on demand. All of this means less person hours spent by engineers maintaining the tool or waiting for slowness to resolve, and more time on what matters: getting that code to artifact.

Requires Scripting:

If you have an extremely simple setup, you can possibly avoid scripting in GitLab â but realistically, there will be scripts. While scripting isnât the bane of our existence, it is time that could be spent shipping code. It is toil. Drone is much more intuitive and is mostly configuration as code. Gone are the days of scripting.

Secrets Management:

GitLab does not offer native secrets management capabilities. They have selected Vault by HashiCorp as their first supported secrets management partner, which means you must first configure your Vault server. Drone offers encryption on its open-source version. Meanwhile, the enterprise version offers these alternatives: encrypted, native, or external, through third-party providers such as AWS Secret Manager, Kubernetes Secrets, and HashiCorp Vault. No matter how you want your secrets to be handled, Drone can rise to the occasion.

Admin & Maintenance:

GitLab, on average, requires 2 FTEs to set up and maintain. Drone is incredibly lightweight and self-service, and as a result, only requires .25 FTEs. The amount of maintenance is extremely low thanks to simple, container-native infrastructure.

Total Cost of Ownership:

When comparing products, it is imperative to take all costs into account. With a product like GitLab, you're not just paying for the tool â you're paying for the 2 FTEs it takes to keep the tool in working condition. It's a large cost to take into account. With Drone, the end result is a tool that is very affordable, with a small commitment of .25 FTEs and low or free pricing (depending on if you decide the features for the enterprise plan are too good to pass up â who doesn't want autoscaling?).

Pricing:

GitLab offers, as mentioned above, a free plan that aligns well with personal projects or small businesses just starting out, with 400 CI/CD minutes built in. Otherwise, plans are priced per user per month, with extra features on each paid plan. For the premium plan, you'll be paying \$19 per user per month. The ultimate plan will run your company \$99 per month per user. Drone does offer an open-source version that is free, and while the enterprise version does provide an arguably more robust product, the free version is already quite feature-rich and will suffice for many use cases. Download Drone now. To familiarize yourself with enterprise pricing, please contact sales.

**Please note: Our competitors, just like us, release updates to their products on a regular cadence. We keep these pages updated to the best of our ability, but there are bound to be discrepancies. For the most up-to-date information on competitor features, browsing the competitor's new release pages and communities are your best bet.*

Try Harness For Free

Continuous Integration

Interested in seeing what's under the hood? Browse through the Harness Continuous Integration Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/comparison-guide/codefresh-vs-drone>

Comparison Guide

Harness

Continuous Integration

VS

Codefresh

Harness Continuous Integration vs Codefresh

Continuous Integration

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

Codefresh

Codefresh's DevOps automation platform brings CI/CD, Kubernetes, GitOps and more to help companies confidently deploy software faster.

100+

2014

42m

Codefresh is categorized as:

Continuous Integration

â

What is the difference between Harness DevOps Tools Vs. Codefresh?

Codefresh vs Drone: DevOps Tools Comparison | Harness

Updated

- Open Source Version
- GitHub Stars
- Self-Service (Simple)
- No Scripting Required
- Container & Cloud-Native
- Traditional App Support
- GitOps (Pipelines as Code)
- Any Source Code Manager
- Containerized Pipelines
- Containerized Plugins
- Secrets Management
- Command Line Interface
- Scalability (Required Infra)
- Admin & Maintenance
- Total Cost of Ownership
- Pricing

Free & Paid

24800

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Vault/KMS/3rd

<yes><yes>

Lightweight

<yes><yes> **.25 FTE**

<yes><yes>

<yes><yes> **Per User**

Free & Paid

50

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<with><with>

<yes><yes>

Lightweight

<with><with> **2 FTE**

<with><with>

<no><no> **Concurrent Builds**

Free & Paid

Free & Paid

24800

50

<yes><yes>

Vault/KMS/3rd

<with><with>

<yes><yes>

<yes><yes>

Lightweight

Lightweight

<yes><yes> .25 FTE

<with><with> 2 FTE

<yes><yes>

<with><with>

<yes><yes> Per User

<no><no> Concurrent Builds

Open source vs. Open core:

Codefresh was founded in 2014. It only offers a paid version â no open-source version is available. Drone is open-source. There is, however, a paid version of Drone that provides access to enterprise support and more integrations and features. Additional features include secrets management options, autoscaling, custom plugins, and more. Weâd also like to mention that Codefresh only provides an on-prem version of their software when youâre on the enterprise plan. Drone is, of course, on-prem on its free version as well as

enterprise.

Self-Service (Simple):

For the most part, Codefresh is self-service. There are gaps, of course, as it is a newer tool. For example, there isn't much community support available and the documentation is spotty. It can therefore take some time to get properly acquainted with the product. However, once you familiarize yourself and get past the learning curve, it's a good platform. Drone is built upon three pillars that enable engineers to build and test code quickly and accurately: simple, scalable, self-service. Drone installs in under 5 minutes, scales on demand, and all plugins run in containers on their latest version. This means less person hours spent by engineers maintaining the tool, and more time on what matters: getting that code to artifact. It is worth noting that Codefresh has fewer plugin selections than Drone, so some time may be spent creating your own solutions, should you choose Codefresh.

Secrets Management:

Codefresh recently added a new feature called Secret Storage, which allows you to keep sensitive data on your cluster and for Codefresh to request it during pipeline execution on user's demand. This feature is only available on their enterprise plan. Drone offers encryption on its open-source version. Meanwhile, the enterprise version offers these alternatives: encrypted, native, or externally, through third-party providers such as AWS Secret Manager, Kubernetes Secrets, and HashiCorp Vault. No matter how you want your secrets to be handled, Drone can rise to the occasion.

Admin & Maintenance:

When it comes to maintenance, Codefresh isn't nearly as bulky as a Jenkins, for instance. Plugins are run in containers, so there's no need for maintenance there. Overall, Codefresh typically requires 2 FTEs to set up and maintain. However, it's still a far cry from Drone's .25 FTEs. Drone is an extremely portable solution without scripting, plugin maintenance, or dependency hell â or much else in terms of maintenance, for that matter. The plug-and-play nature of Drone ensures the only work you have to perform, other than the initial setup and configuration of course, is administration â such as adding and removing users, permissions, etc.

Total Cost of Ownership:

One of the downfalls of Codefresh is the amount of resources it consumes. Users have stated in the past that it's easy to lose track of resources and end up with a heftier-than-expected cloud bill. Keeping a watchful eye on resource consumption will be necessary. Add on top the 2 FTEs required to keep Codefresh running, and it makes for a pricey solution. Drone, however, only costs .25 FTEs â and is completely free. Even the feature-rich enterprise version of Drone is more fiscally prudent.

Pricing:

Codefresh offers 3 different pricing plans. Their basic plan, which is a very pared-down barebones plan, allows up to 3 users and supports one concurrent build. Prices are based on instance size. For example, if you chose 1 small instance, 3 medium, and 1 large, you'll be paying \$444 per month. Their pro plan starts at \$34 per month for up to 10 users with 1 concurrent build on 1 small instance. The same example as the basic plan would cost you \$570 per month on the pro plan. Codefresh also has an enterprise plan, but pricing is unavailable publicly as it will depend on your individual circumstances and needs. This plan is the most feature-rich, of course, offering secrets management, RBAC, SAML, SSO, and can accommodate unlimited users. To add another layer of complexity, these prices are all if you sign up on an annual basis â monthly pricing is more expensive. To compare, our \$570 per month example from above would cost \$825 on a monthly basis. Lastly, there is a free tier in the basic plan that supports 1 concurrent build on 1 small instance and up to 3 users. This could potentially be enough for personal projects. Drone is free and available for download. It also has an enterprise version that is extremely feature-rich, but does have pricing attached to it. To familiarize yourself with enterprise pricing, please contact sales.

**Please note: Our competitors, just like us, release updates to their products on a regular cadence. We keep these pages updated to the best of our ability, but there are bound to be discrepancies. For the most up-to-date information on competitor features, browsing the competitor's new release pages and communities are your best bet.*

Try Harness For Free

Continuous Integration

Interested in seeing what's under the hood? Browse through the Harness Continuous Integration Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/warehouse-group-decreases-lead-time-changes-99-percent>

The Warehouse Group Decreases Lead Time For Changes by 99.1%

Harness CD has helped The Warehouse Group standardize the deployment process and empower their dev squads with easy integration, speed to production, and a range of efficiency features.

Harness Impact

- Deploy on-demand
- Complete squad autonomy
- Lead time for changes decreased from 120 hours to 1 hour by using Harness as a key part of their path to production.

About

If you're from New Zealand, you likely know The Warehouse Group. It's the country's largest retailer, with 260+ retail stores, distribution centers, as well as a strong online presence. With \$3 billion in sales and 12,000 employees, the company has six core brands, including The Warehouse (TWG), Warehouse Stationery, Noel Leeming, Torpedo7, 1-day, and TheMarket. Its vision is to help kiwis live better every day and build New Zealand's most sustainable, convenient, and customer-first company.

Challenges

The Warehouse Group has been focusing on building up its digital business infrastructure, including its DevOps squad, with a focus on automation, Continuous Delivery (CD) platforms, and new processes. As Matt Law, Chapter Lead of The Warehouse Group's DevOps Chapter, put it, "We wanted to ensure that we have a consistent way of working with developers across all of our brands so we have an amount of reuse and common process across the group." Ultimately, they wanted their DevOps squads to be able to work autonomously to support their business goals.

The squad was using Jenkins, an open source automation server known for CI orchestration. The intention was to see if Jenkins was the right fit for both CI and CD. The squad sought to improve processes and tools to support a SaaS first strategy for new work. This meant that any solutions would need to have a rich API and the capability to configure all as code. As Law put it, "We were the definition of inefficiency, because it was all completely bespoke."

1. Deployments Were Inefficient

The Warehouse Group set out to improve and automate many aspects of the deployment process. Two bottlenecks stood out: testing was not integrated into the pipeline and the squad had to wait on a Change Approval Board (CAB) to deploy into production. In fact, manual testing could add up to 6 weeks, and CAB times could be up to 2 weeks to deploy to production. Combined, delivery times were very slow.

2. Onboarding DevOps Squads Was Difficult, Time Consuming

Getting DevOps squads up, running, and delivering value into production was "a huge pain," Law said. To fully provision access to systems and create pipelines to get deployments working correctly, new squad onboarding took up to 6 weeks. New service setup for any squad took approximately 80 hours.

3. Uncertainty on Delivery Times

The Warehouse Group looked to improve internal developer squads' time-to-value for the business. A focus area to accomplish this would include embedding testing frameworks directly into pipelines to alleviate manual effort from the squads. Law explained, "The Warehouse Group needs to be able to react in this very competitive digital world so that we can get value to customers as quickly as possible."

Solution: Confluence of Events Lead Straight to Harness

As DevOps leadership at The Warehouse Group assessed these issues, they determined that they needed a platform they could instantly leverage, with built-in capabilities to meet all their needs. While Jenkins could have solved for their needs, it did not do so out of the box.

Their search began with the goal of automating the CD process, which required reviewing the service release process, understanding the testing results, and having an automated rollback mechanism. The replacement would also need to align with the cloud-first goals of the company so a SaaS solution was the obvious choice. This selection process also dovetailed with a larger internal DevOps project called "The Golden Path." The Golden Path aimed to create a standardized, template-driven approach for creating and deploying software more effectively.

Law said, "I knew about Harness and was aware of some of the capabilities. I love the fact that it's straight out of the box with ServiceNow integration. That was one of the biggest things that we needed to get right," Law said. Harness also had the support for deploying in Kubernetes.

Of course, they needed to do a Proof of Concept (POC). In Law's words: "It's one of the smoothest POCs I've ever had. It was so seamless for us. The Harness team was able to prove what we needed to in two weeks. They did a lot of the work in advance to make sure customers can tick the boxes."

Between the ServiceNow integration, API first approach, and being cloud based, Law was sold on Harness. It quickly became apparent

that Harness CD would be a cost-effective and no-maintenance solution.

Results: Pipeline Onboarding Down to Seconds

As part of the Golden Path project, Backstage orchestrates the provisioning of the DevOps platforms, bootstrapping the environment with Backstage, and within 7 seconds, Harness. Within about 30 minutes of creating a service, the squad has builds deployed in development and test. Previously, the pipeline onboarding process would have taken six weeks.

As 7 of their squads have already ramped up on Harness, The Warehouse Group has seen a lot of growth. The Integration Modernisation Squad, for example, has 24 services so far through Harness, and delivered about half of them into production already in under 3 months. Law said that hasn't happened before in the company: "Harness is playing a key part in delivering value and speed to delivery."

1. Complete Squad Autonomy

As one of the most important goals of the project, developer squads have the freedom to deploy into production whenever they want, made much easier with Harness enabling the ServiceNow integration. Law explained that developers have trust and ownership, leading to a quality-first approach. "This inherent trust and ownership generated by the capabilities of Harness CD is an enabler for us for our culture too. They are one of the main drivers for adopting a new application or way of working."

2. No Training Needed

With Harness, extensive training is not a requirement because development squads can intuitively and easily see the status of the deployment into the different environments — from development, test, and production, and whether they're successful, past, or paused. Law said, "The great thing about Harness is that it's presented in a way that's really user friendly: you can see the stages, you can see what it's doing, you can see any potential errors. So effectively, you don't really train on it because it's quite obvious."

3. Deploying On-Demand

Harness CD has helped The Warehouse Group standardize the deployment process and empower their dev squads with easy integration, speed to production, and a range of efficiency features. Harness played a key part in propagating the necessary information from testing into ServiceNow without additional manual effort from the squads. This resulted in saving some squads 100 development hours in order to successfully deploy and deliver value to the business, which wouldn't have been possible without Harness.

With the help of Harness, the DevOps squads are delivering value at such speed that they've now been brought into other areas of the business. Law shared: "In terms of the business, people are actually looking to what we've accomplished from a deployment standpoint and saying, how did you do that? Recently, I've had the two biggest telecommunications companies in New Zealand reach out. So we've been talking to them about our DevOps journey and what we've invested in."

To learn more about how Harness can help you modernize your DevOps processes, [request a demo](#).

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/tyler-tech-achieves-unparalleled-velocity-feature-flags>

Tyler Tech Takes CI/CD to the Next Level and Achieves Unparalleled Velocity With Feature Flags

To Tyler Tech, feature flags were a natural extension of CI/CD. Learn why they chose - and trusted - Harness to provide that capability.

About Tyler Technologies

Tyler Technologies (NYSE: TYL) is a leading provider of end-to-end information management solutions and services for the public sector. Tyler partners with and empowers clients to become more efficient, more accessible, and more responsive to the needs of their constituents. Tyler Technologies has an annual revenue of \$1 billion, with over 37,000 installations across 12,000 sites and more than 6,600 team members.

Quote

I wouldn't hesitate for a second to tell somebody how satisfied we are with Harness - they've been a great partner. That we continue to invest in additional solutions just emphasizes how happy we are with the relationship.

CI/CD Wasn't Going Far Enough

One of Tyler Technologies' strategic initiatives is moving from a traditional software delivery model to a cloud-based delivery model. A key benefit for all stakeholders with this change is in having a single product version to support, instead of multiple versions across different customer installs. Since customers are always on the latest version, the challenge is how to deliver new features, rather than how to deploy the latest release.

Before introducing Harness for CI/CD, Tyler Tech deployment teams often used a big bang or a rocket launch releases: A broad set of functionality was bundled together on a periodic basis for a release, with a few consequences. First, large releases concentrated risk into a single deployment, so extensive testing was required prior to release to avoid any negative impact in production. Second, the larger releases required coordination between multiple Tyler teams to ensure a smooth deployment experience, which increased the number of personnel involved. Third, multiple Tyler product teams had to coordinate development roadmaps and release management activities, with limited flexibility in terms of scope and timing.

The shift to the cloud enabled Tyler Tech to try a new approach: decoupling deployment and release. Tyler Tech product teams were asking for more control over feature release and exposure, instead of deployment being the end-all-be-all. Tyler's Corporate Development team began reviewing build and partner approaches that would allow product teams to deploy and release on their own schedule.

As part of the Corporate Development team, Mike Teeters is responsible for supporting that transition to the cloud, and to answer important questions such as "how can UX, Development, and Product do their jobs better?" Mike helps internal teams match technology solutions to their challenges, and as a Product Manager, who better to understand the challenges and the desired outcome for Tyler Tech? In this case, Mike was charged with evaluating deploy-and-release options.

To take the next step in leveling up and increasing the velocity and control in their software delivery process, a solution needed to achieve the following:

- Mitigate the risk built up in each release.
- Give product teams greater control over the release cadence.
- Create more atomic feature sets that product teams could apply their own risk management to.
- Have new functionality in production and control who has access to it.
- Allow sales to demonstrate features even if not fully ready.
- Release feature variations to support A/B testing and make product decisions.

Bringing Together Velocity and Control in the Cloud

The evaluation process quickly identified feature flagging as a necessary part of the toolkit. Tyler Tech was already using Harness for CI/CD deployment pipeline management, so Harness Feature Flags quickly became a focus point. Harness Feature Flags built on their trust in the deployment platform and the strong partnership between the two organizations.

Achieving their desired outcomes was certainly possible with a few solutions, so what made Harness Feature Flags the right fit? First, given that their primary use case was giving product teams more control over the release cadence and feature exposure to customers, it made sense to expand use of the platform. Second, broadening platform access also simplified internal adoption. It was a logical next step to state "there's new functionality on the platform you're already using" instead of "we have a new point solution to bring into our process." Third, Tyler Tech sees CI/CD and feature flags being part of the same software delivery process. What better way to help Product, Engineering, and Deployment work better together than having a unified delivery pipeline where both sides win?

Internally, Tyler Tech already had teams using other feature flag tooling from an acquisition. As they evaluated their current tools and Harness, it was important that the tool of choice met all of their core feature management requirements, plus had added capabilities that aligned well with the Tyler Tech deployment solution. It was important that the whole team could work on the same platform and extend their existing CI/CD process. After all, Tyler Tech saw feature flags as a natural extension of CI/CD, so it only made sense.

In the end, Tyler Tech chose to expand their partnership with Harness by selecting Harness Feature Flags to advance their software delivery process. Why Harness? It came down to three things:

â **Harness Feature Flags was natively integrated into their CI/CD process.** Tyler Tech was already leveraging the Harness platform for CI/CD, and it was appealing to simply roll out new functionality on the same platform everyone was already using, and avoiding the risk of having multiple systems in their mission-critical software delivery pipeline.

â **The responsive customer account team** made it easy and comfortable to work with Harness. Across Sales, Customer Success, and Product, Tyler Tech felt well-supported and like their needs were attended to. And, when Tyler Tech needed to make adjustments because of business realities, the Harness team made it a fluid and painless process.

â **It feels like a strong and evolving partnership.** When Tyler Tech started down the path with Harness, they felt good about it. And now, they appreciate that all the things Harness did well, it continues to do, reinforcing that Harness is the right technology partner. And, it doesn't hurt that both organizations have an engineering-oriented culture!

Tyler Tech may be early on in their feature flag journey, but they are excited about the new reality it will bring to their rapidly evolving organization. Moving to the cloud is one thing, but maximizing all of the benefits of doing so is entirely another. By working with technology partners like Harness, Tyler Tech is confident that they'll be able to deliver on their core mission and continue to meet the evolving needs of their stakeholders and customers. For Tyler Tech, it's just the beginning of a new era in software delivery.

â

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform®

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/tyler-technologies-ci-cd>

Tyler Technologies Saved \$1.2 Million in CI/CD Maintenance Costs

Learn how Tyler Technologies saved \$1.2 million in CI/CD maintenance costs

About Tyler Technologies

Tyler Technologies (NYSE: TYL) is a leading provider of end-to-end information management solutions and services for local governments. Tyler partners with clients to empower the public sector – cities, counties, schools, and other government entities – to become more efficient, more accessible, and more responsive to the needs of their constituents. Tyler Technologies has an annual revenue of \$1 billion.

The Cost Of Custom Continuous Delivery

Before Continuous Delivery-as-a-Service, Tyler Technologies took a “Roman Colosseum” approach to software delivery. Delivery gladiators were hired to battle the company’s deployment issues. These software warriors used their unique, unreplicable, talents to build complex pipeline behemoths. Then, when they left the company a new hero would step up with their own BETTER solutionâ|. You see where this is goingâ| so did Tyler Technologiesâ CTO Jeff Green.

Mr. Green began taking stock of Tyler Tech’s deployment solutions in preparation for a Kubernetes migration. Their legacy applications were deployed using a half dozen homegrown and vendor sourced deployment tools. None of these deployment tools were capable of deploying to Kubernetes and each required 1-2 full-time software engineers to maintain. That’s over \$1.2 million in annual maintenance costs.

One of the homegrown deployment tools was built and maintained by 4 FTEâs. It consisted of loosely connected scripts and dozens of custom-written plugins. The key developer of the project left Tyler Tech and no one else knew how to maintain the tool.

Instead of re-writing and maintaining Tyler Tech’s custom deployment tools, Jeff Green decided to search for a better option.

Harness Business Value And Savings

Tyler Tech completed its first deployment with Harness in 6 hours. In a few months, they onboarded 230 unique services. Harness became an integral part of Tyler Tech’s Kubernetes migration.

In addition to a seamless migration, Harness helped Tyler Tech achieve a best in class change failure rate of 10%. If an environment goes down, it only takes Tyler Tech 25 minutes to do a full re-deploy.

Tyler Tech successfully modernized its CD practices with Harness and saved millions in the process.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi’s 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/comparison-guide/azure-billing-vs-harness>

Comparison Guide

Harness

Cloud Cost Management

VS

Azure Cost Management

Harness Cloud Cost Management vs Azure Cost Management

Cloud Cost Management

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

Azure Cost Management

Microsoft Cost Management helps you analyze, monitor, and optimize your Microsoft Cloud costs; understand and pay your bill; and manage your billing account and subscriptions.

1500 (Azure Team Only)

2008 (Azure Launched)

Public Company

Azure Cost Management is categorized as:

Cloud Cost Management (CCM)

How Does Azure Cost Management Compare?

As with all of the native cloud providers, Azure Cost Management is a suite of free tools that provides basic visibility and cost optimization recommendations, but falls short on provide automated cost optimization, or deep support for Kubernetes and proactive cloud governance that Harness provides out of the box.

Azure Cost Management vs Harness: DevOps Tools Comparison

Updated

November 30, 2023

- SaaS
- Audience & Primary Users
- Cloud Support
- Azure & AWS
- Time Granularity
- Tag Management Required
- Hybrid Cloud Visibility
- Cloud Account Visibility
- Cloud Region Visibility
- Application Visibility
- Microservice Visibility
- Environment Visibility
- Non-Cluster Visibility (Services)
- Cluster Visibility (Kubernetes/ECS)

- Root Cost Analysis (3 Metrics)
- Cloud Event Correlation (e.g. deploy)
- Cost Recommendations
- Alerting & Budgets
- Anomaly Detection
- Reporting
- Documentation
- Training
- Support
- Automated Idle Resource Management
- Cloud Cost Business Intelligence

<yes><yes>

Engineering & Finance

All

Hourly

<yes><yes> Optional

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Utilized, Idle, Unallocated

<yes><yes>

Finance

Azure & AWS

Hourly

<yes><yes> Optional

<with><with> Azure & AWS

<yes><yes>

<yes><yes>

<with><with> **Requires Tagging**

<with><with> **Requires Tagging**

<with><with> **Requires Tagging**

<yes><yes> â

<with><with> **Only AKS**

<no><no>

<no><no>

<yes><yes>

<yes><yes>

<no><no>

yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<with><with> **With PowerBI**

SaaS and Self-Hosted

SaaS

AWS, Azure, and GCP

Azure and AWS

<with><with>

Kubernetes Only

<no><no>

Percentage Cloud Spend

Azure: FreeAWS: Percent of Cloud Spend

<yes><yes>

<with><with>

<yes><yes>

<with><with>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

Coming soon

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<with><with>

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<no><no>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<with><with> Azure Only

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

Detailed Feature Comparison

Summary

Azure Cost Management provides a good set of visibility, savings, and forecast tools for Azure, and visibility into AWS (for a fee). As a free tool available to Azure customers, it provides the basics that finance teams and budget owners would want to know. However, they fall short in providing support for multi-cloud setups and container orchestration outside of AKS, which is where the world is headed. To fully leverage the capabilities of Azure Cost Management and get a deep understanding of costs, there's an inordinate amount of tagging required, and customers find themselves lacking support for day-to-day operations. Azure Cost Management is built for point-in-time views into costs, not for the daily toil that goes into efficient cloud cost management.

If you're looking for rich multi-cloud and container cost visibility, optimization and governance, Harness Cloud Cost Management is the clear winner. Harness delivers automated cloud cost savings features that don't exist in Azure Cost Management, as well as deep visibility into multi-cloud and Kubernetes costs. Our Cloud Asset Governance provides proactive cost optimization and guardrails for future cloud spend.Â

Multi-Cloud Cost Visibility and Optimization

We live in a multi-cloud world, and understanding your cloud costs across all of your cloud providers is an essential requirement for most firms. Visualizing costs by resource, region, and management groups is a minimum requirement when assessing cost management tools.

Azure Cost Management does have support for creating connectors into AWS accounts, so they can give customers unified views of cloud costs by cloud provider, as well as importing that data into their Power BI tool for more advanced cost analytics. Unfortunately, this is where their multi-cloud support stops. They do not provide cost recommendations for AWS resources, nor recommendations for AWS SP/RI commitments, which one should reasonably expect when they are being asked to pay for AWS support.

Harness delivers true multi-cloud support, giving powerful cost visibility via Cost Perspectives and Categories that allow users to create granular, business specific views of multi-cloud costs. We can also provide recommendations across all cloud providers, as well as Kubernetes clusters.

Kubernetes Support

Cloud-native deployments depend on Kubernetes to deliver scalable and powerful services that can be deployed on shared infrastructure. The challenge, though, is getting the full picture of what portion of shared clusters are consumed by different teams or applications.Â

Azure Cost Explorer doesn't include any specific support for viewing AKS cluster costs, and can only statically allocate shared costs at a management group or subscription level. Given the dynamic nature of shared cluster usage, this is a major gap in cost visibility. You also won't get any cost optimization recommendations from Azure Advisor, which is fairly limited in the scope of resources it can cover.Â

Harness not only gives you deep insights into shared cluster costs, but we also provide detailed rightsizing recommendations at the workload and node pool levels. We can also help automate your cluster autoscaling to take advantage of spot instances via our Cluster Orchestrator feature, saving up to 90% off on cluster costs.

Automated Cost Savings:

Getting engineers to take action on cloud cost recommendations shouldn't be your only path to cloud cost savings. Automation can unlock cost savings opportunities that simply can't be realized with manual or scripted efforts. Harness CCM has a multitude of automation tools that save our customers money, from idle cloud resource management (Cloud Autostopping) to spot instance orchestration, and Commitment Orchestration, which save our customers up to 90% on cloud costs when used together.

The only savings automation that Azure (in general, not Cost Management) provides is an option to auto-renew savings plans and reserved instance contracts. There is no intelligence built-in to understand what *should* be renewed, and once a savings plan is renewed, it can not be canceled. Customers could be stuck with contracts that no longer meet their needs, leading to cloud cost waste.

Cloud Cost Management

Interested in seeing what's under the hood? Browse through the Harness Cloud Cost Management Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/comparison-guide/circleci-vs-harness>

Comparison Guide

Harness

Continuous Integration

VS

CircleCI

Harness Continuous Integration vs CircleCI

Continuous Integration

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

CircleCI

CircleCI is a modern continuous integration and continuous delivery (CI/CD) platform. CircleCI automates building, testing, and deploying software.

600+

2011

315m

Latest funding: Series F \$100 million round of funding in 2021

CircleCI is categorized as:

Continuous Integration

CircleCI is a purpose-built CI/CD solution that focuses on performance and extensibility, providing a large number of Orbs, or plugins, that integrate with common DevOps tools to automate application development. With a core focus on CI, it lacks many native governance capabilities, can be expensive due to hidden costs, and has historically experienced stability issues that make Harness CI a great option for teams looking to modernize how they build, test, and deploy their applications.

What is the difference between Harness CI vs. CircleCI?

[CircleCI vs Harness CI: DevOps Tools Comparison | Harness](#)

Updated

November 30, 2023

â

Free and Paid

Free and Paid

Free and Paid

Free and Paid

SaaS and On-Premise

SaaS and On-Premise

SaaS and On-Premise

SaaS and On-Premise

26,200

1500

26,200

1500

5000+

2500+ (called âOrbsâ)

5000+

2500+ (called âOrbsâ)

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Per user

Per minute for hosted builds

Per user and pipeline minute

Per user

Per minute for hosted builds

Per user and pipeline minute

<yes><yes>

<no><no>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<with><with>

<with><with>

<yes><yes>

<yes><yes>

<no><no>

<no><no>

<yes><yes>

<no><no>

<no><no>

<yes><yes>

<yes><yes>

<no><no>

<no><no>

<yes><yes>

<yes><yes>

<no><no>

<no><no>

<yes><yes>
<yes><yes>
<no><no>
<no><no>
<yes><yes>
<yes><yes>
<no><no>
<no><no>
<yes><yes>
Vault/KMS/3rd party
<with><with>
<with><with>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<no><no>
<no><no>
<yes><yes>
<with><with>
<with><with>
<yes><yes>
<yes><yes>
<with><with>

<with><with>
<yes><yes>
<yes><yes>
<no><no>
<no><no>
<yes><yes>
<yes><yes>
<no><no>
<no><no>
<yes><yes>
<yes><yes>
<no><no>
<no><no>
<yes><yes>
(Linux/Windows/MacOS)

<yes><yes>
(Linux/Windows/MacOS)

<yes><yes>
<yes><yes>
<with><with>
<with><with>
<yes><yes>
<yes><yes>
<no><no>
<no><no>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>

**Please note: Our competitors, just like us, release updates to their products on a regular cadence. We keep these pages updated to the best of our ability, but there are bound to be discrepancies. For the most up-to-date information on competitor features, browsing the competitor's new release pages and communities are your best bet.*

Continuous Integration

Interested in seeing what's under the hood? Browse through the Harness Continuous Integration Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/comparison-guide/launchdarkly-vs-harness>

Comparison Guide

Harness

Feature Flags

VS

LaunchDarkly

Harness Feature Flags vs LaunchDarkly

Feature Flags

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

LaunchDarkly

LaunchDarkly is a Feature Flag Platform that serves over 100 billion feature flags daily to help software teams build better software, faster.

200-500

2014

\$330M

LaunchDarkly is categorized as:

Feature Flags

What is the difference between Harness Feature Flags vs. LaunchDarkly?

We're the best Feature Flag management tool. See for yourself.

LaunchDarkly vs Harness: DevOps Tools Comparison | Harness

Updated

- Primary Users
- Primary Use Case
- SaaS & On-Premises Support
- Simple Developer Onboarding
- Feature Flag Pipelines
- Native CI/CD Support & Pipeline Extensibility
- Config-as-Code & GitOps
- SDK Support
- Automated Feature Verifications
- Flag Lifecycle Management
- Binary/Multivariate Flags
- Flexible Targets (Users, Groups, Regions, Infra)
- Streaming & Polling Support
- Security/Access Control
- Global Governance (OPA Support)
- Audit Logs
- Reporting & Dashboards
- Proxy Service

Developers, Product Managers, Ops, Eng Managers

Manage risk & velocity; Automation; Governance

<yes><yes> **SaaS & On-Prem Supported**

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

All Major Languages

<yes><yes>

<with><with> **Coming Soon**

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes> **RBAC, SSO, More in Progress**

<yes><yes>

<yes><yes>

<yes><yes> **Reporting Across SDLC**

<yes><yes>

Developers, Product Managers

Manage risk & velocity; Experimentation

<no><no> **SaaS & Only**

<no><no>

<yes><yes> **Workflows**

<no><no>

<no><no>

All Major Languages

<no><no>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

<with><with> **Limited to FF Data**

<yes><yes>

Developers, Product Managers, Ops, Eng Managers

Developers, Product Managers

Developers, Product Managers, Ops, Eng Managers

Developers, Product Managers

Manage risk & velocity; automation; governance

Manage risk & velocity; Experimentation

Manage risk & velocity; automation; governance

Manage risk & velocity; Experimentation

SaaS & On-Prem Supported

<no><no>

SaaS Only

SaaS & On-Prem Supported

<no><no>

SaaS Only

<yes><yes>

<yes><yes>

<no><no>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

(called Workflows)

<yes><yes>

<yes><yes>

(called Workflows)

<yes><yes>

<yes><yes>

<no><no>

<no><no>

<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
RBAC, SSO, more in progress

<yes><yes>
<yes><yes>
<yes><yes>
RBAC, SSO, more in progress

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>
Reporting across SDLC

<with><with> Limited to FF data

<yes><yes>
Reporting across SDLC

<with><with> Limited to FF data

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

SaaS & On-Premises Support:

LaunchDarkly only offers a SaaS version of their product. Harness supports both SaaS and on-prem deployment models.

Primary Users:

Delivering software is a process that starts at building an application. In the same way, feature flags should provide value to teams at each phase of the software delivery lifecycle. It's not just about shipping code faster and creating improvements for users; feature flags should also make daily operations easier and provide visibility. Harness Feature Flags was built to support all of these use cases across Developers, Product Managers, Operations, and Engineering Managers.Â

The focus is unleashing the power of feature flags for software delivery teams holistically. And while Harness focuses on a developer-first experience, the same workflows and processes are built to support software delivery teams where they struggle the most. Hint: it's not just shipping features.

Simple Developer Onboarding:

When building Feature Flags, Harness focused heavily on the developer experience - developers are, after all, the true arbiters of feature flags. Our intent was to make sure that the setup process is fast and intuitive, and that teams can get started and see value from feature flags right away. While LaunchDarkly provides a robust onboarding experience, they are focused more on the operational users of feature flags, such as product managers, rather than the developers themselves.

Feature Flag Pipelines:

In Harness, this feature is called Pipelines, and LaunchDarkly has a comparable feature called Workflows. At its core, this capability allows users to create a custom workflow for a flag that can be applied to multiple flags. Where they differ is in their versatility.

LaunchDarkly's Workflows allow for a user to create rules and steps to take actions against a flag, but we don't see the ability to create automated or reusable templates. What Harness offers is a visual drag-and-drop pipeline builder that supports hundreds of integrations (compared to 34 from LaunchDarkly), as well as fully automatable and repeatable pipelines - and they integrate with CI/CD.

This feature has always been the big differentiator for LaunchDarkly, and it's exciting to see this concept expanded on by Harness with its native extension into - and context from - CI/CD. This unified pipeline extensibility is unique in the space, and LaunchDarkly isn't there yet.

Native CI/CD Support & Pipeline Extensibility

LaunchDarkly, as a third-party tool, is completely abstracted from your CI/CD process, which perpetuates the view that feature flags are an ancillary step. Ideally, this view would be abolished. At Harness, we view feature flags as a crucial part of the CI/CD pipeline - and as such, it becomes a natural step in the everyday workflow of development teams and is integrated into CI/CD as a unified pipeline. This allows for users to get visibility into any change from code to release, and create standardized and automated processes across all stages of the SDLC.

As part of the Harness platform, users also can rest easy knowing that they can get a full audit log, manage teams and security like RBAC, share environments and APIs, and see cross-platform analytics. Of course, all of this is available for just Harness Feature Flags, and its power is more pronounced on the platform.

Config-as-Code & GitOps:

You'll notice that the primary users for both tools are Developers and Product Managers. Where Harness differs in its approach is that it's built to be developer-first. It's important to meet developers where they are and work within their processes. With Harness, teams have the ability to create pipelines visually and combine that with a pipelines-as-code or config-as-code approach; visual pipelines create YAML files that can be changed to reflect back into the visual pipeline, and these can automatically be synced with Git so changes are reflected immediately and without added effort, all from within our platform.

SDK Support:

Harness and LaunchDarkly support all major languages, both server-side and client-side. However, here is where we must admit that LaunchDarkly got a headstart. They provide support for many more SDKs than Harness does today (though we're catching up quick). Both provide OSS for their SDKs.

Automated Feature Verifications:

Harness pioneered the concept of Continuous Verification in the software delivery space. Having battle-tested the capability over years with hundreds of customers, the ability to have Harness automatically monitor deployments and take appropriate rollout or rollback options is extended into Harness Feature Flags. A unique capability in the world of feature flags, users can use the ML-based automated verification feature to have Harness automatically verify the health of live features and take appropriate remediation action should things go wrong - it obviates the need for users to manually monitor and make changes during a feature rollout.

Security/Access Control:

The key difference between Harness and LaunchDarkly is the extensibility of security and access control to more than just feature flags. With the native Harness support for CI/CD, users can extend the same security/access control profiles end-to-end across SDLC, rather than having to learn multiple tools and create those controls from scratch. By simplifying this process end-to-end, users are able to focus on using the tools rather than configuring them.

At Harness, we take security extremely seriously, as evidenced by our DevSecOps approach. We took the same views with Feature Flags, ensuring RBAC and SSO were available from inception, with more controls coming soon.Â

Governance & Compliance:

Harness takes governance and compliance seriously, providing CI/CD users with fine-grained RBAC, audit trails, many ways to manage

secrets, and more. We've taken that same approach with Feature Flags and are investing heavily in the area, letting teams build rules and processes, and helping automate cleanup and flag management. Harness also supports Open Policy Agent (OPA), enabling Enterprise-scale governance as-code across Feature Flags and the rest of the Harness platform. Harness is the only feature management solution to offer this.

Reporting & Dashboards:

Since LaunchDarkly is purely a feature flags tool, reporting and dashboards are limited to data within the tool itself - understandably so. Harness, however, allows users to link Feature Flags data to the rest of the software development lifecycle, and as such, will provide reporting and dashboards for all products in the same easily-accessible place. Harness is actually the only platform on the market that brings together analytics across the whole SDLC into one place - one tool to rule them all, one tool to bind them.

But what if you're using only feature flags and don't care about CI/CD? Harness provides the most complete visibility into feature flags of any tool on the market. Any bit of data that's related to your feature flags use and operation, you can visualize in Harness to create custom dashboards to show exactly what you need at any time.

**Please note: Our competitors, just like us, release updates to their products on a regular cadence. We keep these pages updated to the best of our ability, but there are bound to be discrepancies. For the most up-to-date information on competitor features, browsing the competitor's new release pages and communities are your best bet.*

Try Harness For Free

Feature Flags

Interested in seeing what's under the hood? Browse through the Harness Feature Flags Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/comparison-guide/datadog-vs-harness-ccm>

Comparison Guide

Harness

Cloud Cost Management

VS

DataDog

Harness Cloud Cost Management vs DataDog

Cloud Cost Management

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

DataDog

Datadog is the essential monitoring and security platform for cloud applications. Their SaaS platform integrates and automates infrastructure monitoring, application performance monitoring and log management to provide unified, real-time observability of customers' entire technology stack.

3,200

2010

Public Company

Datadog is a publicly traded company with an IPO in 2019.

DataDog is categorized as:

Cloud Monitoring and Observability

What is the difference between Harness CCM vs. DataDog?

Datadog is a recognized leader in cloud infrastructure and performance monitoring, but their cloud cost reporting is in its infancy, and lacks support for multi-cloud, Kubernetes or any of the cost savings features that you can find in Harness today.

DataDog vs Harness: Cloud Cost Management Tool Comparison

Updated

November 30, 2023

SaaS and Self-Hosted

SaaS

AWS, Azure, and GCP

AWS and Azure

<with><with>

Kubernetes Only

<no><no>

Percentage Cloud Spend

Fee Per EC2 Instance

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<with><with>

AWS Only

<yes><yes>

<no><no>

Coming soon

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<no><no>

<yes><yes>

<with><with>

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<yes><yes>
<yes><yes>
<no><no>
<yes><yes>
<yes><yes>

Summary:

DataDog has been a go-to vendor for observability across on-premises and cloud infrastructures, with services that include log management, application performance monitoring (APM), as well as monitoring for security, network, users and more. Given this, it seems a natural extension for DataDog to add cloud cost monitoring to their portfolio. True to their observability nature, DataDog chose to focus on cloud visibility features in their first release and lack any features that proactively manage cloud costs, or even alert users to cost anomalies as they occur.

Cloud cost management is so much more than reporting costs, and if you want to save money, not just report on it, then Harness Cloud Cost Management is the way to go. True to our nature, we focus on automation to increase the efficiency of not only cloud costs, but also of your engineering teams. We create cost savings opportunities that simply don't exist through manual efforts, saving you up to 70% on your cloud spend.

Multi-cloud and Kubernetes Visibility:

The majority of enterprise's have cloud infrastructure that spans across multiple cloud providers, allowing them to optimize their cloud use to fit their needs. Cloud native deployments leverage shared cluster resources for the best scalability and resiliency, with Kubernetes found in more than 60% of deployments today. Harness knows this well, and provides out of the box cost perspectives that show you all of your multi-cloud costs in a single view, as well as granular Kubernetes shared cost reporting.

Datadog only supports AWS (as of the date of this report), and has no support for Kubernetes, which means their customers only see a single, monolithic cost figure for all their Kubernetes infrastructure. All costs are displayed by tag names, which is not exactly user friendly or easy to parse. There's no way to view cloud costs in a context that makes sense for your business, nor surface how your cloud spend is tracking to your budgets.

Cost Optimization:

Datadog can tell you that you have \$1M in monthly cloud spend. Is that good or bad? Can you make it better? And why did my monthly cost just jump a half a million last week? These are important questions that you need answers to, and Datadog doesn't have any way to answer them for you. While you can drill down into the tag level for cost information, there's no support for reserved instances or savings plans, no recommendations for right-sizing, and most importantly, no cost anomaly alerting in case a test cluster is left running. **Simply put, they can't help you save on your cloud spend.**

With Harness, you get a full featured cost optimization solution with advanced automation to save you up to 70% on your cloud costs overall, or up to 90% when you deploy Cloud Autostopping to proactively manage idle cloud resources. We give you real-time, granular cost anomaly alerts so that you can stop unexpected cloud spend in its tracks.

Cloud Cost Management

Interested in seeing what's under the hood? Browse through the Harness Cloud Cost Management Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/secure-continuous-delivery>

Relativity Selects Harness Over Spinnaker for Secure Continuous Delivery

Relativity selects Harness over Spinnaker resulting in massive security improvements.

About Relativity

Relativity makes software to help users organize data, discover the truth, and act on it. Their e-discovery platform is used by thousands of organizations around the world to manage large volumes of data and quickly identify key issues during litigation, internal investigations, and compliance projects.

Accelerating Developer Velocity Through Azure Microservices

Corey Wagehoft, Lead Systems Engineer at Relativity, led an internal initiative to reduce costs and increase developer velocity by migrating from traditional legacy windows services to container-based microservices running in Microsoft Azure.

âDeployments in the past typically took 8 hours and were manually orchestrated by a team of 3 engineers,â said Corey. âIn addition, we had a 50% change failure rate which was dropping internal confidence and causing downtime.â

Corey knew Relativity needed to transform its deployment pipelines so it could streamline the onboarding of their microservices in the Azure cloud. His team also needed a way to support the legacy Windows services throughout this migration as well.

Managing user data meant that Corey and his team had strict security requirements, which limited the types of tools that could be onboarded. âWe had a cultural tendency to build our own tools internally,â said Corey.

Evaluating Spinnaker and Harness

Coreyâs team first looked at Spinnaker, but quickly determined the open-source solution wouldnât work for their needs.

âIn order to get partial functionality out of Spinnaker, we would need to dedicate an engineer to the project for an entire year,â said Corey.

Relativity uses Linux and containers throughout their tool stack and Spinnaker lacks native Microsoft integrations. Corey also determined that Spinnaker would require large adjustments to comply with Relativityâs security standards.

After a short evaluation, Coreyâs team picked Harness CD as-a-Service for the following reasons:

- Ability to support native Windows services in addition to Azure microservices
- Native integration with Azure Key Vault for secrets management
- Secure communication via the Harness delegate architecture
- CD as-a-Service vs. self-managed solution that requires dedicated engineering resources

Secure Continuous Delivery with Harness

In just 3 months, Corey and his team were able to increase deployment velocity by 30X using Harness.

Harness integrated with Relativityâs entire toolchain that consisted of Jenkins for CI, HashiCorp Terraform for Infra provisioning, New Relic for APM, Splunk for Log Analytics, and Prometheus for Infra monitoring.

His team also leveraged Harness Pipeline templates to create gold standards so development teams could deploy consistently in a repeatable manner no matter whether the artifact was a Windows service or Azure microservice.

Harnessâ native integration with Azure Key Vault (a feature request from Relativity during their Harness trial) allowed Relativity to stay ISO and SOC 2 compliant while also preparing the team for FED ramp certification.

Finally, teams were leveraging Continuous Insights to help them make data-driven decisions around pipeline flow, deployment velocity, change failure rate and MTTR.

80% Reduction in Change Failure Rate

With Harness, Coreyâs team achieved the following results:

- Reduced deployment time by 98% from 8 hours to just a few minutes

- Reduced deployment effort by 99% from 3 full-time engineers for 8 hours to 1 engineer for minutes.
- Reduced change failure rate by 80% from 50% to 10%.
- Saved \$150,000 instead of requiring dedicated engineering resources to manage Spinnaker

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/comparison-guide/cloudcheckr-vs-harness-cloud-cost-management>

Comparison Guide

Harness

Cloud Cost Management

VS

CloudCheckr

Harness Cloud Cost Management vs CloudCheckr

Cloud Cost Management

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration

CloudCheckr

The CloudCheckr CMx platform proactively analyzes cloud infrastructure to provide visibility and intelligence to better manage costs, make infrastructure more secure and in compliance, and optimize resources in use.

10000+

2011

Acquired

CloudCheckr was acquired by NetApp in 2021, and is part of the Spot by NetApp portfolio

CloudCheckr from Spot by NetApp is categorized as:
Cloud Cost Management

What is the difference between Harness Cloud Cost Management vs. CloudCheckr?

CloudCheckr from Spot by NetApp is a cloud cost management that is focused on billing / security features needed for MSPs, but it lacks support for Kubernetes, multi-cloud cost visibility, and any of the intelligent automation features found in Harness Cloud Cost Management.

CloudCheckr vs Harness: Cloud Cost Management

Updated

November 30, 2023

SaaS and Self-Hosted

SaaS

AWS, Azure, and GCP

AWS, Azure

<with><with>
Kubernetes Only

<no><no>

Percentage Cloud Spend

Percentage Cloud Spend

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

<no><no>

Coming soon

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>
<yes><yes>
<with><with> 30 days only
<yes><yes>
<yes><yes>
<yes><yes>
<with><with> No Kubernetes Support
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<yes><yes>
<yes><yes>
<no><no>
<no><no>
<no><no>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Summary:

CloudCheckr from Spot by NetApp (itâs a mouthful, isnât it?) is very focused on MSPs, and as a result has great billing / chargeback and security features. If all you want is billing management, CloudCheckr is the way to go. However, that billing and security focus came at the cost of a truly usable cloud cost management tool. CloudCheckr has no support for Kubernetes, which drives a large majority of cloud deployments today. And while they claim support for GCP on the website, one look at their documentation shows that there are no user docs to support that claim. Although CloudCheckr is now part of the Spot by NetApp portfolio, it remains a standalone tool that lacks any automation for cloud cost savings.

If you are looking for a complete solution for cloud cost management that includes true support for multi-cloud and Kubernetes, Harness Cloud Cost Management is the right solution for you. We give you deep insights into Kubernetes costs and cluster management, automated cost savings features, and excellent business intelligence and reporting capabilities. All the features your FinOps / DevOps / Engineering teams need without the manual overhead they donât.Â

Multi-cloud costs:

Look at any state of the cloud report today, and youâll find that the majority of enterprises leverage more than one cloud provider, with AWS, Azure and Google Cloud Platform accounting for the lionâs share of the market. Supporting all 3 the providers is the minimum bar for cloud cost management, and being able to provide all-in-one views of multi-cloud costs is table stakes for users. Harness CCM gives you a complete view of all your multi-cloud costs, in a business perspective that makes sense for your business.Â

CloudCheckr does not have the ability to show you AWS and Azure costs in a single dashboard, and in fact, youâll need to log into separate tools / accounts to see costs for each provider. Their claim to support GCP is not backed up by their documentation, which only provides instructions for AWS and Azure.Â

Kubernetes Support:

Modern SaaS infrastructure relies on containers and Kubernetes for easy scalability and flexible deployment models. However, shared cost allocation is a challenge unless you have the tools in place to dig into costs by workload / node / pod / label, etc. Harness CCM provides detailed cost analysis for Kubernetes clusters, at a very granular level that can be tailored for your business. We also deliver automated cluster cost management features such as Cluster Orchestrator for EKS that can save our customers up to 90% on cloud costs.Â CloudCheckr has no support for Kubernetes at all.

Forecasting:

With cloud costs, itâs not enough to know what your costs are today, itâs important to see where you are expected to land at the end of the month / quarter / year to understand if you are on target to meet budget expectations. Accurate, long term forecasts are crucial when negotiating savings plans contracts to optimize your cloud spend. Thatâs why Harness CCM gives you accurate 12 month forecasts, while CloudCheckr is limited to 30 day forecasts.

Reporting:

Seeing and understanding your cloud cost trends is important, so that you can see your cloud usage over time. Seems like an easy thing to do, and just about every cloud cost vendor can provide you a report that gives month over month cloud costsâ|except for CloudCheckr. If you want historical reporting, go with Harness CCM.

Cloud Cost Management

Interested in seeing what's under the hood? Browse through the Harness Cloud Cost Management Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/replaced-spinnaker>

MakerBot Replaced Spinnaker & Saved \$275,000

Learn how MakerBot replaced Spinnaker and reduced CI/CD costs by \$275,000

About MakerBot

MakerBot believes thereâs an innovator in everyone. As a global leader in 3D printing, they set the standard in reliability and ease-of-use by providing effective solutions for every stage of the 3D printing process. From the start, they have redefined the possibilities with 3D printing for users across all industries.

Open-Source CI/CD Dilemma

MakerBotâs software delivery process kept Erik Ahrend up at night, literally. As the Lead Cloud Architect, Erik maintained MakerBotâs Spinnaker pipelines. Erik spent three hours a day troubleshooting deployment issues and often was woken up in the middle of the night to fix broken deployments. The total cost of Erikâs time added up to roughly \$118,000 a year.

Spinnaker pipelines and APIâs required advanced knowledge to create and edit. Erik was the only one with enough experience to fix deployment issues. MakerBot hired Armory to help manage their pipelines. Unfortunately, MakerBotâs deployment issues persisted.

Deployments had an unexplainable 48-hour lag between uploading an image and reaching production. Most developers found it easier to manually update Helm than wait for Spinnaker. Erik needed to fix and simplify MakerBotâs software delivery process.

The Cost of Custom Governance

In addition to simplifying deployments, Erik needed to add capabilities. The pipelines lacked basic integrations with APM tools like DataDog, which slowed down the verification process. The pipelines also lacked governance. Without RBAC controls or audit trails, Eric feared for the safety of MakerBotâs applications.

Erik planned to spend 6 months coding these two features and adding them to MakerBotâs pipelines. The project would require at least two developers and cost MakerBot \$150,000 in developer effort.

Realizing the exorbitant costs in front of him, Erik decided to search for a company that could provide out of the box solutions.

Harness CI/CD Provides Priceless Confidence

Erik turned to Harness for that solution.

Harness created a self-service deployment culture. Developers are able to copy Harnessâs pipeline as code into whatever new pipeline they need, and Erik doesnât have to spend 3 hours a day babysitting deployments. Harness also provided MakerBot with advanced RBAC controls, Datadog and Jira integrations, and templated canary deployments. These capabilities reduced MakerBotâs security concerns and increased deployment confidence.

In just 4 months using Harness, Makerbot is deploying ~15 times per day with a very low failure rate of only 8%. Most importantly, Harness was able to provide Erik with some much-needed headspace.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/ruckus-cd-gitops>

Ruckus Scaled Kubernetes Continuous Delivery With GitOps

Learn how Ruckus scaled Kubernetes Continuous Delivery using Harness GitOps

About Ruckus Networks

Ruckus Networks is redefining connectivity by bridging the digital divide and connecting people around the world. Ruckus works with channel partners, OEMs, and strategic partners to deliver ubiquitous connectivity with our access points, switches, and cloud services.

Migrating From Monolith VMs To Kubernetes Microservices

At Ruckus Networks, past monolith application environments (e.g. Dev, QA, Staging, Prod) could take 30 minutes or more to spin up, with DevOps and development teams left waiting.

As a result, horizontal scaling and elasticity of services running in production became a major challenge.

In addition, deployment pipelines were service specific, with little to no reuse of logic or configuration across teams.

A decision was therefore made to migrate to a cloud-native Kubernetes-based microservices architecture, where environments could be provisioned in seconds, and horizontal auto-scaling was embedded natively to the run-time environment.

As a result of this migration, 40+ new Kubernetes microservices would all require new deployment pipelines and onboarding by the DevOps teams.

âOnboarding a new Kubernetes Application could take our DevOps engineers one week,â said Raj Zala, Director of Cloud Engineering at Ruckus Networks.

Build Vs. Buy For Continuous Delivery?

To solve this onboarding challenge, Raj and her DevOps team looked at several options:

Leveraging Harness GitOps And Pipelines-As-Code

One of the key selling points of Harness was to give DevOps and developers self-service Continuous Delivery capabilities, meaning they could empower, automate and scale their software delivery process from dev all the way to production without waiting more than a few minutes.

The second advantage was the ability for the DevOps team to build and reuse pipelines across their portfolio of new Kubernetes microservices and engineering teams.

Our DevOps team now creates pipeline templates in Harness, and developers instantiate these with their own service parameters, said Raj.

Harness GitOps allows DevOps/developers to manage pipeline, service, environment, Helm, and Kubernetes configuration all in one place as code. This means all pipelines can be version controlled and managed like code in any Git repository.

Reduced Time-To-Market By 80% & \$250k Savings

As a result of implementing Harness, Ruckus Networks saw the following benefits:

- Onboarding time of new Kubernetes services dropped 80% from one week to same business day
- Onboarding effort for the DevOps team went from 38 weeks to 38 days
- Saved \$250,000 in DevOps costs to build own MVP CD platform
- Reduced time-to-market of CD process by 92% from 6 months to 2 weeks

â

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform®

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/smu-rollback-deployment-time>

SMU Reduces Rollback & Deployment Time by Over 80%

SMU becomes a deployment powerhouse by leveraging Harness.

About

SMU, a prestigious retail company with stores in Chile and Peru, has grown exponentially in only a few short years. SMU has positioned itself as a leader in the supermarket and convenience store sector. As of today, it operates the most grocery and convenience stores in

Chile, and it also ranks third in terms of revenue for 2021. Likewise, it is the Chilean company with the largest geographic coverage and the only supermarket chain present in all 16 regions of the country.

The company operates its business through the Unimarc, Unimarc.cl, Super 10, Mayorista 10, Alvi, and OK Market brands. Additionally, it operates in Peru through two brands, Mayorsa and Maxiahorro.

A Solid Start With a Single Glaring Flaw

Through a combination of engineering talent, Weaveworks Flux, and GitOps strategies, SMU managed to create a deployment process that allowed the company to deploy at an acceptable rate. This deployment mechanism wasn't a pipeline, it was based on the pull action. The only complaint developers had with using the solution was it was fairly difficult to decode deployment errors. But this wasn't a big enough concern to prompt a tooling change.

This changed when Javier Vera Duran discovered that SMU lacked a deployment verification process and had a long rollback time. As the Lead of Technology and DevOps, Javier had a responsibility to keep SMU's application as stable and secure as possible. Without a verification process and with a long rollback time, SMU was exposed to serious downtime risk. This risk prompted Javier to look for better solutions.

SMU initially evaluated open-source tools to supplement the deployment process they had already created. But they couldn't find a solution that would allow them to keep the GitOps methodologies they had already adopted.

SMU turned to Harness for a better deployment solution.

Harness Provides Unexpected Benefits

SMU now deploys all of their Kubernetes microservices using Harness. The team quickly onboarded their services onto Harness with minimal effort. Onboarding took approximately an hour per service.

The new ecosystem that was created allows SMU to do Canary deployments with automatic rollbacks. In tandem, these two features have reduced rollback time by 83% from 30 minutes to 5 minutes.

Unexpectedly, the team was also able to drastically reduce deployment time. The original process took roughly an hour, but with Harness it takes 5 minutes. That's a 98% reduction in effort and time.

Thanks to Harness, SMU will be able to safely scale its online business.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Iterable Triples Deployment Speed & Reduces Downtime Risk by 85%

Find out how Iterable deployed faster and more securely.

About

Iterable is the growth marketing platform that enables brands to create, execute and optimize campaigns across email, push, SMS, in-app, and more with unparalleled data flexibility. An integrated, cross-channel solutionâbuilt for marketers, trusted by engineers, designed with intelligence.

From âGood Enoughâ to âExtremely Painfulâ

When Tenzin Wangdhen started as an SRE at Iterable, he inherited a software delivery process in disarray.

Iterable had designed a delivery workflow for a simple application consisting of a few servers. But as the applicationâs complexity increased, so did deployment complexity. A few servers turned into over 1000 servers, and the once-simple delivery workflow required a dedicated developer to spend 8 hours a week maintaining CD scripts and tools. Delivery complexity also caused stability and quality issues.

The deployments occurred in weekly batches. Developers deployed from their laptops and declared changes in a Slack channel. This typically took a full day and unintentionally allowed developers to work over each other.

40% of the deployments failed and it took roughly 4 hours to roll-back. Tenzin needed a deployment solution to help increase speed and stability.

Moving Faster Without Breaking Things

Tenzin was able to deploy 15 services in the first 15 days using Harness and gave developers the power to deploy by themselves without worrying about incongruencies.

Harness helped Iterable:

- Increase velocity 3x from 150 deployments per month to 537
- Decrease downtime risk by 85%
- Reduce change failure rate from 40% to 20%
- Reduce rollback time from 4 hours to 20 minute
- Decrease deployment effort by 89% from multiple day global deployments to 45 min

Harness helps Iterable manage reputational risk while increasing developer productivity.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/saves-108k-deployment-effort>

Single Digits Saves \$108k in Deployment Effort

Single Digits made their migration easy with Harness. See how you can too.

About

Single Digits offers complete guest, resident, and customer internet and connectivity solutions, including enterprise-class software, network design, engineering, professional services, ongoing maintenance, network monitoring, multilingual support, revenue reporting, and authentication tracking.

From Legacy to Microservices

Single Digits was embarking on a journey to modernize its platform from legacy applications to an event-driven platform based on microservices. The team hoped that a microservice architecture would lead to greater deployment velocity and less downtime.

As it stood, the legacy platform was deployed using Jenkins jobs and custom shell scripts. This deployment frankenstein allowed Single Digits to deploy their legacy application once every two weeks. The deployment itself took 6 hours, occurred in the middle of the night, and was spread across three days. This was the only way the company could mitigate downtime risk.

As the Director of Cloud Architecture, Clint Nelissen had already put a considerable amount of effort into developing Single Digits' CI/CD strategy. Knowing that the migration would necessitate a new deployment methodology, Clint wanted to make sure his team didn't have to rebuild everything from scratch.

Jenkins was automatically thrown out as an option because of its lack of microservice deployment flexibility. Clint also evaluated CircleCI and Codefresh, but both CI tools were dismissed because the team didn't want to throw out their existing CI solution.

Clint needed a solution that would assist in the migration and wouldn't override all of their pre-existing hard work.

Harness Makes Modernization Easy

Harness helped Single Digits transition to a modernized microservice architecture. Ten new microservices were created and onboarded within the first three months of using Harness. Since then, more than 125 services have been onboarded with more added each month.

Harness has also helped Single Digits reduce deployment time from 6 hours to 20 minutes, which saves them \$108,000 annually in deployment effort.

Every month, companies are deciding to migrate to a microservice architecture and every month, companies are hitting migration roadblocks. Check out Harness to find out how we can ease your migration burden.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/reduce-kubernetes-costs>

Relativity Reduced Kubernetes Cloud Costs by Millions in 5 Months

Learn how Relativity democratized cloud cost data across engineering and product teams with Harness Cloud Cost Management.

About Relativity

Relativity makes software to help users organize data, discover the truth, and act on it. The platform is used by thousands of organizations around the world to manage large volumes of data and quickly identify key issues during litigation, internal investigations, and compliance operations with RelativityOne and its newest offering Relativity Trace. Relativity has over 180,000 users in 40+ countries from organizations including the U.S. Department of Justice, more than 70 Fortune 100 companies, and 198 of the Am Law 200.

Accelerating Deployment Velocity & Reducing Cloud Spend

Earlier this year Relativity kicked off an internal initiative to increase deployment velocity and reduce cloud spend as part of their cloud migration to Windows Azure.

Previously, finance initiated cost savings exercises directly with product managers and engineering teams. Cloud costs were âbubbledâ together into a high-level aggregate view across teams, which made it challenging for an individual product or engineering team to understand their costs.

This lack of visibility into cloud costs by application or microservice meant that engineers struggled to identify the real utilization and idle cost of the resources they were requesting in the Azure cloud, specifically relating to their Kubernetes clusters.

âWe were seeing over-provisioning of 30-40% of our microservices and Kubernetes Clusters,â said Corey Wagehoft, Lead Systems Engineer at Relativity.

Cost Reductions in the First Month

Relativity participated in the early beta of Harness Cloud Cost Management â a new cloud cost management solution that empowers engineering and DevOps teams with unique cloud cost visibility so they can proactively manage cloud spend.

âDuring the first 30 days of implementation, we saw a noticeable change in our cloud spend across our engineering teams, with six-figure annualized savings,â said Shelby Lewin, Technical Product Manager at Relativity. âNot only did we save cost, but our engineering team also sped up two months of work on our roadmap for RelativityOne.

With Harness Cloud Cost Management, Relativity was able to reduce cloud spend by 30-40% for a new microservice in 30-days which represents a 6-figure annualized savings.

The biggest win for Corey and Shelby was unpacking the true cost of their SaaS tenants and customers running on the Relativity platform.

5 Months Laterâ!

Fast forward five months, and Relativity has saved millions using Harness Cloud Cost Management.

Product and engineering teams at Relativity are now proactive with self-service access to their own cloud spend for application workloads so they can take ownership of cloud costs.

With Harness Cloud Cost Management, teams can view cloud cost by:

- Application
- Microservice
- Environment

- Cluster
- Namespace
- Workload
- Node and Pod

In addition, teams can set pro-active budgets and alerting associated with their application, microservice and clusters.

By analyzing the low utilization of Kubernetes pods running on cloud infrastructure (nodes), Relativity was able to double the density of Kubernetes pods per node from 40 to 100, and in some large nodes, 100 pods per node. The cost impact of this was a reduction in Kubernetes costs of 40% per day.

Finally, with Harness Cloud Cost Management, Relativity was able to defer 1 FTE costs associated with building an in-house cost management tool for product and engineering teams.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/comparison-guide/sto-vs-gitlab-ultimate>

Comparison Guide

Harness

Security Testing Orchestration

vs

GitLab Ultimate

Harness Security Testing Orchestration vs GitLab Ultimate

Security Testing Orchestration

500-1000

2016

\$425M

Harness is categorized as:

- Continuous Integration
- Continuous Delivery
- Cloud Cost Management
- Cloud Cost Optimization
- Feature Flags
- Service Reliability Management
- Security Testing Orchestration
- Chaos Engineering
- Software Engineering Insights

GitLab Ultimate

GitLab Ultimate is a DevOps platform with advanced security testing features built-in to provide actionable vulnerability findings to developers while helping security pros manage remaining vulnerabilities through resolution.

â

1,200

2011

413m

GitLab is categorized as:

- Continuous Delivery
- Continuous Integration
- Static Application Security Testing
- Dynamic Application Security Testing

What is the difference between Harness Security Testing Orchestration vs. GitLab Ultimate?

Harness STO Vs. GitLab Ultimate

Updated

November 30, 2023

- SaaS & On-Premises
- Main Users
- Integrations With Leading Application Security Scanners
- Integrates With Any CI/CD Tool
- SCA Tool IntegrationÂ
- SAST Tool IntegrationÂ
- DAST Tool IntegrationÂ
- Container Scanning
- Fuzz Testing Support
- Orchestration of Security Testing with Scanners
- Normalization and Deduplication of Scanner Results
- Automated Prioritization of Vulnerabilities
- Security Guardrails integrated with CI/CD pipelines
- Security Pipelines as Code
- Security Pipeline Visual Builder
- Policy-as-code Pipeline Governance
- Customizable Security Policies
- Custom Vulnerability Reporting
- Security Exemption Tracking
- Vulnerability Visibility Across All Services
- Aggregated Vulnerability Management
- Jira Ticket Integration
- Fine-Grained Role-Based Access Control
- Audit Trails
- Unified Software Delivery Platform

<yes><yes>

Developer, DevOps, DevSecOps

<yes><yes> 30+ Integrations

<yes><yes>

<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
Coming SoonÂ
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
Coming Soon
<yes><yes>
<yes><yes>
<yes><yes>
Coming Soon
Yes, fully customizable
<yes><yes>
<yes><yes>
<yes><yes>
Developer, DevOps, DevSecOps
<yes><yes> limited
<no><no>
Yes, limited
Yes, limited
Yes, limited
Yes, limited
<yes><yes>
Yes, partial
<with><with>
<with><with>
<yes><yes>
<no><no>
<no><no>
<with><with>
<with><with>

<no><no>
<no><no>
<with><with>
<yes><yes>
<yes><yes>
Yes, limited
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
Developer, DevOps, DevSecOps
Developer, DevOps, DevSecOps
<yes><yes> 30+ Integrations
<yes><yes> limited
<yes><yes>
<no><no>
<yes><yes>
Yes, limited
<yes><yes>
Yes, limited
<yes><yes>
Yes, limited
<yes><yes>
Yes, limited
Coming Soon
<yes><yes>
<yes><yes>
Yes, partial
<yes><yes>
<with><with>
<yes><yes>
<with><with>
<yes><yes>
<yes><yes>
<yes><yes>
<no><no>
<yes><yes>
<no><no>

<yes><yes>

<with><with>

<yes><yes>

<with><with>

Coming Soon

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<with><with>

<yes><yes>

<yes><yes>

Coming Soon

<yes><yes>

Yes, fully customizable

Yes, limited

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Summary:

While Harness and GitLab seem to share many of the same capabilities across their software delivery platforms, one major difference is that Harness takes a modular approach. This means that individual modules can be used and integrated with other solutions as part of a DevOps toolchain. In contrast, with GitLab users must purchase the complete solution with the Ultimate license package.

Integrates With Any CI/CD Tool:Â

Harness STO operates independently or integrated with any CI/CD solution.Â

GitLab Ultimateâs Advanced Security Testing features must be purchased with the full GitLab platform. GitLab Advanced Security Testing does not integrate with other CI/CD solutions.

Normalization and Deduplication of Scanner Results:Â

A challenge with shift-left security is that developers can be subjected to additional workload of scanner result analysis. This workload grows with every scanner execution performed by a pipeline and can take hours for every pipeline execution.Â

Harness STO ingests the output from all scanners, then automatically normalizes, deduplicates, and creates a prioritized list of vulnerabilities to remediate. This saves developers hours of manual analysis work.

GitLab Advanced Security Testing provides scanner output without any additional analysis, placing that workload on the developers.

Automated Prioritization of Vulnerabilities:Â

While itâs important to know all application vulnerabilities, itâs more important to know which vulnerabilities should be prioritized based on their severity. This can be difficult for developers to assess when they have multiple application security scanners running in their CI/CD pipelines. Each scanner provides results in different output formats that need to be looked at individually and then manually merged.

Harness STO solves this problem by automatically merging the output from all scanners and creating a unified prioritization of all vulnerabilities.

GitLab security does not provide a prioritized vulnerability list across all scanners.

Security Pipelines as Code:

Dedicated security pipelines offer a way for any CI/CD solution to invoke a robust security scanning process.Â

Harness STO provides application security pipelines that can be configured using YAML. These configurations are automatically updated using a bidirectional sync between Harness and Git.

GitLab Advanced Security Testing does not offer a stand-alone security pipeline solution.

Security Pipeline Visual Builder:

Dedicated security pipelines offer a way for any CI/CD solution to invoke a robust security scanning process.Â

Harness STO provides application security pipelines that can be configured via a graphical UI. This makes it easy for anyone in an organization to build new security pipelines to ensure application security scanning is conducted via CI/CD pipelines.

GitLab Advanced Security Testing does not offer a stand-alone security pipeline solution.

Custom Vulnerability Reporting:

Most organizations want to see vulnerability reports in formats that are customized for their unique requirements.Â

Harness STO offers out-of-the-box reports, as well as fully customizable reporting capabilities.

GitLab provides out-of-the-box reporting, but at this time, there are no options for customization.

Security Exemption Tracking:

Security exemptions management is an integral component of managing security testing outcomes. Â STO offers a common venue for security practitioners and developers to collaborate and actively manage security exemptions. Security findings often contain a mix of issues. Some need immediate attention. Some will be false positives or won't apply to specific product scope or mode of operation. In some instances, there will be complex factors in remediating security issues and need additional planning. To effectively manage these different scenarios, security exemption management will be vital and can be fashioned in a way that fits your organizational needs via STO.Â

Gitlab offers alternative approaches to manage security findings, but it does not support security exemption management.

Security Testing Orchestration

Interested in seeing what's under the hood? Browse through the Harness Security Testing Orchestration Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/comparison-guide/vmware-cloudhealthtech-vs-harness>

Comparison Guide

Harness

Cloud Cost Management

VS

CloudHealth by VMware

Harness Cloud Cost Management vs CloudHealth by VMware

Cloud Cost Management

500-1000

2016

\$425M

Harness is categorized as:
Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

CloudHealth by VMware

CloudHealth by VMware delivers intelligent insights that help you optimize costs, improve governance, and strengthen your cloud security posture.

37,000

2012

Acquired by VMware (2018)

CloudHealth was acquired by VMware in 2018. VMware is currently under acquisition by chip maker Broadcom. CloudHealth by VMware is being rebranded as VMware Aria Cost powered by CloudHealth

Acquired

CloudHealth is categorized as:

Cloud Cost Management

What is the difference between Harness Cloud Cost Management vs. CloudHealth by VMware?

CloudHealth by VMware was built for business intelligence, but it is complex, and lacks the deep Kubernetes support or any of the advanced automated cloud cost savings features found in Harness Cloud Cost Management.

CloudHealth by VMware vs Harness: DevOps Tools Comparison

Updated

November 30, 2023

- SaaS
- Audience & Primary Users
- Finance
- Cloud Support
- Time Granularity
- Tag Management Required
- Hybrid Cloud Visibility
- Cloud Account Visibility
- Cloud Region Visibility
- Application Visibility
- Microservice Visibility
- Environment Visibility
- Non-Cluster Visibility (Services)
- Cluster Visibility (Kubernetes/ECS)
- Root Cost Analysis (3 Metrics)
- Cloud Event Correlation (e.g. deploy)
- Cost Recommendations
- Alerting & Budgets
- Anomaly Detection
- Reporting
- Documentation
- Training
- Support
- Automated Idle Resource Management
- Cloud Cost Business Intelligence

<yes><yes>

Engineering & Finance

All

Hourly

<yes><yes> **Optional**

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Utilized, Idle, Unallocated

<yes><yes>

Finance

All

Daily

<yes><yes>

<yes><yes>

<yes><yes>

<with><with> **Requires Tagging**

<with><with> **Requires Tagging**

<no><no> **Requires Tagging**

<with><with> **Requires Tagging**

<with><with> **Requires Tagging**

<no><no> **No Idle/Unallocated Cost Visibility**

<no><no>

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<no><no>

SaaS and Self-Hosted

SaaS

AWS, Azure and GCP

AWS, Azure, GCP and Oracle Cloud

<with><with>

Kubernetes Only

<yes><yes>

Percentage Cloud Spend

Percentage Cloud Spend

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Coming Soon

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<with><with>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<with><with> Manual Schedule

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Summary:Â

CloudHealth by VMware provides a great set of visibility, savings, and forecast tools for cloud providers, provided that you can dedicate the staff required to manage the complex reporting required. While they may have been one of the early innovators in cloud cost management, they've stagnated in new feature development since their acquisition by VMware (and are expected to stagnate further if the pending Broadcom acquisition is approved). The tool is difficult to implement, and engineering teams simply won't use it.

If you are looking for a tool that your engineers will actually use, that gives deep insights into all your multi-cloud and Kubernetes infrastructure, and provides innovative, automated features that create new cloud cost savings opportunities, then Harness Cloud Cost Management is the clear winner. Harness gives your FinOps teams the tools they need to accelerate cost savings, and maintain the most optimized cloud spend over time.

Cost Reporting:

Understanding cloud costs is the first fundamental for gaining control of your cloud spend. Without accurate and timely reporting, as well as continuous cost monitoring, your cloud costs can quickly spiral out of control.

For those with a PhD in analytics and reporting, CloudHealth by VMware is an amazing tool that provides a wealth of information on cloud costs. For the rest of their users, it is a very difficult tool to learn and understand, requiring extensive training to be able to create usable reports based on custom queries. This has led their customers to designate one or two people in finance to create and maintain weekly/monthly cost reports. Otherwise, no one else in the organization is using the tool, including the engineers tasked with chasing down cloud cost anomalies and implementing recommendations. ÂWe just weren't using itâ is the most common reason customers are moving away from CloudHealth by VMware.

Harness CCM gives you easy to use, easy to understand reports and dashboards out of the box, with an intuitive interface that you can easily customize to suit the needs of your organization. FinOps, DevOps and Engineering teams can get real time visibility into cloud costs, budget status and cost anomaly alerts to keep them on top of their cloud spend.Â

Cost Anomaly Monitoring:

Engineering is always innovating on new features that drive your business revenue, but what happens when they forget to turn off that expensive test cluster, and no one notices?Â End of month bill shock, an unpleasant experience for everyone when your cloud bill is tens of thousands of dollars over budget. Continuous cost anomaly reporting is critical to preventing out of control cloud spend.

Harness CCM provides continuous cost anomaly monitoring, using AI to detect unexpected changes in your cloud spend. Not just big cost anomalies, either. Harness CCM surfaces smaller cost changes that overtime add up to much larger bills. When anomalies are detected, users are alerted immediately, with detailed information on where the anomaly is occurring so that root cause analysis can be performed.

For such a fundamental feature for keeping cloud costs under control, CloudHealth by VMware has not made releasing this a priority. Cost anomaly detection is in beta now, has been in beta for months, and is expected to still be in beta for months. This is simply unacceptable for any fiscally responsible organization to be forced to manually review periodic cost reports to surface cost anomalies, days or weeks after they've begun.Â

Kubernetes Support:

The cloud-native computing world is built on Kubernetes, so having deep insights into your Kubernetes costs and how to optimize those costs is essential. CloudHealth by VMware does have basic cost reporting for Kubernetes workloads, but they do not provide analysis to help customers break out their utilized, idle, and unallocated costs within Kubernetes clusters. Nor do they provide recommendations for right-sizing Kubernetes resources in any way, leaving customers stranded when it comes to optimizing their Kubernetes infrastructure.

Harness CCM provides detailed cost analysis for Kubernetes clusters, at a very granular level that can be tailored for your business. Our Kubernetes cost recommendations can help you right-size your node pools as well as the underlying infrastructure, making your clusters as efficient as possible. We also deliver automated cluster cost management features such as Cluster Orchestrator for EKS that can save our customers up to 90% on cloud costs.Â

Automated Cloud Cost Optimization:

Getting engineers to take action on cloud cost recommendations shouldn't be your only path to cloud cost savings. Automation can unlock cost savings opportunities that simply can't be realized with manual or scripted efforts. Harness CCM has a multitude of automation tools that save our customers money, from idle cloud resource management (Cloud Autostopping) to spot instance orchestration, and more, that saves our customers up to 90% on cloud costs. CloudHealth has fallen behind the competition for automated cost savings. While they do have rudimentary policies for purchasing commitments, it falls short of many customers needs. Other than that, **CloudHealth by VMware has no automated cost savings features available for their customers.**

Cloud Cost Management

Interested in seeing what's under the hood? Browse through the Harness Cloud Cost Management Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/comparison-guide/datadog-vs-harness>

Comparison Guide

Harness

Service Reliability Management

VS

DataDog

Harness Service Reliability Management vs DataDog

Service Reliability Management

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

DataDog

Datadog provides cloud-scale monitoring and security for metrics, traces and logs in one unified platform.

3,200

2010

Public Company

Datadog is a publicly traded company with an IPO in 2019.

DataDog is categorized as:

Cloud Monitoring as a Service

What is the difference between Harness Service Reliability Management vs. DataDog?

Updated

November 30, 2023

- **SaaS & On-Premises**
- **Integrates with Change and Incident Management Sources**
- **Integrates with Leading Observability Solutions**
- **Custom Reliability Policies**
- **Reliability Guardrails Integrated with CI/CD Pipelines**
- **Service Health Indicators**
- **Service Reliability Checks**
- **Automated Deployment Verification**
- **Automated Rollback of Failed Deployments**
- **Correlation of Change Events to Service Health**
- **Correlation of Incidents to Service Health**
- **Error Tracking**
- **Reliability Audit Trails**
- **Fine-Grained Role-Based Access Control**
- **Unified Software Delivery Platform**
- **Composite SLOs**

<yes><yes>

<yes><yes>

<yes><yes>

Notification + Pipeline

<yes><yes>

<yes><yes> SLO or Metric

<yes><yes>

<yes><yes>

<yes><yes> WithÂ Harness CD

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<with><with> SaaS Only

<yes><yes>

<with><with> **A Few**

<with><with> **Notification Only**

<no><no>

<no><no>

<yes><yes>

<no><no>

<no><no>

<no><no>

<no><no>

<yes><yes>
<yes><yes>
<with><with> Not for SLOs
<no><no>
<yes><yes>
<yes><yes>
<with><with>
SaaS Only
<yes><yes>
<yes><yes>
<yes><yes>
<with><with> A Few
Notification + Pipeline
<with><with> Notification Only
<yes><yes>
<no><no>
<yes><yes> SLO or Metric
<no><no>
<yes><yes>
<yes><yes>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<with><with> Not for SLOs
<yes><yes>
<no><no>
<yes><yes>
<yes><yes>

Summary:

Harness and DataDog both provide SLO management capabilities. Harness Service Reliability Management (SRM) is a SLO management module that is part of a larger software delivery (CI/CD) platform. Harness SRM is designed to facilitate greater collaboration between SREs and Developers while also automating many actions related to SLO management like governing software deployments.

DataDog is a monitoring and observability vendor that provides some basic SLO management capabilities. DataDog's SLO management features can be useful to SRE teams but it does not provide automation across the software delivery lifecycle as evident in the following analysis.

SaaS & On-Premises:

Harness SRM is available as either SaaS or self-managed software that can be deployed anywhere.

DataDog is available as a SaaS solution only.

Integrates with leading observability solutions:

Harness SRM integrates with many of the leading monitoring, logging, and observability solutions to collect the data needed to build and track SLIs, SLOs, and Error Budgets.

Since DataDog is mainly a monitoring and observability solution, it does not integrate with many other competing products.

Custom Reliability Policies:

Harness SRM contains a built in policy engine based on Open Policy Agent (OPA). This enables Harness users to define and enforce custom policies with via UI or via policy-as-code. Policies within Harness can be used to send notifications or to actively manage software delivery pipelines via reliability guardrails.

DataDog offers the ability to create policies for notification purposes only.

Reliability Guardrails integrated with CI/CD pipelines:

Harness SRM reliability guardrail policies are used to automatically determine if deployment pipelines should proceed or stop given the status of SLOs and Error Budgets. This makes it possible to effectively standardize and scale the management of pipelines via SLO management processes.

DataDog does not offer this capability.

Service Health Indicators:

Harness SRM can determine and display the health of application services using SLOs and/or metrics to derive the overall health. By using a combination of SLOs and metrics, Harness can provide a more precise calculation of service health.

DataDog does not provide service health indications based upon SLOs.

Automated Deployment Verification and Rollback (Continuous Verification):

Continuous Verification is the process of monitoring your app for abnormalities after a deployment. For example, Continuous Verification could catch a latency issue or 5xx errors and automatically roll back your app to the previous version. The idea is to catch errors as quickly as possible – ideally, before customers notice – and make a seamless transition back to the prior version.

Harness provides Continuous Verification out of the box, effectively reducing risk and reputational damage from downtime. Harness supports many vendors, including Prometheus, Datadog, AppDynamics, New Relic, StackDriver, CloudWatch, and custom monitoring and observability tools.

DataDog does not offer this capability. They are a monitoring solution that can be used to analyze the quality of software services over time.

Correlation of Change Events to Service Health:

Harness SRM ingests change events and shows them on a timeline that is aligned with SLO and Error Budget charts. The user can select and drill down into specific timeframes to perform root cause analysis on the impacted SLOs.

DataDog does not offer this capability.

Correlation of Incidents to Service Health:

Harness SRM ingests incident events and shows them on a timeline that is aligned with SLO and Error Budget charts. The user can select and drill down into specific timeframes to perform root cause analysis on the impacted SLOs.

DataDog does not offer this capability.

Fine-Grained Role-Based Access Control:

Harness enables full customization of role based access controls (RBAC) with the SRM module and across the larger platform. Harness RBAC was built to offer the ultimate in flexibility as required by the largest enterprises.

DataDog does have an RBAC system but it has limited roles related to SLO management.

Unified Software Delivery Platform:

Harness SRM is the SLO management module of the Harness Software Delivery Platform. Each module can be used standalone (integrated as a best of breed solution to a DevOps toolchain) or as part of the platform. When used as part of the platform, each module passes meta-data and can provide greater levels of automation than if used standalone.

DataDog does not offer a unified software delivery platform.

**Please note: Our competitors, just like us, release updates to their products on a regular cadence. We keep these pages updated to the best of our ability, but there are bound to be discrepancies. For the most up-to-date information on competitor features, browsing the competitor's new release pages and communities are your best bet.*

Request a Demo

Service Reliability Management

Interested in seeing what's under the hood? Browse through the Harness Service Reliability Management Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/priceline-move-gcp-canary-deployments>

Priceline Accelerates Move to GCP and Canary Deployments Using Harness

Harness helped Priceline quickly and easily build deployment pipelines from scratch for its migration to GCP and GKE. Find out how.

About

At Priceline, part of Booking Holdings, Inc. [NASDAQ: BKNG], they believe every trip is a big deal to their customers. Whether it's a much-needed getaway, wedding, reunion, graduation, or rooting on a favorite team—those are the moments that nobody should miss out on. Their mission? Be the best travel dealmakers in the world. They provide their customers with smart and easy ways to access the very best deals available on hotel rooms, flights, rental cars, vacation packages and cruises, so they can be there for the moments that matter the most to them.

Cloud Migration and Deployment Pattern Changes

Priceline originally built a custom deployment solution, which they successfully used to deploy services to their on-prem environments. This worked because the on-prem environments were relatively homogeneous and Priceline kept to a strict set of deployment patterns.

When the team at Priceline decided to move applications from on-prem to GKE and simultaneously wanted to develop new deployment patterns, they knew they had to make a change to their tooling. Priceline's DevOps team quickly realized these initiatives were out of scope for the homegrown tool and began the search for a new solution.

Using their existing solution, the team would need at least 3 engineers for 6 months just to implement canary deployments in GKE.

Technically, the team could have repurposed their homegrown pipelines for the cloud, but Priceline wasn't willing to waste the time and effort developing and maintaining their own software when they could find a tool that specializes in what they needed.

Modernization With Little Effort

Priceline chose Harness to modernize its technology stack. With Harness, Priceline was able to adopt advanced deployment patterns with little effort. The canary deployments that would have taken Priceline 6 months and 3 engineers to set up took 1 developer a couple of days using Harness.

Harness also made the transition to GKE easy by providing things like secrets management, and by allowing developers to easily create

new pipelines for any new Helm chart.

So far, the move to GCP and Harness has worked out great. Harness helped Priceline quickly and easily build deployment pipelines from scratch for its migration to GCP and GKE. Currently, 45% of Priceline's applications are hosted on GKE, with the goal being to migrate 100% of the applications.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/reduce-developer-toil>

Lavasoft Reduces Developer Toil by 94% Using Drone and Harness

Lavasoft uses both Drone and Harness to create the ideal software delivery environment.

About Lavasoft

Lavasoft, a subsidiary of Avanquest Software, is a world-leading developer of personal and professional software, encompassing a large number of categories including Creativity and Learning software, Utilities, and Multimedia.

Identifying Delivery Issues

Mathieu Fecteau is a software delivery manager at Lavasoft and helped uncover the company's software delivery issues.

Lavasoft's applications were written in a variety of languages and 80% of them were hosted on-premise. Lavasoft managed this complexity using equally complex Jenkins Pipelines stitched together with custom scripts. Any change to the Jenkins CI pipelines could take hours, and Lavasoft didn't have the engineering resources to effectively manage the system. The complexity and lack of spare resources made it difficult for Lavasoft to implement DevOps principles.

Lavasoft's product team had little visibility into deployment schedules and metrics. Product leaders weren't aware of deployment failures leading to miscommunication about what was actually in production. The deployment failures required intensive manual intervention and resulted in a developer spending 8 hours every week on verification. Rollbacks of a failed deployment required the manual re-deployment of local files.

Mathieu needed to reduce Lavasoft's delivery complexity and increase confidence in the deployment process.

Finding a Container Native CI Tool

After evaluating other CI vendors, Mathieu decided to evaluate Drone.

Mathieu onboarded his first application in 15 minutes. As he onboarded more applications, Mathieu discovered the secret behind Drone's success. Simplicity. Drone's containerized plugins reduced inter-dependencies and didn't require custom groovy scripting. Upgrading Drone took 2 minutes, a 97% decrease from the 1 hour it took LavaSoft to upgrade their Jenkins pipelines. Drone is so intuitive that a new hire can use Drone on their first day at LavaSoft.

Stop Babysitting Deployments

After solving continuous integration, Mathieu turned his attention to continuous delivery. Harness piqued his interest because of its Kubernetes focus and compatibility with on-premise applications.

In his first 15 minutes of using Harness, Mathieu connected his pipelines to Drone. In his first week using Harness, Mathieu onboarded 40 applications and reduced Kubernetes migration time to 1 hour per application.

Mathieu also gave LavaSoft's product managers visibility into the deployment process, removing friction between PMs and Devs. Continuous verification plugged into Datadog and ELK to reduce verification time spent by 94% from 8 hours a week to 30 minutes.

Mathieu found his perfect CI/CD solution using a combination of Drone CI and Harness CD.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/jobvite-reduces-deployment-time-surges-deployment-volumes-saves>

Jobvite Reduces Deployment Time by 90%, Surges Deployment Volumes, and Saves \$500k With Harness

Harness is core to Jobvite's deployment scale and growth, without impacting developers' day-to-day work with tools and process changes. Find out how.

- 90% reduction in deployment time
- 1,500% increase in deployment volumes
- \$500,000 savings over building in-house CD

Jobvite, an Employ Inc. brand, delivers a purpose-built, scalable, and proven end-to-end solution that streamlines complex talent

acquisition activities. Jobvite acts as a strategic partner to empower customers to meet their ever-changing business needs, driving predictable outcomes, creating engaging candidate experiences, and increasing efficiency by optimizing TA capacity. With Jobvite's Evolve Talent Acquisition Suite, organizations can streamline talent acquisition across the entire talent lifecycle and deliver quality candidates to drive their business forward.

Challenge: Growing Complexity of Maintaining Custom Scripts

Jobvite has a mature continuous integration (CI) platform in Jenkins, which allows developers to automate building and testing to surface potential errors before code reaches higher level environments. Over the past few years, this Jenkins platform has been extended, primarily with custom scripts to cover various parts of their continuous delivery (CD) process. A key challenge the release team faced was the growing complexity of maintaining custom scripts as their core application, services, and technologies evolved.

âOur deployment process consisted of a lot of manual work, a lot of scripting, and it wasn't repeatable,â recalled Joshua Jackson, Senior Director of Engineering at Jobvite. âWe were running 40 applications across clouds, all with manually triggered deployments.â

In addition, while some portions of testing for performance and quality were automated, a large portion of manual effort went into each build, verified by the team of Quality Analysts. This manual process was complex to maintain and debug when deployments failed. With 40 applications and a six-week release cycle â plus development and QA environments â Jobvite was running about 500 deployments per month.â

In addition to their key initiative for internal tooling teams, Jobvite needed a true CD process that was sustainable, repeatable, and consistent. According to Jackson, âOur long-term goal was to have developers deploying daily and hourly. We needed to ensure performance and quality for all deployments, without impacting customers.â

Using homegrown scripts helped streamline deployments, but Jobvite was still manually triggering and monitoring deployments across systems. âWhen the team moved to triggering deployments via Jenkins, it gave them little visibility into failures, which added more manual work to understand why something might fail. That made it painful for the engineering organization to resolve,â Jackson said.â

So, internal tooling teams set out to enable engineers to deploy their own code and artifacts through automation, visibility, and velocity.

Solution: A Central Tool to Realize Their Full CI/CD Vision

In 2016, while Harness was still in beta, Jobvite jumped at the chance to automate and accelerate its CD while maintaining Jenkins for CI. The company had Harness deployed within a few hours, and at the end of the first day, teams were already building and deploying CD pipelines, such as one for its production microservices built using Java, Docker, Kubernetes, and AWS. Harness also integrated with Jobvite's ecosystem of DevOps tools, including Jenkins, AppDynamics, New Relic, AWS Codedeploy, and Elastic.

âHarness was a natural fit as we transitioned from EC2 to Kubernetes,â said Jackson. âWe were increasing the number of applications through splitting some macro services into microservices, and through a number of strategic business mergers and integrations that brought new applications into the mix. We went from 40 applications deployed every six weeks to about 240 deployed multiple times per day across development and QA environments, and every two weeks for our production environment.â

Harness is core to Jobvite's deployment scale and growth without impacting developersâ day-to-day work with tools and process changes. At the time, Jobvite estimated it would have cost \$500,000 (not including ongoing support and maintenance) to build a system similar to Harness. Now, Jobvite runs Harness on every developer's machine to manage local development while integrating with Argo CD, an open source GitOps CD tool for Kubernetes. This integration makes it easier to manage GitOps-based deployments at scale and gives development teams more autonomy over how, and when, they deploy.

âWith its integration to Argo CD, Harness is the central tool of observability, visibility, and centralization to realize our full CI/CD vision,â Jackson shared. âWe can plug in any tool we want. We can natively use Argo for deployments. And, for workflows on other dev, QA, staging, and production environments, we can layer the visibility and UI of Harness on top to give our developers a central view of where their code has gone out. Harness sees it all, automatically, and makes it very transparent, so we can see everything from the cost of running applications to security to development to operations.â

Result: Deployment Time Reduced by 90%

From about 500 deployments *per month* to more than 2,000 *per week*, Jobvite increased its deployment volume and velocity by 1,500%. Prior to Harness, a typical production deployment for the team would take 27 minutes. Now that same deployment takes just two minutes â a reduction of more than 90%. Jobvite didn't even need to rewrite existing CD processes, it simply reused and automated how existing scripts, repositories, and Jenkins CI platform performed together.

âWe've doubled or tripled our capacity for the number of instances we're running, and we are also doing dynamic scaling of services, which we weren't doing before,â Jackson added. âHarness keeps track of it all, and allows us to do the modifications we need to do.â

When it comes to value, Jackson points not only to the drastically increased deployment velocity, but also to the day-to-day benefits that improve the quality of life for developers.

âIt's the simple things that stand out,â said Jackson. âHarness lets us set the deployment trigger, and that's immensely powerful. Seeing information in a central tool lets us quickly find the cause of issues, and Harness filters the information so we can avoid information overload. Harness templates help drive behaviors and are one of the strongest capabilities of the platform. Our failure rate has dropped significantly and our lead times have also dropped.â

Harness also lets Jobvite diagnose and debug live applications rather than trying to reproduce issues locally. And, Harness ensures deployment best practices are followed, which Jackson calls the “easy golden path for developers.” But, most of all, it’s the visibility that Harness provides that helps Jobvite continue on its path to daily releases.

“Harness has given us a single pane of glass into our deployment process,” concluded Jackson.

About Employ Inc.®

Jobvite is an Employ Inc. brand. Employ Inc. empowers organizations of all sizes to overcome their greatest recruiting and talent acquisition challenges. Offering a combination of purpose-built, intelligent technologies, services, and industry expertise, Employ provides SMB to global enterprises with a single solution for recruiting and growing a diverse workforce. Through its JazzHR, Lever, Jobvite, and NXTThing RPO brands, Employ serves more than 18,000 customers across industries. For more information, visit www.employinc.com.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform®

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/raisin-runs-from-jenkins-nightmare>

Raisin Runs from Jenkins Nightmare â Saves 525k per Year

Raisin removed their CD maintenance migraine, find out how you can too!

About

Raisin is the leading pan-European wealth management platform, connecting retail customers with financial institutions looking to expand or diversify their deposit reach. They are available in English across Europe and operate country-dedicated platforms in Germany, France, Spain, the United Kingdom, Ireland, the Netherlands, and Austria. In 2020, they launched a savings-as-a-service branch in the U.S.

Modernize, They Said! Itâs Easier and Cheaper, They Said!

Raisin migrated its applications to the cloud to take advantage of modern technologies like Kubernetes.

Makes sense. Everyone is doing it.

A side effect of their application modernization was the need for a new software deployment strategy to reduce microservice deployment effort.

Issue identified.

Raisin created a central DevOps team led by José Meyer to tackle the deployments.

Plan created.

The team used Jenkins and custom scripts to build deployment pipelines for various developer teams.

Problem solved! hold up. Incoming migraine.

The Jenkins pipelines allowed Raisin to deploy 10 times every month, but the 6 new members of the DevOps team spent 80% of their time keeping Jenkins standing. That equates to roughly \$700k per year in salaries (assuming \$70 per hour per team member). Between Jenkins maintenance, updates, plugin additions, and troubleshooting, the team hardly had time for anything else. Jenkins was too fragile to be left on its own.

Because of the Jenkins pipeline complexity, onboarding a new application took roughly 2 days. That doesn't sound terrible on its own, but figure in the massive maintenance effort after onboarding and you're talking about a major time suck.

José and the DevOp team initially picked Jenkins because it was a "free" solution. But after dealing with the Jenkins nightmare, José began looking for better ways to deploy software.

A Modern CD Solution for a Modern Architecture

During José's search, he came across Harness. With Harness, José and the DevOps team distributed software deployment responsibility to the application teams. The application teams now manage their own services, pipelines, and triggers. José's team of 6 now only spends 20% of their time administering software delivery. That's a 60% reduction in effort worth roughly \$525k.

The democratization and automation of software delivery have allowed Raisin to deploy 53 times a month â a 5x increase from their previous deployment velocity.

The news of Harness's deployment success spread quickly around Raisin. Application teams began lining up to have their app deployed using Harness. Luckily, it only takes José's team 30 minutes to onboard a new application due to Harness's templatization and reusable connectors.

Modernizing applications and CI/CD can be a challenge. Maybe it's a challenge you're getting ready to experience. Check out how Harness makes the transition less painful.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform®

Need more info? Contact Sales

Burst SMS Removes Outage Risk for Their Customers

Harness delivers better DevOps practices to all our customers. Find out how Burst SMS used Harness to reduce outages.

About

Burst SMS is a global online messaging platform, delivering messages since 2008. They specialise in helping businesses communicate with their customers via SMS with simple yet highly intuitive messaging tools. Innovation is what they strive for by building new features with their clients in mind. Their products are developed in house but refined by their customers.

Deployments Shouldn't Be a âBig Bangâ

Burst SMS struggled to deploy software. Deployments were extremely manual, infrequent, and risky. These factors compounded on top of each other to create what Burst SMS refers to as âBig Bangâ deployments. The manual nature of the deployments meant deployments only occurred once every 3-4 months. Because deployments only occurred every 3-4 months, releases were large with a ton of new features in each batch. Because releases were large, something would often break in production. Because deployments were manual, rolling back was extremely difficult. This resulted in roughly 6 days of downtime a year.

Burst SMS began evaluating new delivery solutions. The company quickly ruled out Jenkins and Jenkins X because of the large time and resource commitment needed to stand the tools up. Their search was further complicated by an organizational shift to a hybrid cloud hosting solution. Burst SMS would need to find a tool to help with both AWS and GCP deployments.

Kicking Off a New DevOps Journey With Harness

Harness brought accessible DevOps practices to Burst SMS. Developers now have access to advanced deployment strategies without having an advanced knowledge of Kubernetes or detailed knowledge of a specific cloud provider.

Now that Burst SMS deploys using Harness, deployment velocity has increased 8x to once every two weeks. This has reduced the size of each deployment, which has reduced deployment time to 5 minutes and inherently reduced the risk of failure for each deployment. If something does fail in production, Harness allows Burst SMS to roll back in seconds. This decreased annual downtime to minutes.

Harness also allows Burst SMS to run a hybrid cloud model by providing a layer of abstraction on top of each cloud. Users just have to learn how to use Harness â they don't have to learn the ins and outs of AWS and GCP.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform^{Â®}

Need more info? Contact Sales

Harness

Feature Flags

VS

Split

Harness Feature Flags vs Split

Feature Flags

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

Split

Split is the leading feature delivery platform for engineering teams to confidently release features fast.

200-500

2015

\$110M

Split is categorized as:

Feature Flags

We're the best Feature Flag management tool. See for yourself.

Harness Feature Flag Management vs. Split

Split vs Harness: Feature Flag Tool Comparison | Harness

Updated

November 30, 2023

- Primary Users
- Primary Use Case
- SaaS & On-Premises Support
- Simple Developer Onboarding
- Feature Flag Pipelines
- Native CI/CD Support & Pipeline Extensibility
- Config-as-Code & GitOps
- SDK Support
- Automated Feature Verifications
- Flag Lifecycle Management
- Binary/Multivariate Flags
- Flexible Targets (Users, Groups, Regions, Infra)
- Streaming & Polling Support
- Security/Access Control

- **Global Governance (OPA Support)**
- **Audit Logs**
- **Reporting & Dashboards**
- **Proxy Service**

Developers, Product Managers, Ops, Eng Managers

Manage risk & velocity; Automation; Governance

<yes><yes> **SaaS & On-Prem Supported**

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

All Major Languages

<yes><yes>

<with><with> **Coming Soon**

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes> **RBAC, SSO, More in Progress**

<yes><yes>

<yes><yes>

<yes><yes> **Reporting Across SDLC**

<yes><yes>

Product Marketers, Product Managers

Experimentation

<no><no> **SaaS Only**

<no><no>

<no><no>

<no><no>

<no><no>

All Major Languages (Except Go)

<with><with>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<with><with>

<no><no>

<yes><yes>

<with><with> **Limited to FF Data**

<no><no>

Developers, Product Managers, Ops, Eng Managers

Product Marketers, Product Managers

Developers, Product Managers, Ops, Eng Managers

Product Marketers, Product Managers

Manage risk & velocity; automation; governance

Experimentation

Manage risk & velocity; automation; governance

Experimentation

SaaS & On-Prem Supported

<no><no>

SaaS Only

SaaS & On-Prem Supported

<no><no>

SaaS Only

<yes><yes>

<yes><yes>

<no><no>

<no><no>

<yes><yes>

<no><no>

All major languages

<yes><yes>

<yes><yes>

<with><with>

<with><with>

<yes><yes>

<with><with>

<with><with> Coming soon

<no><no>

<no><no>

<with><with> Coming soon

<no><no>

<yes><yes>

RBAC, SSO, more in progress

<yes><yes>

<yes><yes>

<yes><yes>

RBAC, SSO, more in progress

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Reporting across SDLC

<with><with> Limited to FF data

<yes><yes>

Reporting across SDLC

<with><with> Limited to FF data

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

SaaS & On-Premises Support:

Split only offers a SaaS version of their product. Harness supports both SaaS and on-prem deployment models.

Primary Users:

Delivering software is a process that starts at building an application. In the same way, feature flags should provide value to teams at each phase of the software delivery lifecycle. It's not just about shipping code faster and creating improvements for users; feature flags should also make daily operations easier, and provide visibility and management ability. Harness Feature Flags was built to support all of these use cases across Developers, Product Managers, Operations, and Engineering Managers.Â

The focus is unleashing the power of feature flags for software delivery teams holistically. And while Harness focuses on a developer-first experience, the same workflows and processes are built to support software delivery teams where they struggle the most. Hint: it's not just shipping features.

Simple Developer Onboarding:

When building Feature Flags, Harness focused heavily on the developer experience - developers are, after all, the true arbiters of feature flags. Our intent was to make sure that the setup process is fast and intuitive, and that teams can get started and see value from feature flags right away. As of today, we don't see this sort of first-class support for developers coming from Split. In fact, with their primary focus being on marketing and product teams for experimentation, we're not confident that they ever will create this support.Â

Feature Flag Pipelines:

Harness features a visual drag-and-drop pipeline builder that supports hundreds of integrations, and this is a standard part of the Harness platform. Users unlock automated progressive delivery, and Harness uniquely allows for automated feature verification so that users

don't have to monitor stages of their rollout and can truly automate rollout and kill switches.

Native CI/CD Support & Pipeline Extensibility:

Split, as a third-party tool, is completely abstracted from your CI/CD process, which perpetuates the view that feature flags are an ancillary step. Ideally, this view would be abolished. At Harness, we view feature flags as a crucial part of the CI/CD pipeline - and as such, it becomes a natural step in the everyday workflow of development teams and is integrated into CI/CD as a unified pipeline. This allows for users to get visibility into any change from code to release, and create standardized and automated processes across all stages of the SDLC.

As part of the Harness platform, users also can rest easy knowing that they can get a full audit log, manage teams with security like RBAC, share environments and APIs, and see cross-platform analytics. Of course, all of this is available for just Harness Feature Flags, and its power is more pronounced on the platform.

Config-as-Code & GitOps:

Split's user focus is on marketing and product teams for experimentation. However, it's typically the developer teams who are the arbiters of feature flags, with experimentation being a resulting capability. At Harness, we chose to focus on the development point of the value chain.Â

It's important to meet developers where they are and work within their processes. With Harness, teams have the ability to create pipelines visually and combine that with a pipelines-as-code or config-as-code approach; visual pipelines create YAML files that can be changed to reflect back into the visual pipeline, and these can automatically be synced with Git so changes are reflected immediately and without added effort, all from within our platform.

SDK Support:

Harness and Split support all major languages, both server-side and client-side (except for Split not supporting Golang). However, it would be unjust not to mention that Split, as of today, supports a few more SDKs. Harness Feature Flags has been on the market since July 2021 and supports a total of 10 SDKs. Split supports 14, and it's only a matter of time until Harness far outpaces the rate of SDK support from Split because of our developer-first focus.

Automated Feature Verifications:

Harness initially pioneered the concept of Continuous Verification in the software delivery space. Having battle-tested the capability over years with hundreds of customers, the ability to have Harness automatically monitor deployments and take appropriate rollout or rollback options is extended into Harness Feature Flags. Users can use the ML-based automated verification feature to have Harness automatically verify the health of live features and take appropriate remediation action should things go wrong - it obviates the need for users to manually monitor and make changes during a feature rollout.

While Split does provide feature verification, as of today it seems limited in its support and level of automation. Notably, it seems to have a pretty shallow integration library, and it focuses more on alerting of issues rather than allowing users to preconfigure desired actions based on verification outcomes.

Security/Access Control:

The key difference between Harness and Split is the extensibility of security and access control to more than just feature flags. With the native Harness support for CI/CD, users can extend the same security/access control profiles end-to-end across SDLC, rather than having to learn multiple tools and create those controls from scratch. By simplifying this process end-to-end, users are able to focus on using the tools rather than configuring them.

At Harness, we take security extremely seriously, as evidenced by our DevSecOps approach. We took the same views with Feature Flags, ensuring RBAC and SSO were available from inception, with more controls coming soon.Â

Governance & Compliance:

Harness takes governance and compliance seriously, providing CI/CD users with fine-grained RBAC, audit trails, many ways to manage secrets, and more. We've taken that same approach with Feature Flags and are investing heavily in the area, letting teams build rules and processes, and helping automate cleanup and flag management. Harness also supports Open Policy Agent (OPA), enabling Enterprise-scale governance as-code across Feature Flags and the rest of the Harness platform. Harness is the only feature management solution to offer this.

Reporting & Dashboards:

Since Split is a standalone feature flags tool, reporting and dashboards are limited to data within the tool itself - understandably so. Harness, however, is in the process of linking Feature Flags data to the rest of the software development lifecycle, and as such, will provide reporting and dashboards for all products in the same easily-accessible place. Harness is actually the only platform on the market that brings together analytics across the whole SDLC into one place - one tool to rule them all, one tool to bind them.

But what if you're using only feature flags and don't care about CI/CD? Harness provides the most complete visibility into feature flags of any tool on the market. Any bit of data that's related to your feature flags use and operation, you can visualize in Harness to create

custom dashboards to show exactly what you need at any time.

**Please note: Our competitors, just like us, release updates to their products on a regular cadence. We keep these pages updated to the best of our ability, but there are bound to be discrepancies. For the most up-to-date information on competitor features, browsing the competitor's new release pages and communities are your best bet.*

Feature Flags

Interested in seeing what's under the hood? Browse through the Harness Feature Flags Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/onboard-kubernetes-applications>

ConsumerTrack Onboards Kubernetes Apps in 3 Hours

ConsumerTrack saves \$580k with Harness.

About ConsumerTrack

ConsumerTrackâ¢ helps brands connect with smart, aspirational audiences. Through our partnerships and owned-and-operated websites, we engage in-market consumers and connect them with the products and services that they are seeking.

Migrating to Kubernetes on AWS

ConsumerTrack were in the process of migrating to a microservices architecture running on AWS and Kubernetes. Approximately 24 services (4 existing & 20 new) would require new deployment pipelines and onboarding within a 12-month period. The Continuous Integration (CI) process (code to artifact) at ConsumerTrack was mature with tools like Git, Jenkins, CircleCI, Selenium, and BrowserStack. However, the Continuous Delivery (CD) process and deployment pipelines at ConsumerTrack still remained a manual ad-hoc scripted process owned by the DevOps team. âOnboarding a new Kubernetes service took on average 2 weeks for 2 DevOps engineers to manually script using Jenkins pipelines.â said Jeremy Malara, DevOps Manager at ConsumerTrack. The total effort/resource for onboarding these 24 new Kubernetes apps was ~96 weeks of effort and ~\$290,000 in cost.

Manual, Scripted Deployments, Babysat by DevOps

Development teams at ConsumerTrack currently deploy 6-7 times per day.

âIn the past, deployments were manual, ticketed and often babysat by 2 DevOps engineers and 1 developer from each dev team,â said Jeremy. Promoting code from each Dev to QA environment could take 2 hours (best case), with code promotion from QA to production taking 1 hour (best case). In addition, the verification/health checks of production deployments were manual, meaning DevOps and developers had to manually check metrics in Datadog/Prometheus and check application events in Splunk for a deployment to be deemed successful.

Self-Service Continuous Delivery for Developers

To automate and streamline their manual Continuous Delivery (CD) process, ConsumerTrack evaluated Harness. âIn our Harness Evaluation alone, we reduced deployment time from Dev to QA from 2 hours to 5 minutes,â said Jeremy. The ultimate goal was to empower developers with self-service Continuous Delivery. Instead of DevOps engineers owning each deployment (and troubleshooting), they now govern and template the pipelines, which developers can instantiate using Harness pipeline templates and workflows from the DevOps team. However, the biggest win for ConsumerTrack and Harness was the onboarding of new Kubernetes apps. âInstead of 2 DevOps Engineers spending 96 weeks manually scripting Kubernetes Jenkins pipelines, we can do the same in Harness in one week,â said Jeremy. The effort and resource to create a new Kubernetes pipeline dropped from 2 weeks and 2 DevOps engineers to just 3 hours and 1 DevOps engineer. With Harness, Kubernetes pipelines are now automated end-to-end so code is promoted from dev to production in minutes vs. hours. Harness also uses unsupervised machine learning to automate deployment verifications using data from Datadog, Splunk and Prometheus.

First Year Savings of \$580,000

Today, Developers at ConsumerTrack now deploy their own code.

ConsumerTrack also saw the following business benefits from Harness Continuous Delivery:

- **Reduced deployment time (commit to production) by 92%** from 3 hours to 15 mins.
- **Reduced deployment effort by 16 hours per day** equivalent of 2 DevOps FTEs (\$290,000)

- **Reduced onboarding time by 96%** from 80 hours (2 weeks) to 3 hours per service.
- **Reduced onboarding effort of 24 Kubernetes services from 96 weeks to 72 hours**, the equivalent of 2 DevOps FTEs (\$290,000).

The above benefits at ConsumerTrack translate to total year-one saving of approximately \$580,000.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/comparison-guide/blameless-vs-harness>

Comparison Guide

Harness

Service Reliability Management

vs

Blameless

Harness Service Reliability Management vs Blameless

Service Reliability Management

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management

Blameless

Blameless is an SRE software platform that helps organizations and engineers streamline their reliability efforts.

79

2017

\$50.1M

Blameless is categorized as:
End-to-end SRE platform

What is the difference between Service Reliability Management vs. Blameless?

Blameless vs Harness: Service Reliability Management Tool Comparison

Updated

November 30, 2023

- **SaaS & On-Premises**
- **Integrates with Change and Incident Management Sources**
- **Integrates with Leading Observability Solutions**
- **Custom Reliability Policies**
- **Reliability Guardrails Integrated with CI/CD Pipelines**
- **Service Health Indicators**
- **Service Reliability Checks**
- **Automated Deployment Verification**
- **Automated Rollback of Failed Deployments**
- **Correlation of Change Events to Service Health**
- **Correlation of Incidents to Service Health**
- **Error Tracking**
- **Reliability Audit Trails**
- **Fine-Grained Role-Based Access Control**
- **Unified Software Delivery Platform**
- **Composite SLOs**

<yes><yes>

<yes><yes>

<yes><yes>

Notification + Pipeline

<yes><yes>

<yes><yes> SLO or Metric

<yes><yes>

<yes><yes>

<yes><yes> WithÂ Harness CD

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<with><with> **SaaS Only**

<yes><yes>

<yes><yes>

<with><with> **Notification Only**

<no><no>

<with><with> **SLO based**

<no><no>

<no><no>

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

<with><with> **3 Roles**

<no><no>

<yes><yes>

<yes><yes>

<with><with>
SaaS Only

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Notification + Pipeline

<with><with> Notification Only

<yes><yes>

<no><no>

<yes><yes> SLO or Metric

<with><with> SLO based

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes> With Harness CD

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>
<no><no>
<yes><yes>
<yes><yes>
<yes><yes>
<with><with> 3 Roles
<yes><yes>
<no><no>
<yes><yes>
<yes><yes>

Summary:

Harness and Blameless both provide SLO management capabilities. Harness Service Reliability Management (SRM) is a SLO management module that is part of a larger software delivery (CI/CD) platform. Harness SRM is designed to facilitate greater collaboration between SREs and Developers while also automating many actions related to SLO management like governing software deployments.

Blameless provides SLO management capabilities as part of a larger SRE-centric platform. Blameless does not provide SLO-based automation across the software delivery lifecycle as evident in the following analysis.

SaaS & On-Premises:

Harness SRM is available as either SaaS or self-managed software that can be deployed anywhere.

Blameless is available via SaaS only.

Custom Reliability Policies:

Harness SRM contains a built in policy engine based on Open Policy Agent (OPA). This enables Harness users to define and enforce custom policies via UI or via policy-as-code. Policies within Harness can be used to send notifications or to actively manage software delivery pipelines via reliability guardrails.

Blameless offers the ability to create policies for notification purposes only.

Reliability Guardrails integrated with CI/CD pipelines:

Harness SRM reliability guardrail policies are used to automatically determine if deployment pipelines should proceed or stop given the status of SLOs and Error Budgets. This makes it possible to effectively standardize and scale the management of pipelines via SLO management processes.

Blameless does not offer this capability.

Service Health Indicators:

Harness SRM can determine and display the health of application services using SLOs and/or metrics to derive the overall health. By using a combination of SLOs and metrics, Harness can provide a more precise calculation of service health.

Blameless can determine and display the health of application services using SLOs only.

Service Reliability Checks:

Harness SRM can determine the health of services across all stages of software delivery by using a combination of AI/ML techniques on logs and metrics and through error tracking. These capabilities make it possible for developers to proactively improve the quality and reliability of their application services before deploying to production.

Blameless does not offer service reliability checks.

Automated Deployment Verification and Rollback (Continuous Verification):

Continuous Verification is the process of monitoring your app for abnormalities after a deployment. For example, Continuous Verification could catch a latency issue or 5xx errors and automatically roll back your app to the previous version. The idea is to catch errors as quickly as possible – ideally, before customers notice – and make a seamless transition back to the prior version.

Harness provides Continuous Verification out of the box, effectively reducing risk and reputational damage from downtime. Harness supports many vendors, including Prometheus, Datadog, AppDynamics, New Relic, StackDriver, CloudWatch, and custom monitoring and observability tools.

Blameless does not offer this capability.

Error Tracking:

Harness SRM is capable of detecting all run-time exceptions within Java applications and providing detailed debugging information so that developers can resolve problematic exceptions before they deploy to production. This improves the quality of software which also improves reliability over time.

Blameless does not offer error tracking capabilities.

Fine-Grained Role-Based Access Control:

Harness enables full customization of role based access controls (RBAC) with the SRM module and across the larger platform. Harness RBAC was built to offer the ultimate in flexibility as required by the largest enterprises.

Blameless offers 3 roles (admin, reader, writer) with a total of 3 rules (read, create, execute) available for SLO management.

Unified Software Delivery Platform:

Harness SRM is the SLO management module of the Harness Software Delivery Platform. Each module can be used standalone (integrated as a best of breed solution to a DevOps toolchain) or as part of the platform. When used as part of the platform, each module passes meta-data and can provide greater levels of automation than if used standalone.

Blameless does not offer a unified software delivery platform.

**Please note: Our competitors, just like us, release updates to their products on a regular cadence. We keep these pages updated to the best of our ability, but there are bound to be discrepancies. For the most up-to-date information on competitor features, browsing the competitor's new release pages and communities are your best bet.*

Request a Demo

Service Reliability Management

Interested in seeing what's under the hood? Browse through the Harness Service Reliability Management Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/comparison-guide/travisci-vs-drone>

Comparison Guide

Harness

Continuous Integration

VS

Travis CI

Harness Continuous Integration vs Travis CI

Continuous Integration

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

Travis CI

Travis CI is a hosted continuous integration service used to build and test software projects hosted at GitHub and Bitbucket.

51-100

2011

â

Travis CI is categorized as:
Continuous Integration

â

What is the difference between Harness DevOps Tools Vs. Travis CI?

Travis CI vs Drone: DevOps Tools Comparison

Updated

November 30, 2023

- Open Source Version
- GitHub Stars
- Self-Service (Simple)
- No Scripting Required
- Container & Cloud-Native
- Traditional App Support
- GitOps (Pipelines as Code)
- Any Source Code Manager
- Containerized Pipelines
- Containerized Plugins
- Secrets Management
- Command Line Interface
- Scalability (Required Infra)
- Admin & Maintenance
- Total Cost of Ownership
- Pricing

Free & Paid

24800

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Vault/KMS/3rd

<yes><yes>

Lightweight

<yes><yes> **25 FTE**

<yes><yes>

<yes><yes> **Per User**

Free

8200

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<with><with> **(Limited Plugins)**

<with><with>

<yes><yes> **25 FTE**

<yes><yes>

<with><with>

âPer Job

Free & Paid

Free

24800

8200

<yes><yes>

<yes><yes>
<yes><yes>
Vault/KMS/3rd
<with><with> (Limited Plugins)
<yes><yes>
<with><with>
Lightweight
<yes><yes> 25 FTE
<yes><yes> 25 FTE
<yes><yes>
<yes><yes>
<with><with>
<yes><yes> Per User
âPer Job

Open source vs. Open core:

Travis CI loves the open-source community. Validated open-source projects are free to test, forever. Itâs their way of giving back. Travis CI is not, however, open-source itself. There is a free plan that can accommodate small projects and small businesses, but itâs still a SaaS solution that is, for all intents and purposes, paid. When Harness acquired Drone, it committed to keeping it open-source forever. Harness recently reaffirmed its investment in the open-source solution with a massive release where a sleeker interface, new visual pipeline builder, governance and security features, and real-time debugging tools were added. While this feature-rich version is free, there is also a paid version of Drone that provides access to enterprise support and more integrations and features yet. Additional features include secrets management options, autoscaling, custom plugins, and more.

Self-Service (Simple):

One of the great things about Travis CI is how easy it is to get started. Once you get set up and configured, you can be running unit tests in as little as 5 minutes. Travis CI supports 35 languages and supports cloud-native and traditional apps. Travis CI could, however, improve when it comes to documentation, integrations, and plugins. While it is a good basic tool, it does currently lack modern features more experienced users have come to expect, especially when it comes to security and governance, and overall flexibility/customization for complex projects. On the other hand, Drone offers the same easy âget startedâ experience where you can be up and running in 5 minutes. Drone also benefits from roughly 150 containerized plugins, profoundly extending the functionality of the tool. Drone also scales on demand. All of this means less person hours spent by engineers maintaining the tool or waiting for slowness to resolve, and more time on what matters: getting that code to artifact.

Secrets Management:

Travis CIâs only secrets management option is to encrypt secrets. It doesnât offer any third-party integrations. Drone offers encryption on its open-source version. Meanwhile, the enterprise version offers these alternatives: encrypted, native, or external, through third-party providers such as AWS Secret Manager, Kubernetes Secrets, and HashiCorp Vault. No matter how you want your secrets to be handled, Drone can rise to the occasion.

Pricing:

Travis CI offers a free plan that aligns well with small personal projects or small businesses just starting out. Paid plans are priced by concurrent jobs, billed monthly or annually (at a discounted rate). There is also an enterprise plan that provides an on-premise alternative to SaaS, but itâs a fairly hefty plan that sells licenses in packs of 20, at \$8,000 per pack. This pricing structure is fairly new, and we wanted to share a user review on this new structure: âIt costs us more than half of the credits to make just one build. The price per minute is just insane. When the credits are done, then the CI will just block.â Profound insight into Travis CIâs costs â the provided link above has many reviews pointing out the new price change and how they are switching to a different CI tool. Drone, on the other hand, is open-source software and is available for download. It also has an enterprise version that is extremely feature-rich, but does have pricing attached to it. To familiarize yourself with enterprise pricing, please contact sales.

**Please note: Our competitors, just like us, release updates to their products on a regular cadence. We keep these pages updated to the best of our ability, but there are bound to be discrepancies. For the most up-to-date information on competitor features, browsing the competitorâs new release pages and communities are your best bet.*

Try Harness For Free

Continuous Integration

Interested in seeing what's under the hood? Browse through the Harness Continuous Integration Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/microservices-ci-cd>

GoSpotCheck Masters Microservices CI/CD

Learn how GoSpotCheck creates true Continuous Delivery pipelines using Harness.

About GoSpotCheck

GoSpotCheck's real-time execution management platform empowers field teams to collect merchandising, sales, and compliance data in addition to completing tasks in the field.

CI/CD Before Harness

Deployment pipelines in the past were scripted using CircleCI to support the many AWS ECS microservices at GoSpotCheck.

As a result, these pipelines were complex, limited, and time-consuming to manage, taking 1 engineer a few weeks to create with ongoing support. In addition, pipelines weren't agnostic or portable across clouds, stacks, and platforms.

The overall deployment user experience was also painful, with limited insight into the status of pipelines, making it difficult to troubleshoot and debug. A history of YAML configuration also meant pipelines became brittle; a single typo could stop a pipeline from working!

Deployment verification and health checks were performed manually by the QA team using basic smoke tests in production. Rollbacks would take anywhere from 30 to 60 minutes depending on the microservice team.

The Compelling Event

As GoSpotTeam became more cloud-native, leveraging microservices technologies like Go, containers, and Kubernetes the existing deployment process became too complex to manage. In addition, they lacked portability across different cloud providers. It was at this time the GoSpotTeam decided to evaluate Harness Continuous Delivery Platform.

Integrating Harness

Harness successfully integrated with GoSpotCheck's technology and tools consisting of:

- Go, Docker, & Kubernetes
- Multi-Cloud of Heroku, Google Cloud Platform (GCP), and Amazon Web Services (AWS)
- CircleCI for CI
- HashiCorp Terraform for infrastructure provisioning
- New Relic, Sumo Logic, and Prometheus for monitoring and verification

CI/CD Today

Today, using Harness, GoSpotCheck is able to migrate several applications (and microservices) between cloud providers and regions in hours with zero downtime.

Pipelines are agnostic, portable, and dynamic that requires minimal maintenance by engineers or the DevOps team.

Deployment pipelines now take hours to create vs. days or weeks with the previous CircleCI workflows.

Continuous Verification and automatic rollback prevent engineers from having to understand service configuration and dependencies â they don't need to worry about when or how to rollback. Deployment anomalies and regressions are automatically detected by Harness and its integrations with New Relic, Sumo Logic, and Prometheus.

Development teams can all watch, manage, and troubleshoot deployments together with ease, in real-time, with complete context of what's going on. In short: Harness provides GoSpotCheck with a true Continuous Delivery as-a-Service platform for all developers.

The entire engineering organization has been exposed to Harness, with 3-4 agile teams actively using Harness on a day-to-day basis so

they can continuously deliver.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/comparison-guide/cloudzero-vs-harness>

Comparison Guide

Harness

Cloud Cost Management

vs

CloudZero

Harness Cloud Cost Management vs CloudZero

Cloud Cost Management

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

CloudZero

CloudZero is the cloud cost intelligence platform that puts spend into the context of your business. By aligning engineering, infrastructure, and finance teams around metrics like cost per product feature, customer, and development team, CloudZero enables better strategic decisions, improved unit economics, and efficient spending.

50

2016

\$15.6M

CloudZero raised an \$8M Series A-III (Venture) funding round in Dec 2020

CloudZero is categorized as:

Cloud Cost Management (CCM)

How Does CloudZero Compare?

CloudZero can provide cost visibility across a variety of cloud and SaaS vendors beyond AWS, Azure and GCP, but lacks cost optimization for anything other than AWS. They also require third party tools to provide automated cost savings features that Harness includes as part of a complete solution for cost visibility, optimization and governance.

[CloudZero vs Harness: DevOps Tools Comparison | Harness](#)

Updated

November 30, 2023

- SaaS
- Audience & Primary Users
- Cloud Support
- Time Granularity
- Tag Management Required
- Hybrid Cloud Visibility
- Cloud Account Visibility
- Cloud Region Visibility
- Application Visibility
- Microservice Visibility
- Environment Visibility
- Non-Cluster Visibility (Services)
- Cluster Visibility (Kubernetes/ECS)
- Root Cost Analysis (3 Metrics)
- Cloud Event Correlation (e.g. deploy)
- Cost Recommendations
- Alerting & Budgets
- Anomaly Detection
- Reporting
- Documentation
- Support
- Training
- Automated Idle Resource Management
- Cloud Cost Business Intelligence

<yes><yes>

Engineering & Finance

All

Hourly

<yes><yes> **Optional**

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>
<yes><yes>
<yes><yes>
<with><with> **Utilized, Idle, Unallocated**

<yes><yes>

Engineering

<with><with> **AWS Only**

Hourly

<yes><yes> **Optional**

<with><with> **AWS Only**

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<with><with> **Anything in AWS**

<with><with> **Utilized, Idle**

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<no><no>

SaaS and Self-Hosted

SaaS

AWS, Azure, and GCP

AWS, Azure, GCP + SaaS Apps

<with><with>

Kubernetes Only

<no><no>

Percentage Cloud Spend

Percentage of Cloud Spend

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<with><with>

AWS, GCP Only

<yes><yes>

<yes><yes>

Coming soon

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<yes><yes>

<with><with> via Xosphere Partnership

<yes><yes>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<no><no>
<yes><yes>
<yes><yes>
<yes><yes>
<no><no>
<no><no>
<no><no>
<yes><yes>
<with><with> via ProsperOps Partnership
<yes><yes>
<with><with> via ProsperOps Partnership
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<yes><yes>
<no><no>
<yes><yes>
<yes><yes>
<yes><yes>

<yes><yes>

Detailed Feature Comparison

Harness Cloud Cost Management Vs. CloudZero

Summary

CloudZero provides strong visibility into costs across multiple cloud environments beyond just AWS, Azure, and Google Cloud Platform. They've developed a unique system for importing costs from third-party cloud providers such as Snowflake, NewRelic, MongoDB, Databricks, and more that can give their users a more complete view of their application spend. As with many vendors (like Harness), they can normalize these costs and present them in views that match their users' business contexts.

However, if you want to save money, and not just count it, then you'll need to look beyond CloudZero, as they don't have any automated cost savings features built-in, and their recommendations only apply to AWS (and in fact, you must be an AWS user to use CloudZero.) They've partnered with 2 other firms for cost-savings automation features, but that then requires their customers to manage two or three vendors, not one.Â

Harness is your one-stop for all your cloud cost visibility, optimization, and governance needs, providing native automation that can help you save up to 80% on your cloud costs.Â

Cost Optimization

You don't want to just see your cloud costs, you want to reduce your costs and have the best cloud efficiency possible for your cloud spend across your multi-cloud environments, preferably with strong automation built-in to reduce the burden for your engineering teams.

CloudZero primarily focuses on cost visibility, which has left its cost optimization features lacking. The Cloud Advisor tool that is meant to surface cost savings opportunities remains in beta. Their Insights focus on AWS savings opportunities only, and it's likely you'll get the same level of information from AWS directly. In order to provide automated cost savings, CloudZero requires you to bring third-party vendors on board to provide AWS Spot Instance Orchestration (Xospere) and Savings Plan/Reserved Instance contract management (ProsperOps). They have no native capabilities to automate cloud cost savings.

Harness goes beyond just recommendations to deliver a fully integrated set of tools for automating cloud cost management. Our Commitment Orchestrator automatically manages your savings plan and reserved instance commitments based on your savings goals, and leverages your RI commitments when making resource recommendations. Our Cloud AutoStopping feature allows you to proactively manage idle cloud resources, shutting them down when not in use, and bringing them back up when needed. With Spot Instance Orchestration built in, our Cloud AutoStopping can save customers up to 80% on idle cloud resource spend.

Kubernetes Support

Cloud-native infrastructure relies on Kubernetes to manage shared cluster resources. You need strong insights into cluster usage and savings opportunities to maximize your cluster performance.Â

CloudZero can provide visibility into Kubernetes spend across multiple environments, but it can't give you detailed insights into your Kubernetes clusters outside of AWS. Even on AWS, they require both a cluster agent and AWS CloudWatch in order to gain the telemetry needed to allocate shared costs correctly. They provide some basic recommendations for the underlying cluster infrastructure, but can't help you optimize your node pools to match your actual usage.

Harness gives you deep visibility into your Kubernetes spend across all your multi-cloud environments (including on-premises clusters) with cost perspectives that match the way you do business. Our recommendations are granular to the workload and the node pool configurations to fully optimize your clusters. We can also automate cluster management with Cluster Orchestrator, which can both autoscale clusters as well as integrate native spot instance orchestration for cluster instances.

Cloud Event Correlation

It's a developer's world, and the features and applications that they bring to market drive your company's revenue. But wouldn't it be great to understand the cost of deployments, as well as the impact of new features on your cloud costs?Â

To correlate feature releases or engineering deployments with changes in cloud costs, a cloud cost management tool needs to have context behind an organization's continuous delivery (CD) pipelines. CloudZero provides insight into cost changes as they correlate to deployments via their integrations into continuous integration (CI)/CD tools. However, we are unsure how they assign this context to specific cost anomalies.

Harness ties right into a customer's pipelines to make the correlation between deployments and cost changes as changes happen, with full visibility into where that context is pulled from.

Cloud Cost Management

Interested in seeing what's under the hood? Browse through the Harness Cloud Cost Management Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/meltwater-scaled-pipelines-with-harness>

Meltwater Runs 1200 Pipelines Per Day With Harness CI

Learn how Meltwater builds 1200 artifacts a day.

About

Meltwater helps companies make better, more informed decisions based on insights from the outside. We believe that business strategy will be increasingly shaped by insights from online data. Organizations will look outside, beyond their internal reporting systems to a world of data that is constantly growing and changing. Their customers use these insights to make timely decisions based on real-time analysis.

If it ain't broke! Well, actually it's broke.

Meltwater was losing developers' confidence. The Jenkins instances used to patch together Continuous Integration pipelines lacked autoscaling, meaning developers waited in a line to complete their builds. Meltwater used multiple Jenkins instances, so no one wanted to spend the months of maintenance it would take to fix the issue.

Only 15% of Meltwater's developers were using the corporate-issued Jenkins pipelines. Teams began building their own pipelines using TravisCI and CircleCI. Even with developers using custom pipelines, only 60% of Meltwater's repos were integrated into a CI pipeline.

Developers were happy using one-off CI tools until Meltwater decided to migrate to Kubernetes. Travis CI and CircleCI became security risks because they required Meltwater to externally expose their APIs. Meltwater needed to find an on-premise CI solution.

Simplicity = Adoption

Meltwater chose Harness CI to be the base of their Kubernetes migration. Since moving to Harness CI, adoption rates have increased 3x to 222 developers, which is 50% of the organization. Those developers run 1200 pipelines per day. Harness CI has also increased Meltwater's repo automation from 60% to 75%.

Harness CI's simplicity offered a low learning curve and has since become the defacto language to share best CI practices. Harness CI lets developers create both simple and complex pipelines.

Harness CI's autoscaling feature ensures there are always enough instances to meet demand, while significantly reducing cloud infrastructure cost by spinning down agents when demand decreases. Harness CI plugins become another pipeline step and can be set up instantaneously, without requiring coordination between developers and build administrators.

And if something ever does come up, Meltwater has one of the best open-source communities to reference. For example, when Docker imposed new rate limits, there was a Harness CI solution posted on a forum within a couple of days. For other CI tools, Docker rate limits are still an issue.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/comparison-guide/gremlin-vs-harness>

Comparison Guide

Harness

Chaos Engineering

VS

Gremlin

Harness Chaos Engineering vs Gremlin

Chaos Engineering

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

Gremlin

Gremlin was the first SaaS chaos engineering platform designed to improve web-based reliability. Offered as software-as-a-service (SaaS), Gremlin has 11 attack types that can be used in a chaos experiment.

51-100

2016

28m

Gremlin has raised \$28M in funding over three rounds. Their latest funding was raised on September 28, 2018 from a Series B round.

What is the difference between Harness Chaos Engineering Vs. Gremlin?

Harness Chaos Engineering (CE) is a fault injection tool part of a modern continuous integration and continuous delivery (CI/CD) platform. Harness CE facilitates greater collaboration between SREs, developers, and stakeholders while enabling Continuous ResilienceTM through automating resilience tests in the software delivery pipeline.

Gremlin provides various chaos experiments and test cases but does not provide native integration across the software delivery lifecycle, as evident in the following analysis.

Gremlin vs Harness CE: Chaos Engineering Comparison | Harness

Updated

November 30, 2023

â

<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<no><no>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<yes><yes>
<no><no>
<no><no>
<yes><yes>
<no><no>

Developers, QA Engineers and Testers, SREs, DevOps, Platform Engineers, APIs

SREs

Developers, QA Engineers and Testers, SREs, DevOps, Platform Engineers, APIs

SREs

Continuous ResilienceTM, Continuous Reliability Management, Developer Productivity, Digital Transformation, Customer Experience, IT Disaster Recovery

Reliability Management, Fault Injection

Continuous ResilienceTM, Continuous Reliability Management, Developer Productivity, Digital Transformation, Customer Experience, IT Disaster Recovery

Reliability Management, Fault Injection

92+

11

<yes><yes>
<yes><yes>
<no><no>
<no><no>
<yes><yes>
<yes><yes>
<no><no>

<no><no>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<no><no>
<no><no>
<yes><yes>
<yes><yes>
<no><no>
<no><no>
<yes><yes>
<yes><yes>
<no><no>
<no><no>
<yes><yes>
HTTP, Command, Kubernetes, Prometheus, Service Level Objectives

<yes><yes>
Status Checks

<yes><yes>
Templated and fully customizable

<no><no>
<no><no>
<yes><yes>
Datadog, New Relic, Dynatrace, AppDynamics, Splunk, Google Cloud Operations, Prometheus

<yes><yes>
Datadog

<yes><yes>
<yes><yes>
<no><no>
<no><no>
<yes><yes>
<yes><yes>
<no><no>
<no><no>
<yes><yes>
<yes><yes>
<yes><yes>
Multiple Experiment Setup

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<no><no>

<yes><yes>

Customizable weighted values per experiment

<yes><yes>

Not customizable and scored per service

<yes><yes>

<yes><yes>

<yes><yes>

Additional Add-on

Continuous ResilienceTM

<yes><yes>

<yes><yes>

<no><no>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

Custom

<yes><yes>

Grafana

<yes><yes>

<yes><yes>

<no><no>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<no><no>

<no><no>

<yes><yes>

<yes><yes>

<no><no>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<with><with>
<with><with>
<yes><yes>
<no><no>
<no><no>
<yes><yes>
Automated into AWS S3 buckets

<yes><yes>
Manual

<yes><yes>
2 Years

<yes><yes>
<yes><yes>
<yes><yes>
<yes><yes>
<no><no>
<no><no>
<yes><yes>
<yes><yes>
<no><no>
<no><no>
<yes><yes>
<yes><yes>
<no><no>
<no><no>
<yes><yes>
<yes><yes>

<no><no>

<no><no>

<yes><yes>

<no><no>

<no><no>

<yes><yes>

<yes><yes>

<no><no>

<no><no>

Summary:

Harness and Gremlin provide fault injection and chaos engineering capabilities. Harness Chaos Engineering (CE) is designed to facilitate greater collaboration between SREs, QA engineers, and developers while automating many actions related to resilience testing.

Gremlin rebranded into a Reliability Management vendor that provides failure testing strategies for a strict list of use cases. Gremlin's reliability management features can be useful to SRE teams getting started with chaos engineering. Still, it does not provide automation across the software delivery lifecycle to scale out enterprise reliability, as evident in the following analysis.

Chaos Platform:

Harness supports SaaS and self-managed deployment models, enabling businesses to deploy a chaos platform across every architecture, including high-security platforms with air-gapped solutions. This enables our primary users to be developers, QA Engineers and Testers, SREs, DevOps, Platform Engineers, and APIs. Our fault injection tool allows outcomes and use cases for Continuous ResilienceTM, Continuous Reliability Management, Improve Developer Productivity, Accelerate Digital Transformation, Improve Customer Experience, and IT Disaster Recovery Automation.

Gremlin offers a SaaS version of its product, primarily supports SREs, and uses cases such as reliability management and fault injection. The reliability management platform provides static scheduled snapshots of measured reliability on a defined service.

Chaos Experiments and Faults:

Harness has 92+ experiments to run out of the box, enabling teams to scale quickly across all platforms. Each month, Harness releases more to support new platforms and new failure modes to test. Harness provides an Enterprise ChaosHub that enables users to organize their experiments as well as quickly execute tests and directly schedule from the ChaosHub. The experiment workflow is customizable with a visual workflow builder and a built-in YAML editor that enables flexibility for the user to work.

Harness also supports various experiments across VMware, GCP, Azure, AWS, Kubernetes, and Serverless enabling enterprises to build resilience across multiple environments no matter where their developers work. For platforms not supported, we offer a "Bring Your Own Chaos" experience.

Gremlin has 11 different types of faults that are flexible to run on many platforms. Still, you need to configure them specifically for the type of experiment you want to build requiring a lot of infrastructure knowledge and associated failure modes.

Observability and Steady State Management:

Harness Chaos Infrastructure includes native health metrics for Kubernetes to ensure your infrastructure and experiment are in a healthy

state of operations.

Harness has five resilience probe types that are highly tunable and enable custom configuration for various environments. The probes allow engineers to query HTTP responses, Kubernetes services, Prometheus metrics, and a command probe that can be configured with a custom script. Resilience probes can also be configured to run before, during, and after the execution of an experiment so a user can monitor the precise impact on their system.

Harness has connectors to integrate with other observability tools like Datadog, New Relic, Dynatrace, AppDynamics, Splunk, Google Cloud Operations, and Prometheus. These provide a full integration experience to remove the cognitive load from complex manual integration setup.

Gremlin has a status check feature that allows users to manually set up a health status. It has a basic configuration and doesn't allow the user to be flexible in monitoring experiments.

Experiment Control Methods:

Harness has ensured that we support cloud-native platforms by having a declarative approach to an experiment supporting YAML. We also enable users to run experiments in parallel on multiple Kubernetes clusters and programmatically abort experiments safely.

Gremlin has had a basic attack and scenario control method. It provides serial testing built to support traditional testing but not cloud-native ephemerality. The ability to abort a test is available.

Chaos Orchestration and Reliability Management:

Harness provides multiple orchestration methods, including native integration with Harness CD and a simplified API for integration with other CI/CD tooling. GitOps also uses event-driven chaos injection in which target resources can be configured to automatically trigger chaos experiments with any changes in the resource specification. Currently, the events supported for chaos injection are resource image change, configuration change, change in replicas, and many more. The event-driven chaos injection allows Harness to be integrated with traditional GitOps flow that involves automated deployment of applications or workloads. Chaos experiments can be automatically triggered based on your organization's needs.

Harness also provides simple experiment scheduling and the ability to export an experiment to a new ChaosHub to share experiments with other teams.

Harness provides a resilience score per experiment that can be configured specifically to your needs. Each score can have custom weighted averages configured so you can prioritize which experiments drive the measurement of the score. This can also be leveraged in the GameDay feature

Gremlin provides an API integration to orchestrate experimentation automatically. They have manual scheduling that is simple to turn on.

Chaos Integration:

Harness CE features are built to function within the entire software delivery lifecycle. The native integration with Harness CD is an industry first for enabling developers to automate chaos experiments in a pipeline to achieve Continuous ResilienceTM. For users that use the module standalone, the API is still simple to use with other CD tools.

The native platform integration for Harness also gives you a CD experience that provides automated deployment verification, rollback of failed deployments, and reliability guardrails to make Service Level Objectives and impact from a chaos experiment visible.

Harness has many integrations, such as Dynatrace, Datadog, and New Relic. New integration connectors can be quickly built, and the support keeps improving. We can also support integrations with Feature Flags, other SLO providers, and load-testing tools.

Gremlin has an integration with Datadog for experiment workflows, Jira for ticket creation, and Grafana K6 for load testing. There is no native integration with CI/CD tooling but you can leverage their API to run experiments.

SDKs:

Harness officially supports Python, GoLang, and Ansible SDKs, allowing users to build custom experiments and API wrappers that can be used with custom self-service enterprise solutions. This is important for enterprises building platform solutions as you can easily wrap the Harness API with an SDK to make it self-service.

Gremlin has unofficial SDK support for Python.

Administration, Security, Governance:

At Harness, we take security extremely seriously, as evidenced by our DevSecOps approach. We took the same views with Chaos Engineering, ensuring RBAC and SSO were available from inception, with more controls being added as security postures get updated. We have a comprehensive API to automate every task. Chaos experiment logs can automatically be backed up to an AWS S3 bucket to ensure you have access to logs and reporting.

As part of the platform integration, you also benefit from full audit trails, policy-based governance through Open Policy Agent,

integrated secrets management, and much more.

Gremlin supports RBAC, SSO, and provides audit logs and reporting. They are a SaaS-only provider and can't support a self-hosted model.

Support:

Harness provides SLAs for all of our product modules and platform. If your enterprise needs training and support, we have enterprise options that can be leveraged. We also provide a Community Developer Hub which includes tutorials, documentation and customer driven content that can help you build a community.

Gremlin provides an SLA, training and support.

Unified Software Delivery Platform:

Harness CE is the Chaos Engineering module of the Harness Software Delivery Platform. Each module can be used standalone (integrated as a best-of-breed solution to a DevOps toolchain) or as part of the platform. When used as part of the platform, each module passes meta-data and can provide greater levels of automation than if used standalone.

Gremlin does not offer a unified software delivery platform.

**Please note: Our competitors, just like us, release updates to their products on a regular cadence. We keep these pages updated to the best of our ability, but there are bound to be discrepancies. For the most up-to-date information on competitor features, browsing the competitor's new release pages and communities are your best bet.*

Request a Demo

Chaos Engineering

Interested in seeing what's under the hood? Browse through the Harness Chaos Engineering Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/particle-reduces-human-error-deployments>

Particle Reduces Human Error in Deployments

Harness was the right out-of the box solution to help Particle deploy software more effectively.

About Particle

Particle provides an integrated IoT Platform-as-a-Service that helps businesses connect, manage, and deploy software applications to connected devices, from edge to cloud and back. Particle powers IoT products for hundreds of businesses â from fast-growing startups to Fortune 100 companies â with more than 240,000 developers for IoT product development.

Particleâs expertise goes beyond world-class technology, enabling next-generation business intelligence, insights, and expert customer support to ensure IoT projects succeed.Â

Challenges: Homegrown Continuous Delivery Limitations

Several years before Debbie Gillespie, Engineering Manager, took over Particleâs infrastructure and site reliability engineering team, Particle lacked a consistent way to deploy software. Particle spun up a Slack chatbot tool that allowed developers to deploy to various environments. This was crucial for the growth of the small startup; however, fast forward to present day, and the custom ChatOps tool was no longer scaling.Â

In a single production deployment, a developer might have to type out five different commands to deploy a single service, each requiring a specific order to be successful. All this was required to complete a single production deployment. This left the door open to human error.Â

Once a command was executed, there was a manual process to validate success or failure. The only way to see if a change took place was to open up a kubectl â the Kubernetes command-line tool â and sift through hundreds of lines to locate an update. This lack of feedback from the tool frustrated developers and created vulnerabilities for the business.Â

For instance, when attempting to roll back a failed deployment, there was no indication of whether the rollback was successful or not. In one instance, a developer tried rolling back two to three times, but they couldn't tell if the rollback had worked. They instead had to go

manually redeploy an old artifact.â

Gillespie wanted to provide Particleâs developers with a better CI/CD experience, but her team was relatively small, and they didnât have the bandwidth to update the existing tool to meet their needs. She estimated it would have taken two engineers eight months to update their existing solution. There was no way that upper management would approve the project given the resource constraints.â

âWe are not a CI/CD company. CI/CD is not paramount to what we do for our customers, itâs what we use to deliver value to our customers,ââ Gillespie said. âI wanted an out-of-the-box solution that would provide a great experience for our developers.ââ

Solutions: Harness Provides the Best CD Out-of-the-Box Experience

Gillespie chose Harness to conduct Particleâs software deployments. Harness handles getting an image out with the correct sequencing, so developers donât have to worry about manually deploying artifacts to the right order of stages.â

The Harness UI lets developers know exactly whatâs happening at any given moment with their deployments. If something fails, Harness notifies the user what went wrong, where it went wrong, and gives the user the option to immediately roll back if needed. This visibility extends to managers, who no longer have to search through Slack to find deployment records. Instead, Harness Audit Trails has everything in one place.

Harness was the right out-of the box solution to help Particle deploy software more effectively.

âHarness gives us a top-class deployment solution that we know will be continually invested and innovated on,â Gillespie said.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/comparison-guide/nobl9-vs-harness>

Comparison Guide

Harness

Service Reliability Management

VS

Nobl9

Harness Service Reliability Management vs Nobl9

Service Reliability Management

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

Nobl9

Nobl9 helps software developers, DevOps practitioners, and reliability engineers deliver reliable features faster. Through software-defined Service Level Objectives (SLOs) they help you link monitoring and other logging and tracing data to user happiness and business KPIs.

97

2019

\$28.5M

Nobl9 is categorized as:

SLO software and Tools Management

What is the difference between Service Reliability Management vs. Nobl9?

Nobl9 vs Harness: Service Reliability Management Tool Comparison

Updated

November 30, 2023

- **SaaS & On-Premises**
- **Integrates with Change and Incident Management Sources**
- **Integrates with Leading Observability Solutions**
- **Custom Reliability Policies**
- **Reliability Guardrails Integrated with CI/CD Pipelines**
- **Service Health Indicators**
- **Service Reliability Checks**
- **Automated Deployment Verification**
- **Automated Rollback of Failed Deployments**
- **Correlation of Change Events to Service Health**
- **Correlation of Incidents to Service Health**
- **Error Tracking**
- **Reliability Audit Trails**
- **Fine-Grained Role-Based Access Control**
- **Unified Software Delivery Platform**
- **Composite SLOs**

<yes><yes>

<yes><yes>

<yes><yes>

Notification + Pipeline

<yes><yes>

<yes><yes> SLO or Metric

<yes><yes>

<yes><yes>

<yes><yes> WithÂ Harness CD

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<with><with> **On-Prem Only**

<with><with> **Minimal - annotations via UI or API**

<yes><yes>

<with><with> **Notifications Only**

<no><no>

<with><with> SLO based

<no><no>

<no><no>

<no><no>

<with><with> **Minimal - annotations via UI or API**

<with><with> **Minimal - annotations via UI or API**

<no><no>

<no><no>

<yes><yes>

<no><no>

<yes><yes>

<yes><yes>

<yes><yes>

<with><with> Minimal - annotations via UI or API

<yes><yes>

<yes><yes>

Notification + Pipeline

<with><with> Notifications Only

<yes><yes>

<no><no>

<yes><yes> SLO or Metric

```
<with><with> SLO based  
<yes><yes>  
<no><no>  
<yes><yes>  
<no><no>  
<yes><yes> With Harness CD  
<no><no>  
<yes><yes>  
<with><with> Minimal - annotations via UI or API  
<yes><yes>  
<with><with> Minimal - annotations via UI or API  
<yes><yes>  
<no><no>  
<yes><yes>  
<no><no>  
<yes><yes>  
<yes><yes>  
<yes><yes>  
<no><no>  
<yes><yes>  
<yes><yes>
```

Summary:

Harness and Nobl9 both provide SLO management capabilities. Harness Service Reliability Management (SRM) is a SLO management module that is part of a larger software delivery (CI/CD) platform. Harness SRM is designed to facilitate greater collaboration between SREs and Developers while also automating many actions related to SLO management like governing software deployments.

Nobl9 is useful to SRE teams but does not provide automation across the software delivery lifecycle as evident in the following analysis.

SaaS & On-Premises:

Harness SRM is available as either SaaS or self-managed software that can be deployed anywhere.

Nobl9 is available via SaaS and provides an on-prem option that can be installed only in AWS.

Integrates with Change and Incident management sources:

Harness SRM shows deployments, Kubernetes change events, and incidents directly in context with SLO and Error budget changes. Harness allows you to drill down into these changes to determine the root cause of SLO violations.

Nobl9 has an API that allows users to add annotations to SLO charts. No further analysis is offered.

Custom Reliability Policies:

Harness SRM contains a built in policy engine based on Open Policy Agent (OPA). This enables Harness users to define and enforce custom policies with via UI or via policy-as-code. Policies within Harness can be used to send notifications or to actively manage software delivery pipelines via reliability guardrails.

Nobl9 offers the ability to create policies for notification purposes only.

Reliability Guardrails integrated with CI/CD pipelines:

Harness SRM reliability guardrail policies are used to automatically determine if deployment pipelines should proceed or stop given the

status of SLOs and Error Budgets. This makes it possible to effectively standardize and scale the management of pipelines via SLO management processes.

Nobl9 does not offer this capability.

Service Health Indicators:

Harness SRM can determine and display the health of application services using SLOs and/or metrics to derive the overall health. By using a combination of SLOs and metrics, Harness can provide a more precise calculation of service health.

Nobl9 can determine and display the health of application services using SLOs only.

Service Reliability Checks:

Harness SRM can determine the health of services across all stages of software delivery by using a combination of AI/ML techniques on logs and metrics and through error tracking. These capabilities make it possible for developers to proactively improve the quality and reliability of their application services before deploying to production.

Nobl9 does not offer service reliability checks.

Automated Deployment Verification and Rollback (Continuous Verification):

Continuous Verification is the process of monitoring your app for abnormalities after a deployment. For example, Continuous Verification could catch a latency issue or 5xx errors and automatically roll back your app to the previous version. The idea is to catch errors as quickly as possible – ideally, before customers notice – and make a seamless transition back to the prior version.

Harness provides Continuous Verification out of the box, effectively reducing risk and reputational damage from downtime. Harness supports many vendors, including Prometheus, Datadog, AppDynamics, New Relic, StackDriver, CloudWatch, and custom monitoring and observability tools.

Nobl9 does not offer this capability.

Correlation of Change Events to Service Health:

Harness SRM ingests change events and shows them on a timeline that is aligned with SLO and Error Budget charts. The user can select and drill down into specific timeframes to perform root cause analysis on the impacted SLOs.

Nobl9 offers the ability to annotate change events on SLO charts but does not offer any further drill down or analysis.

Correlation of Incidents to Service Health:

Harness SRM ingests incident events and shows them on a timeline that is aligned with SLO and Error Budget charts. The user can select and drill down into specific timeframes to perform root cause analysis on the impacted SLOs.

Nobl9 offers the ability to annotate incident events on SLO charts but does not offer any further drill down or analysis.

Error Tracking:

Harness SRM is capable of detecting all run-time exceptions within Java applications and providing detailed debugging information so that developers can resolve problematic exceptions before they deploy to production. This improves the quality of software which also improves reliability over time.

Nobl9 does not offer error tracking capabilities.

Reliability Audit Trails:

Harness SRM keeps detailed audit logs of all activities including creating, editing, and deleting of SLIs, SLOs, and Error Budgets. This is in addition to the standard audit trails included with the Harness platform.

Nobl9 does not offer audit trails.

Unified Software Delivery Platform:

Harness SRM is the SLO management module of the Harness Software Delivery Platform. Each module can be used standalone (integrated as a best of breed solution to a DevOps toolchain) or as part of the platform. When used as part of the platform, each module passes meta-data and can provide greater levels of automation than if used standalone.

Nobl9 does not offer a unified software delivery platform.

**Please note: Our competitors, just like us, release updates to their products on a regular cadence. We keep these pages updated to the best of our ability, but there are bound to be discrepancies. For the most up-to-date information on competitor features, browsing the competitor's new release pages and communities are your best bet.*

[Request a Demo](#)

Service Reliability Management

Interested in seeing what's under the hood? Browse through the Harness Service Reliability Management Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/metrikus-commit-to-prod-time>

Metrikus Shortens Commit-to-Production Time by 66%

Metrikus made feature flag management easy with Harness. See how you can too.

About Metrikus

Metrikus gathers information from building systems and sensors to help their customers understand their buildings more efficiently. They work alongside industry experts to bring customers the best solutions with the most meaningful output, aggregating data across 4400+ different connectors. Metrikus is all about removing barriers, bringing data together, and putting it into the hands of the people who need it. It is their mission to digitise the entire built environment leading to smarter, healthier, and more sustainable spaces.

Sam Hall, Head of Technology at Metrikus, led the adoption of feature flags across the organization. This is their story.

Too Much Risk, Too Slow Of A Release Process

To put it simply, Metrikus needed to be able to reduce the risk of feature deployment while simultaneously increasing the velocity at which they shipped code.

Where many of their peers can accept a 0-15% change failure rate, for Sam and Metrikus, this tolerance is nonexistent: they need to essentially hit 0%. Because of how much data they aggregate and how sensitive changes can be, any deployment carries with it high levels of risk. And as it is core to their business, it compounds the need to get it right.

Their process was governed by a monthly release cycle, so each deployment was a âbig rocket launch,â and the Metrikus team wanted to get away from that. Not only did it introduce more risk, but it was a reflection of how long customers had to wait to get changes or new features in their hands.

At the same time, it meant that product releases were solely the responsibility of Engineering, when that responsibility needed to be shared with Product and Marketing. Sam wanted to decouple engineering pushes to production from the actual feature release to the customer, which would reduce inefficiencies on launch day and give non-Engineering teams control over the customer experience.

As part of the solution, Sam and team wanted to do smaller, more controlled releases. They wanted to be able to test multiple versions of a feature, to QA features in production, and to reduce the effort and risk associated with each release. With their existing setup, if they wanted to roll out a feature to a small subset of customers, they would need to use complex deployment patterns such as blue-green or canary, or suffer a moment of downtime. And then, theyâd need to deal with rollback mechanisms â a set of problems on their own! Even if they did adopt these advanced deployment patterns, they still wouldnât have the desired level of feature control.

Based on this assessment, the Metrikus team chose to pursue feature flags as the solution of choice to solve their issues â but the right solution needed to elegantly solve for their complex needs. When they found Harness Feature Flags, they were thrilled with the results.

Getting Started With Harness Feature Flags

Sam and team chose Harness as their feature flag solution. It was important to them to easily integrate their feature flag solution into their code, and fit into existing processes to create a good developer experience. With Harness, thatâs exactly what they got. Integrating the right SDK âwas as easy as it could possibly be,â according to Sam, and the feature workflows allowed everything to flow nicely and seamlessly into what they had in place. To get started with Harness Feature Flags, it took the team â6 commits and a few hundred lines of code.â

Since part of their value proposition is efficient data aggregation, naturally, Metrikus Â ingests lots of data from their 4400+ connected data points. Because so much is dependent on service state (and state changes rapidly with so much data), the team found that other tools introduced too much latency due to bloat, making it more difficult to deliver on their core value and to effectively instrument a scalable feature flag solution that met all of their needs.

And because of the scale at which they needed to operate, in addition to latency concerns, it was important that a feature flag solution work nicely with their CI/CD pipeline â a concern they found that was alleviated by Harness, as the platform integrates feature flags into

the CI/CD pipelines! In Sam's words: "The other feature flag tools we evaluated wouldn't have scaled with us, we would have had to transition off of them as we grew."

The Results

Harness Feature Flags has allowed Metrikus to deploy code faster with less risk. There are no more "big rocket launches" and they are able to deploy and test small incremental changes. They've been able to **cut down their lead time for change by a factor of 3x, or 66%.** When speaking to the difference, Sam mentioned their new Wellness Panel, stating that previously, it would have taken around 3 months to start trialling with customers. Instead, they were able to do their trial release in month 1, incorporate feedback in month 2, and release on schedule in month 3. It is allowing us to release features earlier to a set number of customers.

With Harness, Metrikus is able to test different versions of features in production and choose the best working version for continued use. This lets them try more things or take more risks (ironically enough).

Now at Metrikus, responsibility for releases has been diversified from just the Engineering organization to Product and Marketing. This allows other sides of the business to decide the opportune moments to release new features, and frees up Sam's Engineering team to focus on building the right solutions for their customers, while the people closest to the customers can ensure they get what they need, when they need it.

With a better developer experience, more feature releases, and less risk to the business, Metrikus is well on their way to creating the technology platform that they have envisioned. With improvements like these just a few commits away, why don't you give Harness Feature Flags a spin? Try it today.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/kubernetes-migration>

Tilting Point Migrates to Kubernetes With One Developer and Harness

Find out how Tilting Point sped up their cloud migration with Harness.

About

Tilting Point is a leading, award-winning free-to-play games publisher that specializes in growing the games of independent developers. Their most successful games include Star Trek Timelines, Warhammer: Chaos & Conquest, and SpongeBob: Krusty Cook-Off. The last

one achieved top 10 standing in 97 different countries when it launched and is still enjoyed by millions of players, along with their extended catalog of games. Tilting Point employs a data-driven approach to game development and user acquisition, powering up the existing live games of independent developers who may become closer partners on a new joint project as their relationship strengthens.

Growing Upâ' Beanstalk to Kubernetes

Tilting Point's Lead Software Engineer, Evan Thomas, used Amazon Elastic Beanstalk and Jenkins to deploy the company's flagship application. Evan built CI/CD pipelines in Jenkins that allowed Tilting Point to deploy twice a week. But as the company progressed, Elastic Beanstalk and Jenkins started to show their limitations.

The deployment pipelines were only capable of risky rolling deployments. If something went wrong in production, it wouldn't be easy to roll-back. Monitoring alerts integrated into the pipelines returned 10% false positives, resulting in tedious troubleshooting. Evan was the only Tilting Point engineer capable of building new pipelines and maintaining existing ones, meaning Evan was spending 20% of his time babysitting Jenkins.

Tilting Point decided it was time to move from Elastic Beanstalk to Kubernetes. They knew Kubernetes would help them scale their applications and eventually reduce the number of manual hours needed to maintain software delivery. But Tilting Point didn't have a dedicated DevOps team or the bandwidth to hire resources to support the migration. They needed a solution to help kickstart the migration.

Harness Gives One Developer the Power of a Full DevOps Team

Evan chose Harness to kickstart Tilting Point's Kubernetes migration. Evan completed his first deployment on the first day. Harness has been empowering Tilting Point to modernize and mature their applications.

Harness provides out of the box canary deployments and reduces Evan's time spent maintaining CI/CD. Because of Harness, Tilting Point was able to forgo hiring a DevOps engineer saving well over \$100,000 in labor costs. The combination of Harness and Kubernetes allows Tilting Point to deploy every day, a 3x increase in deployment frequency.

Harness helped create Tilting Point's new deployment process. The Harness support team continues to partner with Tilting Point to level up their software deployments. Â

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform®

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/eliminate-delivery-toil>

ABC Fitness Eliminates Delivery Toil

Learn how ABC Fitness drastically reduced the effort spend on software delivery.

About ABC Fitness

At ABC Fitness Solutions, they keep the fitness industry moving forward. Their innovative software solutions empower fitness leaders to focus on what matters the most: building relationships and improving the lives of their club members. With their industry-leading payment processing and billing solutions, streamlined membership portals, and partnership services, the work ABC Fitness does is shaping the future of fitness. Their journey to being the primary SaaS solution for fitness clubs is made possible by the extremely passionate and talented professionals who propel their company and software solutions forward.

Jenkins? More like Jankinsâ

Developers want to build things that either improve customersâ lives or improve other developersâ lives. The DevOps team at ABC Fitness didnât have time for either. Engineers Chris Jowett and David Leonard instead spent a significant amount of time maintaining and troubleshooting Jenkins.

ABC Fitness built its software deployment process using Jenkins and created a DevOps team to manage and administer the Jenkins pipelines. The DevOps teamâs original goal was to increase deployment velocity and developer productivity, but that priority had to be shifted to keeping Jenkins from exploding. The DevOps team spent a monthâs worth of work every quarter fixing broken pipelines and maintaining scripts. Jenkins pipeline failures began eroding developersâ deployment confidence.

David and Chris chased down defect after defect in Jenkins. At some point, Jenkins became the butt of a bad inside joke.

jank-y | *adjective* | Ă jaNGkĂ | of extremely poor or unreliable quality. âthe software is pretty jankyâ

ABC Fitness needed to find a better way to deliver code to customers.

Harness Relieves Deployment Burden

When ABC Fitness switched from Jenkins to Harness, they instantly reduced the troubleshooting and maintenance effort associated with software delivery. That effort has been refocused to improving the companyâs platform.

Developers have regained confidence in the DevOps team by tracking deployment success with the metrics Harness provides. This confidence has led to a 20% increase in deployment velocity.

Most importantly, Harness has lifted a major burden from David and Chrisâs shoulders.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform^{Â®}

Need more info? Contact Sales

Comparison Guide

Harness

Service Reliability Management

VS

Sloth

Harness Service Reliability Management vs Sloth

Service Reliability Management

500-1000

2016

\$425M

Harness is categorized as:

Continuous Integration
Continuous Delivery
Cloud Cost Management
Cloud Cost Optimization
Feature Flags
Service Reliability Management
Security Testing Orchestration
Chaos Engineering
Software Engineering Insights

Sloth

Sloth generates understandable, uniform and reliable Prometheus SLOs for any kind of service. Using a simple SLO spec that results in multiple metrics and multi window multi burn alerts.

â

2021

â

Sloth is categorized as:

Prometheus SLOs

What is the difference between Service Reliability Management vs. Sloth?

Sloth vs Harness: Service Reliability Management Tool Comparison

Updated

November 30, 2023

- **SaaS & On-Premises**
- **Integrates with Change and Incident Management Sources**
- **Integrates with Leading Observability Solutions**
- **Custom Reliability Policies**
- **Reliability Guardrails Integrated with CI/CD Pipelines**
- **Service Health Indicators**
- **Service Reliability Checks**
- **Automated Deployment Verification**
- **Automated Rollback of Failed Deployments**
- **Correlation of Change Events to Service Health**
- **Correlation of Incidents to Service Health**
- **Error Tracking**
- **Reliability Audit Trails**

- **Fine-Grained Role-Based Access Control**
- **Unified Software Delivery Platform**
- **Composite SLOs**

<yes><yes>

<yes><yes>

<yes><yes>

Notification + Pipeline

<yes><yes>

<yes><yes> SLO or Metric

<yes><yes>

<yes><yes>

<yes><yes> WithÂ Harness CD

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<yes><yes>

<with><with> On-Prem Only

<with><with> **Required Grafana**

<with><with> **Prometheus Only**

<no><no>

<no><no>

<with><with> SLO based

<no><no>

<no><no>

<no><no>

<no><no>

<no><no>

<no><no>

<with><with> Grafana-based

<no><no>

<yes><yes>

<yes><yes>

<with><with> On-Prem Only

<yes><yes>

<with><with> Required Grafana

<yes><yes>
<with><with> Prometheus Only
Notification + Pipeline
<no><no>
<yes><yes>
<no><no>
<yes><yes> SLO or Metric
<with><with> SLO based
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes> With Harness CD
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<no><no>
<yes><yes>
<with><with> Grafana-based
<yes><yes>
<no><no>
<yes><yes>
<yes><yes>

Summary:

Harness and Sloth both provide SLO management capabilities. Harness Service Reliability Management (SRM) is a SLO management module that is part of a larger software delivery (CI/CD) platform. Harness SRM is designed to facilitate greater collaboration between SREs and Developers while also automating many actions related to SLO management like governing software deployments.

Sloth is an open-source do it yourself solution that provides value to SREs but requires more effort and has fewer features than all competing vendor solutions. Sloth could be a good option for SREs who are looking to heavily customize their SLO management solution since they can modify the source code as needed.

SaaS & On-Premises:

Harness SRM is available as either SaaS or self-managed software that can be deployed anywhere.

Sloth is available as an on-prem solution only.

Integrates with Change and Incident management sources:

Harness SRM shows deployments, Kubernetes change events, and incidents directly in context with SLO and Error budget changes.

Harness allows you to drill down into these changes to determine the root cause of SLO violations.

Sloth is dependent on Grafana. Grafana can integrate with change and incident management solutions.

Integrates with leading observability solutions:

Harness SRM integrates with many of the leading monitoring, logging, and observability solutions to collect the data needed to build and track SLIs, SLOs, and Error Budgets.

Sloth only integrates with Prometheus.

Custom Reliability Policies:

Harness SRM contains a built in policy engine based on Open Policy Agent (OPA). This enables Harness users to define and enforce custom policies with via UI or via policy-as-code. Policies within Harness can be used to send notifications or to actively manage software delivery pipelines via reliability guardrails.

Sloth does not offer the ability to create reliability policies.

Reliability Guardrails integrated with CI/CD pipelines:

Harness SRM reliability guardrail policies are used to automatically determine if deployment pipelines should proceed or stop given the status of SLOs an Error Budgets. This makes it possible to effectively standardize and scale the management of pipelines via SLO management processes.

Sloth does not offer this capability.

Service Health Indicators:

Harness SRM can determine and display the health of application services using SLOs and/or metrics to derive the overall health. By using a combination of SLOs and metrics, Harness can provide a more precise calculation of service health.

Sloth can determine and display the health of application services using SLOs only.

Service Reliability Checks:

âHarness SRM can determine the health of services across all stages of software delivery by using a combination of AI/ML techniques on logs and metrics and through error tracking. These capabilities make it possible for developers to proactively improve the quality and reliability of their application services before deploying to production.

Sloth does not offer service reliability checks.

Automated Deployment Verification and Rollback (Continuous Verification):

âContinuous Verification is the process of monitoring your app for abnormalities after a deployment. For example, Continuous Verification could catch a latency issue or 5xx errors and automatically roll back your app to the previous version. The idea is to catch errors as quickly as possible â ideally, before customers notice â and make a seamless transition back to the prior version.

Harness provides Continuous Verification out of the box, effectively reducing risk and reputational damage from downtime. Harness supports many vendors, including Prometheus, Datadog, AppDynamics, New Relic, StackDriver, CloudWatch, and custom monitoring and observability tools.

Sloth does not offer this capability.

Correlation of Change Events to Service Health:

Harness SRM ingests change events and shows them on a timeline that is aligned with SLO and Error Budget charts. The user can select and drill down into specific timeframes to perform root cause analysis on the impacted SLOs.

Sloth does not offer this capability.

Correlation of Incidents to Service Health:

Harness SRM ingests incident events and shows them on a timeline that is aligned with SLO and Error Budget charts. The user can select and drill down into specific timeframes to perform root cause analysis on the impacted SLOs.

Sloth does not offer this capability.

Error Tracking:

Harness SRM is capable of detecting all run-time exceptions within Java applications and providing detailed debugging information so

that developers can resolve problematic exceptions before they deploy to production. This improves the quality of software which also improves reliability over time.

Sloth does not offer error tracking capabilities.

Reliability Audit Trails:

Harness SRM keeps detailed audit logs of all activities including creating, editing, and deleting of SLIs, SLOs, and Error Budgets. This is in addition to the standard audit trails included with the Harness platform.

Sloth does not offer audit trails.

Fine-Grained Role-Based Access Control:

Harness enables full customization of role based access controls (RBAC) with the SRM module and across the larger platform. Harness RBAC was built to offer the ultimate in flexibility as required by the largest enterprises.

Sloth relies on Grafana's permissions capabilities which are not customized for SLO management.

Unified Software Delivery Platform:

Harness SRM is the SLO management module of the Harness Software Delivery Platform. Each module can be used standalone (integrated as a best of breed solution to a DevOps toolchain) or as part of the platform. When used as part of the platform, each module passes meta-data and can provide greater levels of automation than if used standalone.

Sloth does not offer a unified software delivery platform.

**Please note: Our competitors, just like us, release updates to their products on a regular cadence. We keep these pages updated to the best of our ability, but there are bound to be discrepancies. For the most up-to-date information on competitor features, browsing the competitor's new release pages and communities are your best bet.*

Request a Demo

Service Reliability Management

Interested in seeing what's under the hood? Browse through the Harness Service Reliability Management Product

See how Harness stacks up against these other tools.

Ready To Get Started?

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/lessonly-by-seismic-harness-cie>

Lessonly by Seismic Says Goodbye to Jenkins and Toil With Harness CI Enterprise

Learn how Lessonly by Seismic went from toil with Jenkins to peace of mind with Harness CIE.

About

Lessonly by Seismic (acquired in August 2021) is a powerfully simple training, coaching, and enablement solution that helps teams ramp faster, deliver effective feedback, and continuously improve. With Lessonly by Seismic, more than four million learners at 1,200+ leading companies—including Scholastic, Jostens, and U.S. Cellular—share knowledge, develop skills, and reinforce best practices. The results? Higher NPS scores, more deals closed, and a superior customer experience. To learn more, visit www.lessonly.com.

Back for More

Lessonly by Seismic is not new to Harness. About a year ago, its product team migrated to Harness CD after having a sporadic CI/CD process using Jenkins and Spinnaker. Using Jenkins as their CI server, they felt the pain from managing it with all the server maintenance issues, downtime, and developer toil. Lessonly had two people managing the patched-together server. Rather than becoming pros at managing all of the infrastructure for the tool for the rest of their careers, they decided there had to be a better way.

With the successful migration to and investment in Harness CD, Lessonly looked at Harness CI Enterprise. The team saw the value of a single-pane-of-glass CI/CD solution where a developer could follow an artifact from creation all the way to deployment in multiple environments.

A Tool Made for Developers

With Harness CI Enterprise, Lessonly is easing the burden of the development process from start to finish by providing better logging and exposability around building artifacts. With more developers involved and aware of the full process, managers and individual contributors alike are empowered to do their jobs more efficiently and provide a far better developer experience. Using Jenkins, Lessonly only allowed two admin usersâbut with Harness CI, the team granted access to over 30 developers. This gave developers ownership and the ability to track the progress of their application throughout the build, test, and deploy processes. Even the SRE team noticed that developers were able to handle build failures on their own!

The SRE team is able to easily and painlessly onboard developers to Harness CI. Whenever developers have questions, the SREs are able to quickly point them in the right direction and have them create the pipelines. They consistently praise Harnessâ easy-to-use UI and validate that itâs a better developer experience for the entire team. Out-of-the-box Harness CI comes with a YAML editor and Visual Pipeline Builders so that developers, regardless of skill level, can create build pipelines. The SRE team at Lessonly loves this feature; itâs super helpful to see how to create pipelines in both views. This toggle feature, along with pre-built steps that are baked into the tool, allowed for easy initial setup.

Harness CI allowed Lessonly by Seismic to better empower their developers so that they could own the entire application delivery processâwhich we aim to do, one dev team at a time.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform^{Â®}

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/empower-developers>

Vuclip Empowers Developers With Harness

Vuclip found that Harness was the only solution that fully integrated with their GCP, Kubernetes, APM, and Log stack.

About Vuclip

Vuclip is a leading global technology-driven media company delivering on demand entertainment to emerging markets including India, Southeast Asia, Africa and the Middle East.

Vuclipâs market success in on demand entertainment is built on its leadership at the intersection of technology, consumer insights and media.

Viu is a leading OTT Video service by PCCW, and is available in 15 markets with 16 million active users. Viuâs unique value proposition of fresh and localized regional and local premium TV shows, movies, and originals entertains millions of consumers every day.

CI/CD Before Harness

Vuclip has six development teams that deploy to production twice a week.

Deployments in the past were handled by a centralized deployment team that would maintain Jenkins and Chef deployment scripts.

An average rolling deployment would take 4 people approximately 6 hours each to deploy, verify, and complete. Deployment verification was done manually by engineers looking at New Relic and ELK stack.

Production rollbacks would occur roughly once a quarter and would take ~2 hours due to the manual process and scripts.

With 2 deployments per week, Vuclip was spending almost 2,500 engineering hours per year deploying and verifying new application versions.

The Compelling Event

Like many organizations, Vuclip decided to transition from monolithic Xen based on-premises applications to cloud-native microservices running in Google Cloud Platform (GCP) using Docker and Google Kubernetes Engine (GKE).

Automating their existing deployment process to support this migration, therefore, became critical.

Evaluating Harness, Open-Source and ARA Solutions

Vuclip evaluated leading open-source CD tools and commercial ARA software and Harness came out on top.

Harness's ability to pass any configuration to Kubernetes, their support for init containers, daemon sets, HPA and also tight integration with New Relic and ELK was not available out of the box with any other solution.

The Harness user interface also stood out. It was simple, clean, and more intuitive â an advantage that would accelerate the adoption of CD within Vuclip. Overall, Harness required less setup, configuration, and maintenance than other solutions.

Automating Deployment Pipelines and Verification

With Harness, Jishnu is able to now empower individual development teams at Vuclip so they can deploy, verify, and rollback on their own. The centralized deployment team manages all deployment pipelines, templates, and configuration.

What took 4 people 6 hours (24 hours) each to rolling deploy & verify now takes 2 people 2 hours each (4 hours) using canary deployments from Harness. In addition, rollbacks have been fully automated from 2 hours to just 15 minutes.

The Harness ROI

With Harness, Vuclip was able to accelerate their cloud migration and CD project by 4 months.

Over the first twelve months Vulclip expects to save ~2,080 hours (equivalent to \$200,000 annual savings) of engineering time spent deploying, verifying deployments and maintaining a homegrown solution.

Vuclip worked together with the Harness customer success team so that development teams were onboarded and early adoption of their CD platform was possible.

âHarness has very good people. They listened, partnered, and advised our team so we know where we're going with microservices and serverless in the future,â said Jishnu.

In its first month, during a production deployment, Harness was able to automatically identify a slow running critical business transaction in Vuclip's application (via New Relic), and was able to automatically rollback to the previous working version in just 15 minutes. Over the first year, Harness rollback alone is expected to save 420 minutes of unplanned downtime.

Without Harness, Vuclip would have needed to invest several engineers to build their own basic CD capabilities, with at least one engineer per year to maintain those capabilities. For less than a quarter of the cost of a DevOps engineer, Vuclip was able to leverage Harnessâ CD-as-a-Service.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/discover-dollar-saves-on-cloud-bill>

Discover Dollar Saves 70% On Their Cloud Bill

Discover Dollar made idle cost management easy with Harness. See how you can too.

About

Discover Dollar helps enterprises identify and resolve overpayments and leakages by analyzing a vast amount of data, including unstructured data like emails, with Machine Learning and AI. Their solution is sought after by Fortune 500 companies like Target and Metro and they've recovered over \$300 million to date! Additionally, they have won many awards, including:

Too Much Wastage, Too Much Cloud Spend

When Harness met Discover Dollar, they were using AWS and Digital Ocean, and that involved lots of testing. Most of their experimental servers were very large (200 GB RAM) and the price to run them was exorbitant.

Discover Dollar was enamored with Harness' Intelligent Cloud AutoStopping feature, which automatically stops resources that are no longer in use and restarts them when needed. They had actually tried building their own solution in house, as prior to Harness, there wasn't a viable auto-stopping solution on the market.

With their homegrown solution, they had to take snapshots of their running instances every morning and evening so that they could terminate them and avoid dealing with idle costs. They didn't get charged for terminated instances, but still got charged for shut-down instances, which is why they had to do this.

Because of all of this, Dheemanth was thrilled when he found out about Harness - the solution was perfect for what they needed!

Getting Started With Intelligent Cloud AutoStopping

Implementing Harness Cloud Cost Management, in Dheemanth's own words, was one of the easiest setups I've seen. The fact that everything is fully automated blew Dheemanth away - he believes innovation like this should be native functionality on any cloud cost management platform.

In fact, Dheemanth recommends Harness Cloud Cost Management to a wide spectrum of organizations, including companies just starting off, and large companies with thousands of instances to manage.

Getting set up was simple for the Discover Dollar team. For Intelligent Cloud AutoStopping, they simply needed to set up the gateway for the resources they wanted managed, set up routing, and voila! They were on the path to no idle cloud costs. Initial setup is complete within a few hours, and at that point, you start reaping the benefits on your cloud bill.

When it comes to cloud cost, Dheemanth had a valuable tidbit to share:

"With cloud computing, what you mostly pay for is not what you use, but what you've allocated that sits idle. It can be very tricky to cut that waste down and optimize it. Intelligent Cloud AutoStopping is a must for beginners and large teams alike, bringing a huge benefit to cash flow. I also love spot orchestration - it brings savings to our production workloads."

The Results

When we talked to Dheemanth, we weren't surprised to hear about the massive cloud bill savings that Harness Cloud Cost Management made possible for Discover Dollar — results in line with Harness's expectations.

Best of all, the experience was seamless for the developers. Harness handled turning off idle instances in the background, eliminating the manual toil they were undertaking. And developers could work normally, as if the instances were always online, because Harness automatically detected incoming load and turned them back on just in time. It meant Discover Dollar could reduce their cloud costs without any impact on the developer experience.

With savings like that, and a sharp decrease in pain and toil, why don't you give Harness Cloud Cost Management — and its valuable Intelligent Cloud AutoStopping feature — a spin? Try it today.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/deployment-frequency-and-time>

Marketron Triples Deployment Frequency & Decreases Deployment Time 75%

Find out how Marketron tripled velocity with Harness.

About

Marketron is a leading provider of enterprise revenue and profitability management solutions for radio, television and digital outlets. The company offers revenue generation and management solutions, ad tech and mobile advertising platforms, and an array of digital audience engagement tools that drive new growth opportunities.

Faster! But Not Fast Enough

Marketron struggled to deliver software. The company deployed its legacy applications twice a year using Jenkins and manual executions. In a way, this was better than the alternative: website outages that could impact revenue. ^

Enter Mike Jackson, the new SVP of Software Engineering. Mike was tasked with modernizing Marketron's deployment strategy. Using DevOps principles, Mike increased deployment frequency to once a quarter. A 2x increase. But not fast enough. These quarterly deployments required a 4-hour downtime window of straight developer labor. On top of that, Marketron wanted to begin migrating its legacy applications to AWS and Kubernetes. Mike's team was lean, and further improving deployment frequency in the midst of a cloud migration would be impossible without more people.

Mike had reached an impasse. Either hire more developers or find a better way. Â

Harness: The Better Solution

After evaluating Harness, Mike knew he had found his better solution. Marketron quickly onboarded their services onto the Harness platform and began increasing their deployment frequency and velocity.

With Harnessâ help, Marketron increased production deployment frequency to monthly and decreased deployment time to 1- hour. Marketron now deploys to non-production environments 1200 times a month, thereby increasing the number of automated tests and improving the quality of every service deployment.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform^{Â®}

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/global-continuous-delivery>

Linedata Achieves Global Continuous Delivery with Harness

Linedata saved \$500k in developer toil. Find out how you can too.

About Linedata

Linedataâs 1,300 employees in 20 offices provide global humanized technology solutions and services for the asset management and credit industries. Headquartered in France, Linedata had revenues of EUR 179.0 million in 2017. Software delivery within Linedataâs Asset Management AMP platform covers 4 business teams with around 50 developers/QA engineers in 5 geolocations around the world.

Next-Gen Asset Mgmt Thru Microservices

Linedata is transforming into a modular, next-generation asset management platform (AMP) based on .NET core microservices running in the public cloud. As part of this transformation, an internal Continuous Delivery (CD) initiative was created to increase developer velocity, strengthen feedback loops between dev and QA, and reduce time-to-market.

Deployment Scripts Didnât Scale

In the past, deployments were performed by QA engineers manually, using Terraform, and custom bash scripts. A typical deployment would take up to 1 hour per environment. In addition, deployment verification took 3-4 QA engineers a further 2 hours (min) with manual handoffs/approvals around the world as release candidates were promoted across each environment.

This manual deployment process meant feedback loops between developers and QA engineers could take days, given the geolocation and time zone differences between teams.

Moving to a microservices architecture in the public cloud meant that Linedata's current deployment process wouldn't scale. Engineering and QA needed a new approach to Continuous Delivery, a self-service CD initiative was born.

Empowering Teams with Self-Service CD

"We wanted to empower our Engineering and Business teams with self-service Continuous Delivery," said Andrey Budzar, Director of DevOps at Linedata.

"Strengthening the feedback loop between Engineering and the Business would provide our customers with maximum value," said Andrey.

Linedata's vision was to trigger and automate deployment pipelines from their Continuous Integration (CI) tool TeamCity. Developers could use CI webhooks to initiate an automated pipeline that would promote, test and verify new builds across their dev, QA, staging and production environments.

Harness Continuous Delivery as-a-Service

Within a few weeks of implementing Harness, Andrey and his team had successfully rolled out a new self-service CD platform for development, QA, and the business. "We realized tangible benefit within weeks of implementation," said Andrey.

Today, Linedata has automated deployment pipelines across all their environments, with automated environment provisioning, and automated verification to test new builds and release candidates. Harness is now successfully integrated across Linedata's technology stack of:

- Docker for .NET core microservices & .NET 4.X
- Amazon Web Services – EC2, ECS, and Fargate
- Teamcity for CI
- Atlassian Jira for pipeline ticketing & approvals
- Resharper, xUnit, WhiteHat, Selenium and LoadRunner for automated testing

\$500,000 Savings in DevOps Costs

With Harness, Linedata eliminated the need to incur \$500,000 in DevOps costs by scaling Continuous Delivery manually thru scripts and a team of DevOps engineers. Engineering, Business, and Ops teams across 5 geolocations are now empowered with self-service Continuous Delivery, maximizing collaboration and providing exceptional business value.

Linedata has also seen deployment times drop by 80% with verification/testing times drop by as much as 88%.

The net result is faster feedback loops for dev and the business, a reduced time-to-market, and better product experience for Linedata's customers.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps

engineer efficiency.

The Modern Software Delivery Platform®

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/increase-deployment-velocity-36x>

How Zeotap Increased Deployment Velocity 36x

From Downtime Windows to Deployment Nirvana: Learn How Zeotap Increased Deployment Velocity 36x.

About

Zeotap is the next-generation Customer Data Platform. Its Customer Intelligence Platform (CIP) empowers brands to unify, enhance, and activate customer data in a cookieless future, all while putting consumer privacy and compliance front and centre. Recognized by Gartner as a âCool Vendor,â Zeotap works with over 80 of the worldâs top 100 brands, including P&G, Audi, and Heineken. It is also the founding member of ID+, a universal marketing ID initiative.

Modernizations Create More Modernizations

Aditya Chandra, VP of Infrastructure and Security, was in the middle of modernizing Zeotapâs Continuous Integration pipelines when he realized the projectâs scope needed to be drastically increased. The switch from Jenkins to CircleCI exposed flaws in the Continuous Delivery process.

Zeotap had amassed over 400 Jenkins pipelines for 40 applications. With no deployment standardization, applications had a change failure rate of roughly 15-25%. Zeotap managed that risk by bundling deployments together and scheduling 4 hour downtime windows. During the downtime window, a team of three developers would slowly roll changes out. Zeotap then used a 10 day post deployment feedback loop to identify any issues the deployment might have caused. Once an issue was identified, it could take Zeotap 4 hours to roll back to the last working version.

Jenkins didnât provide Aditya with any visibility into the companyâs deployment velocity or the deployment success rate, but Aditya estimated that the existing process only allowed Zeotap to deploy once a month. Even if Aditya was able to modernize CI, without an efficient CD solution, Zeotap wouldnât be able to scale software delivery.

Deployment Nirvana

Aditya turned to Harness for a better software delivery experience. He was able to onboard 40 applications in the first three months using Harness.

Now, Zeotap is able to deploy multiple times a day using a standardized pipeline and automated rollbacks. The companyâs deployment frequency has increased roughly 36x because developers are given the independence to deploy on their own schedules.

A variety of deployment strategies are available to developers, such as blue-green deployments. Deployment effort reduced from a team of developers for 4 hours to 1 developer for 30 minutes, and rollback time reduced from 4 hours to seconds. Subsequently, developers have more time to work on high-priority projects.

Harness also helped Zeotap reach GDPR, ISO27001, and CSA STAR compliance. Using Harnessâ advanced RBAC, audit, and reporting capabilities, Aditya is able to pull detailed historical deployment information instantaneously.

Harness has delivered Zeotap into a state of software deployment bliss. Care to join them? Start your free trial today.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform®

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/harness-spinnaker>

Datastax Selected Harness over Spinnaker for CI/CD

Learn how Datastax upgraded their CI/CD with Harness

About DataStax

DataStax is the company behind the massively scalable, highly available, cloud-native NoSQL data platform built on Apache Cassandra®. DataStax gives users and enterprises the freedom to run data in any cloud at global scale with zero downtime and zero lock-in.

Developer Self-Service Feature Branching

The Cloud Services and Infrastructure team led by Frank Moley was tasked with empowering DataStax Astra engineers with self-service feature branching deployment capabilities to accelerate velocity and time-to-market.

DataStax Astra is a database-as-a-service built on Apache Cassandra and designed from the ground up to run anywhere, on any cloud, in any data-center, and in every possible combination.

Challenges With Operating Spinnaker For CD

Over the past year, Frank's team had self-hosted and self-managed the open-source project Spinnaker prior to their feature branching initiative.

âIn the end, the overhead of managing Spinnaker became unworkable for the team,â said Frank. âWe needed a solution that would allow us to focus on implementing pipelines versus managing the platform that provided them.â

In addition, debugging and troubleshooting failed deployments could often take 4-5 hours when changes were made. It became difficult to understand and trust what versions of services were running on specific clusters and environments.

DataStax CD requirements changed from basic containerized Kubernetes deployments to dynamic feature branching deployments, where engineers could deploy to their own namespaces within Kubernetes clusters.

With Spinnaker, onboarding new Astra teams/services could take several hours in addition to implementing canary deployments and automatic rollbacks, which was estimated to be several weeks of work.

Plug And Play Continuous Delivery With Harness

Harness Continuous Delivery as-a-Service allowed the DataStax Astra team to focus on implementing pipelines vs. managing the CD platform that provided those pipelines.

GitOps and GitSync capabilities combined with pipeline templates meant deployments could be managed as code with repeatable feature branching processes and rapid onboarding for development teams.

Native integrations with HashiCorp Terraform meant dynamic infrastructure provisioning, along with seamless integration of AWS secrets manager into deployment pipelines, meant dependencies with Jenkins could be eliminated altogether.

Today, troubleshooting deployments with Harness now takes less than a minute with granular transparency of deployment execution.

Audit trails also mean teams can understand and trust what versions of services are running on any cluster or environment in seconds.

Franks team also hopes to implement Canary deployments and automatic rollback within days so they can perform progressive delivery across their environments.

Best Practice Continuous Delivery Across 25-30 Engineers

With Harness, Frank's team achieved the following results:

- DataStax Astra team has implemented best-practice Continuous Delivery across 25-30 engineers
- Reduced engineering TOIL (20-30% of engineer time) associated with managing old CD platform which more than paid for the cost of Harness
- Reduced onboarding time of new services from a couple of hours to a couple of minutes
- Reduced deployment troubleshooting by 98% from 1 hour to under 1 minute
- Reduced time of implementing canary deployments and automatic rollback from several weeks to days

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform®

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/empower-engineers-with-cd>

Advanced Empowers 700 Engineers With Harness

Advanced saw a 10x return on their investment in 2 months. Find out how.

About Advanced

Advanced is the UK's third largest provider of business software and services with a £254m turnover, 16,000 customers and 2,200 employees. Advanced provides enterprise and market-focused solutions that allow customers to reimagine what is possible, innovate in their sectors and improve the lives of millions of people in the UK.

AWS Migration à The Trigger For CD

Advanced, like many other organizations, are migrating their traditional on-premise applications to the public cloud à more specifically AWS. With over 100 different products/teams and ~700 software engineers, an internal CI/CD initiative was kicked off earlier this year to empower all development teams with a self-service Continuous Delivery platform to help them increase velocity and reduce time-to-market.

Ad-hoc Continuous Delivery Didn't Scale

In the past, a typical deployment pipeline at Advanced would take months to build using a combination of Jenkins, AWS CloudFormation, Octopus Deploy, Puppet, SSM and Bash scripts. Pipelines required several thousand lines of Puppet code to be maintained by DevOps engineers and teams, said Martin Reynolds, Development and DevOps manager at Advanced.

In addition, these pipelines could take anywhere from 3 to 30 hours to execute with many manual interventions. Deployment health checks and verifications were also manual, ranging from 2 engineers for 2 hours for minor releases to 4-5 engineers for 1 day for major releases.

It was difficult to make Continuous Delivery scale with ad-hoc pipelines said Martin. That's when we decided to kick off a new CI/CD initiative internally, with the goal of making Continuous Delivery self-service for all our developers.

It was at this point that Martins team evaluated the Harness platform.

CD as-a-Service With Harness

Today, Harness helps Advanced automate software delivery across 100+ product teams, 700+ software engineers, 6 data centers and several technology stacks/tools that include:

- AWS EC2, ECS, EKS and Lambda for public cloud
- Jenkins, GitHub, and JFrog Artifactory for Continuous Integration (CI)
- AppDynamics, CloudWatch, and Elastic/ELK Stack for monitoring

Jenkins builds it, Harness deploys it, said Martin, Harness makes our deployments easy.

Jenkins builds it, Harness deploys it. Harness makes our deployments easy.

Harness provides Advanced software engineers and DevOps teams with the following capabilities:

- Intuitive Self-Service CD Platform to accelerate onboarding across teams
- Fully automated Canary deployments to reduce the risk of rollouts
- Auto-verification of deployments using machine learning to baseline performance and quality data from AppDynamics, CloudWatch, and Elastic/ELK stack
- Auto-rollback of deployments that fail performance/quality baselines
- Templatization of deployment pipelines by DevOps
- Re-use of pipeline templates by development teams

Upgrading Jenkins With Harness

Not only is Harness automating the delivery of software inside Advanced, but it also helps automate the upgrade of tooling like Jenkins.

For example, the DevOps team at Advanced would spend 2-3 hours a week upgrading Jenkins with patches and upgrades. With Harness, this process is automated and takes 15-20 minutes, saving the DevOps engineers a few hours a week.

10X ROI in Just 2 Months

The first 10 application teams and 100 services were onboarded with Harness in just 2 months, many of which were enabled on day one.

Deployment pipelines now take DevOps engineers a few hours to create/templatize with virtually no maintenance versus several months with ongoing maintenance. Developers reuse these pipeline templates for their specific services and environments without having to re-create the wheel every time, thus reducing their effort and time-to-market.

We saw immediate results, said Martin. One of our education teams saw deployment time drop from 3-hours to 1-minute across 19 services; another team saw deployment time drop from 30 hours to under 30 minutes.

Overall, Harness has helped Advanced reduce their average deployment time by 88% from 2 days to 2 hours, with a 90% reduction in onboarding time for dev teams from 2 days to 90 minutes. We've conservatively saved 50-60% of total DevOps and engineering time spent on deployments and our previous CI/CD process, said Martin.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

[Sensormatic optimizes retail operations with Harness SEI](#)

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform®

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/standardized-ci-cd-pipelines>

LogMeIn Standardized CI/CD Pipelines Across Teams

Learn how LogMeIn onboards new microservices in just one hour.

About LogMeIn

LogMeIn, Inc. (NASDAQ: LOGM) simplifies how people connect with each other and the world around them to drive meaningful interactions, deepen relationships, and create better outcomes for individuals and businesses. One of the world's top 10 public SaaS companies, and a market leader in communication & conferencing, identity & access, and customer engagement; support & solutions, LogMeIn has millions of customers spanning virtually every country across the globe. LogMeIn is headquartered in Boston with additional locations across North America, Europe, Middle East, Asia, and Australia.

Standardizing CI/CD Pipelines

Like many organizations, LogMeIn has been in the process of standardizing its Continuous Delivery process across teams, with a view to consolidating its toolsets. An internal initiative was kicked off, and Kyle Flavin, Sr. Staff Engineer at LogMeIn was tasked with making this happen.

Legacy tool deployments were becoming too complex to manage in legacy & modern environments. Complexities with various home-grown deployment frameworks mainly support and maintenance. Automated deployments using the existing CI tool were scripted using pipeline global libraries.

Onboarding new microservices could take a week or more for teams.

Evaluating Spinnaker and Harness

LogMeIn evaluated commercial and open-source tools and selected Harness for ease of install, use, and adoption.

"When I got started with Harness it literally took 1-hour to onboard my first service," said Kyle Flavin, Staff Software Engineer at LogMeIn. "One of our dev teams ran with Harness on their own and was in production within a week," said Kyle.

Today, development teams at LogMeIn get self-service Continuous Delivery with standardized deployment pipelines for their microservices.

The Harness CD abstraction model (CDAM) means LogMeIn teams can install, use and onboard in 1-hour regardless of their service container orchestrator or cloud infrastructure.

In addition, teams leverage Harness Canary Deployments and Continuous Verification to automate QA testing, health checks, and deployment verification so that performance anomalies and error regressions can be pro-actively identified in any environment by the deployment pipeline, and if need, code can be automatically rolled back.

Harness also provides complete audit capabilities and deployment dashboards to give teams instant visibility and feedback on who deployed what, where.

Reducing Deployment Time, Toil, and Effort

Since migrating to Harness, LogMeIn has seen deployment time drop by 98% from 10-12 hours per deployment to around 30 minutes. Deployment effort (# of people/hours) also dropped from 120 hours per deployment to just 2 hours. Onboarding time went from 1 week

to just 1 hour representing a 98% reduction.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/increasing-deployment-velocity>

intelliflo Repurposed 2 FTEs + Increased Deployment Velocity 20x

Find out how intelliflo reduced their time spent on Jenkins maintenance.

About

intelliflo widens access to financial advice through leading technology which powers the financial advisory experience. They use open software architecture combined with unmatched industry experience to simplify a complex digital landscape to help their customers thrive and grow. Their solutions support over 30,000 financial professionals worldwide, representing over three million end-investors, with over \$1 trillion advised across their platforms.

Migrations Are Never Simple

As intelliflo expanded its customer base internationally, the company wanted to migrate to AWS for better multiregional customer support. I'm sure some of you reading this article just had a visceral reaction to the words "migrate to AWS." Cloud modernization projects are never cut and dry and are full of potential headaches. One headache that immediately presented itself to intelliflo was the need to modernize software delivery.

Using a combination of Ansible and Jenkins scripts, intelliflo had created a fragile deployment pipeline structure. The pipelines were able to facilitate 10 deployments a day. Not bad. But in order to keep the pipelines working, intelliflo needed 2 full-time engineers dedicated to Jenkins maintenance. That's roughly \$250k in engineering resources spent on maintenance every year.

intelliflo was initially hesitant to start a new deployment process from scratch during a cloud migration, so the company asked a team of developers to estimate the time needed to make their Jenkins pipelines work for AWS. The team reported back that modernizing Jenkins would take 5 months and a team of 6 developers.

That wasn't an option. intelliflo needed a better way to deploy software, and they needed it fast.

Harness is One Less Headache

Intelliflo quickly adopted and embraced Harness.

The services onboarded onto Harness immediately got access to out-of-the-box blue-green deployments, a simple GitOps experience, and an easy way to manage environment configurations. Harness became the single source of truth for the company.

intelliflo now deploys 200+ times a day, all without the previous maintenance overhead.

Does the thought of cloud migration give you PTSD? Check out the Harness website to learn how we can help ease the burden.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/aws-cloud-migration>

Beachbody Accelerates AWS Cloud Migration

Beachbody avoided the headache associated with cloud migrations by leveraging Harness continuous delivery.

About Beachbody

Beachbody is a leading provider of fitness, nutrition, and weight-loss programs with over 23 million customers and \$1 billion in gross sales. Beachbody is the creator of the nation's most popular fitness and weight-loss solutions including P90X[®] Series, INSANITY[®], FOCUS T25[®], 21 Day Fix[®], Body Beast[®], PiYo[®], and Hip Hop Abs[®].

Migrating to AWS, Kubernetes & Lambda

Beachbody over the past year has refactored many of their on-premises monolith applications to Kubernetes microservices and Lambda functions running in Amazon Web Services (AWS). In 2019, another 90 API services are scheduled for onboarding and migration to the public cloud.

Jenkins Pipelines à Complex and Costly to Manage

In the past, developers and DevOps engineers at Beachbody would onboard and deploy services using Jenkins Pipelines. Over time, this process became extremely difficult, with significant infrastructure costs and time spent upgrading and troubleshooting the Jenkins platform. "Our Jenkins Pipelines were manual, brittle, and costly to manage," said Bob Chen, Senior Director of Tech Ops, DevOps, and SRE at Beachbody. "Onboarding new cloud-native services with Jenkins could take our DevOps team 3 days," said Bob. Deployment velocity with Jenkins was daily in non-production and bi-weekly for production. Production deployments typically averaged 6-8 people for approximately 2 hours, with manual verification (health checks) taking 30-60 minutes using Dynatrace APM and Elastic

stack. Bob and his team also looked at open-source solutions such as Spinnaker, but estimated it would take months to set up, and would require 2-3 full-time engineers to manage long term.

Cloud-Native Continuous Delivery with Harness

Today, Harness helps Beachbody automate software delivery across ~100 cloud-native services and their entire DevOps toolchain of :

- GitHub, Jenkins, and TravisCI for Continuous Integration (CI)
- AWS, Terraform, Kubernetes, and Lambda for cloud infra, container orchestration, and serverless
- Dynatrace APM and Elastic stack for monitoring

Developers have self-service Continuous Delivery (CD) capabilities that abstract all cloud-native services, artifacts, tools, and technologies. Pipelines are templated, version controlled, and managed as code using Harness GitOps. HashiCorp Terraform scripts are also integrated into each pipeline stage, so environments are consistent and dynamically provisioned/decommissioned when they are required. Onboarding new Kubernetes and Lambda services now takes 2 hours instead of 3 days, said Bob. In addition, Harness integrated with Dynatrace APM and Elastic stack out-of-the-box to auto-verify deployments using unsupervised machine learning to detect anomalies/regressions in any time-series performance metrics or log events. With Harness, our QA and testing is now fully automated and takes less than 5 minutes, said Bob. As a result, Beachbody has been able to automate their deployment pipelines end-to-end so environments, testing, deployment, and verification can be done in minutes vs. hours or days.

Ensuring Security & Compliance

Empowering engineers with self-service deployment capabilities at Beachbody required a focus on security and compliance. Engineers are authenticated through Harness OKTA SSO integration and are authorized/restricted through a well-defined role-based access control (RBAC) system, which provides a separation of duties and permissions across services, environments, and teams. Harness also provides a full audit trail giving instant insight into who changed what, when services are impacted. This allows Bob and his team to pinpoint the cause of impact in a matter of seconds. In addition, Harness keeps a record of all service versions across all environments so teams can report what software artifacts are deployed to specific clusters.

Reducing Deployment Time and Effort by 90%

Beachbody saw the following business benefits and ROI from implementing Harness Continuous Delivery:

- **Accelerated AWS Cloud Migration** from 1-year to 3-months
- **Doubled production deployment velocity** in 3 months
- **Reduced QA testing time by 92%** from ~1 hour to less than 5 minutes
- **Reduced deployment effort by 90%** from 7 people for 90 minutes to 3 people for 20 mins
- **Reduced deployment verification time by 78%** from 45 minutes to 10 minutes
- **Reduced application onboarding time by 92%** from 3 days to 2 hours
- **Deferred DevOps costs of \$600,000** from not supporting OSS CD solution

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform®

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/entur-improves-deployment-frequency-rollback-time>

Entur Improves Deployment Frequency from Twice a Week to 14 Times a Day, Rollback Time from Two Hours to Five Minutes

As the number of development teams nearly doubled, Entur looked to Harness CD & Feature Flags to gain more control in production, increase deployment frequency.

- Increased deployment frequency from twice per week to 14 times per day
- Decreased change failure rate from 20% to 5%
- Reduced rollback time from two hours to five minutes

About

Entur's mission is to create a more sustainable path for public transport in Norway. For this to happen, they must create new and unified solutions for travelers across multiple train network operators. To meet this need, Entur provides a solution to gather, enhance, and share data for public transport to help simplify travel across Norway.

Challenge: Stabilizing Pre-Production Environments for Scale

As the number of development teams nearly doubled (12 teams became 20 teams), Entur recognized their growth as an inflection point - an opportunity to reimagine how they deliver software. Each dev team had a different methodology for how they deliver and operate deployments. Therefore, a new, more unified approach was required to accommodate their immediate and future growth if they desired the ability to accelerate and become more efficient.

Entur's current state included three major environments: development, staging, and production â either deploying to production manually or employing CircleCI for continuous integration (CI) and continuous delivery (CD). While they were happy with their CI process, they had reached the velocity limitations of their custom scripted CircleCI pipelines in terms of deployment, and they were drowning in the volume of maintenance required to keep them up and running. According to Tommy BÃ, DevOps Engineer at Entur, âThese CircleCI pipelines were configured using thousands of YAML lines, making them a nightmare to maintain and update.â

The team needed to reduce the amount of maintenance required to keep their DevOps toolchains functioning and standardize on a deployment method for Entur that would deliver the kind of velocity gains they would need as their business continued to scale. They were deploying software on a bi-weekly basis and were experiencing a 20% change failure rate. Failure observed in a production deployment required a two-hour manual roll-back. These metrics weren't even close to where they needed to be to enable growth for their business.

Additionally, Entur developers were spending far too much time maintaining DevOps tools. They were also concerned about deployment pipeline stability when they did need to use those tools to deploy and were operating with the worry of unintentionally causing a major outage â far from an ideal developer experience.

The pain developers were feeling trying to scale velocity in their current software development lifecycle (SDLC) was clear. BÃ explained that âwhenever you have to manually deploy, you have to configure every environment correctly. That wasn't feasible as we were scaling.â

Feature management was also a functionality that Entur desired to leverage in their SDLC, but were making use of an in-house developed system that wasn't meeting their needs. As new team members joined and new features were being developed, the process of controlling the flag state of newly developed features proved to be inefficient.

Christian Nyvoll, Systems Developer at Entur, remarked that âthe way we were doing feature toggling slowed down our deployment process. We wanted to be able to pick up a new task and start doing it without thinking about what other team members are doing.â

The desire to alleviate these challenges led Entur to distill down their requirements for a new SDLC solution that was able to automate and execute on:

- Rolling deployments
- Canary deployments
- Instant rollbacks on failures
- Infrastructure provisioning through the deployment process

BÃ shared that âwe didn't want a tool that we would have to manage ourselves, so that developers could focus on their domain and on our core business.â

Solution: Intuitive, Self-Service CD Tooling

Entur chose Harness to simplify a complex continuous delivery (CD) process, reduce the amount of DevOps tools maintenance, reduce complexity in feature management, and provide developers with self-service tooling. According to BÅ., âWe chose Harness for the compelling user interface and intuitive workflows. Itâs a modern, state-of-the-art solution for software delivery that is very suitable for cloud services and microservices architecture applications.â

Nyvoll added: âEveryone has been pleased with the usability of Harness and how easy it is to actually just click in the UI. No need to learn commands. You just click the service that you want to deploy for or roll back for and the version that you want to use.â

Result: Higher Deployment Velocity, More Control in Production

Harness CD gave Enturâs developers self-service deployment capabilities without worrying about breaking their DevOps toolchain in the process. âOur deployment velocity increased from twice per week to more than 14 times per day with Harness. The standardized and automated process that we now employ makes it easier to deliver smaller features more often and with lower risks. In fact, our change failure rate decreased from 20% to 5%,â said BÅ.. Even if a deployment failure occurred, Entur could roll-back by just clicking a button. In under 5 minutes, the failing deployment would be rolled back via automation instead of making use of the thousands of lines of YAML that comprised their previous rollback solution.

Using the role-based access control (RBAC) capability within Harness, Entur can segment permissions and privileges by team, thereby decreasing the risk of accidentally making changes to other peopleâs systems. Furthermore, templated pipelines have let the team increase control of the approval process as well as more automated deployments. BÅ, shared: âIt's easier to track who's responsible for deployments using Harness. We can trace changes, not only internally within the team, but also across teams. This has a lot of value, especially for the support team thatâs trying to identify the cause of an incident.â

As for resolving issues caused by feature toggles, Nyvoll and his team now leverage Harness Feature Flags (FF) to reduce blockers when testing features, to get features out the door quicker, and to gain better control when feature errors arise. With both Harness CD and Harness FF, the Entur team is now delivering more features, more frequently, as well as remediating production issues in minutes instead of hours.

Nyvoll observed, âNow that we have Harness Feature Flags established in our team, we donât need to worry about everyone else during deployment. This makes it much easier to deal with complexities stemming from the increasing number of applications and services. And then when something is going wrong in production or in a staging environment, we have the ability to turn it off.â

Entur is not only seeing improvements in deployment velocity and reductions in the time it takes to resolve issues, but also seeing an overall reduction in stress levels given the significant improvement in developer experience with Harness. Making the shift to Harness has equipped Enturâs development teams with the software delivery platform they require to serve the transportation needs of Norway today and in the future.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Campspot Reduces Outage Risk by 78%

Campspot turned to Harness for reliable deployments with less outage risk.

About

Campspot crafts reservation software for resort owners that delivers their customers the perfect camping experience. They are a team of motivated people who use their talents to solve tough problems and build tested software.

Founded in early 2015, Campspot is looking to grow with people who embody their culture of continuous learning, innovation, and respectful collaboration. They're made up of smart and innovative individuals who work together. Every individual adds value to their product with proven skills, ingenuity, and creativity, and they expand these qualities in a cycle of teaching and learning.

AWS Code Deploy != Applications With Multiple Ports

Campspot originally deployed all of its applications using AWS Code Deploy, but as the company scaled, Code Deploy didn't scale with it. Campspot started adding multiple ports for its applications and Code Deploy's blue-green deploys didn't have native support for this structure. The team's only option was to add layers of custom scripted workarounds to deploy updates. These custom scripts solved the problem in the short term but in the long-term introduced three issues: complexity, large troubleshooting efforts, and slow manual rollbacks.

The deployment process went from straightforward to a complex nightmare. New developers had to read through long wiki descriptions to learn how to deploy. *Click this button, then navigate over here and click this button, then go back and click this other button.* Something often went wrong during this process.

This complexity meant Campspot's application could only be deployed once a week in the middle of the night. Developers were tired of staying up late to deploy their software.

This complexity meant the DevOps team spent large amounts of time troubleshooting failed deployments. The team got three or four pings every week about failed deployments. Each ping resulted in an hour of lost work time, which led to frustration by both the developers and DevOps teams.

The most concerning issue with the custom scripted Code Deploy solution was the downtime risk. Rollbacks took roughly 45 minutes to resolve, and bugs often went unnoticed for several hours. Campspot experienced some sort of issue every two weeks. This culminated into back-to-back outages in the summer of 2021. These outages each lasted for 15 minutes and caused large disturbances to users.

This was the final straw. Campspot needed a better, *safer* way to deploy.

Consistency and Safety

Campspot turned to Harness for reliable deployments with less outage risk.

The complexity that caused so much developer confusion is all but a memento from the past. Teams now have the capability to deploy every day and don't have to worry about exclusively deploying at night. Developers don't have to read pages of wiki notes, they simply press a button with Harness.

The DevOps team freed up time to work on other mission-critical projects, and developers no longer have to wait for the DevOps team to troubleshoot all of their deployment issues.

However, the following example showcases Campspot's largest improvement:

It now takes Campspot 10 minutes to roll back â that's 35 minutes saved each time.

Using Harness, Campspot is able to deploy faster and more securely. We call that a win.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/democratizes-continuous-delivery-using-harness>

Lessonly Democratizes Continuous Delivery Using Harness

Find out how Lessonly distributed the deployment workload to its developers.

About

Lessonly is a powerfully simple training software that helps teams learn, practice, and do better work. They enable busy teams to get on the same page, stay ahead of change, and deliver amazing experiences to customers and prospects. Learn more at Lessonly.com.

Whatâs Comfortable Isnât Always Best

Lessonly hosted its applications in Heroku, and for a time, life was good. Heroku provided deployment tools that developers used to deploy their services. But as Lessonly grew, site reliability engineers Stephen Gregory and Tyler Henrichs encountered issues. Heroku lacked infrastructure flexibility and governance features, and Herokuâs pricing structure exponentially increased the cost of hosting as the number of applications grew.

Stephen and Tyler were spending monthsâ worth of work to maintain and troubleshoot Heroku issues. The duo decided it was time to find a better solution. A solution that would provide advanced RBAC capability, more audit trail tracking, and help foster a âself-serviceâ deployment culture with developers.

Trial and Error

In the face of increasing Heroku expenses, Lessonly decided to migrate to AWS. Stephen and Tyler needed to figure out how they were going to deploy the new AWS applications.

The team began experimenting with Jenkins and Spinnaker-based pipelines. These Frankenstein pipelines used Jenkins to build artifacts and Spinnaker to deploy artifacts to production. During their experimentation, Tyler and Stephen found It took a full week to onboard a single service. Tyler and Stephen also estimated it would take a full 6 months to build a deployment solution using these tools, and even after 6 months, they would still lack RBAC and audit trails. They both decided delivering a lackluster solution wasnât an option.

Harness Delivers

After it was clear the Jenkins-Spinnaker combo wouldnât cut it, Lessonly turned to Harness. Harness gave developers deployment ownership over their own services and came out of the box with advanced RBAC and audit capabilities. Developers at Lessonly were surprised how easy it was to pick up.

Instead of taking a full week to onboard a new service, now it could be done in minutes. Stephen and Tyler no longer had to worry about the maintenance time. With their newfound free time, Stephen and Tyler directed more resources into Lessonlyâs application as opposed to working on getting said application into production.

The best part of moving to Harness? Hardly any of the developers even noticed the shift.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform®

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/carvana-reduces-resource-creation-time>

Carvana Cuts Infrastructure Provisioning Time by 92%, Saves \$1.4 Million With Granular Cloud Cost Management

Harness helped Carvana decrease manual effort and increase scalability during a huge growth period. Then, as Carvana needed to cut costs, Harness helped Carvana build on those automation gains while providing deeper and more granular insights into its cloud infrastructure costs.Â

Harness Impact

- 92% decrease in infrastructure resource provisioning time
- \$1.4M annualized savings in cloud spend
- Granular cost insights for cloud infrastructureÂ
- 4 months to deploy automated provisioning

About

Founded in 2012, Carvana is on a mission to change the way people buy and sell used cars. By focusing on customers, technology, and innovation, Carvana offers an intuitive and convenient online car buying, selling, and financing experience. Its customers can search and browse more than 70,000 vehicles online or visit one of its more than 30 automated car vending machines. By cutting out dealership overhead, Carvana saves customers thousands of dollars per purchase and provides a better car buying experience.

Challenges

Carvana's online experience and logistical efficiency is driven by the company's proprietary technology created by its innovative development team. Supporting that development is the company's eight-person infrastructure team, which has been using a script-based process to provision new instances. When a request for new infrastructure came along, the team created a custom image and deployed each one manually: an engineer would run scripts to build the machine, run scripts to install software, and run scripts to join the domain. The process took 2 to 3 hours for each of the 5 images requested per week.Â

âWe manage three different cloud environments: AWS, GCP, and Azure,â explained Will Adams, Manager, Cloud Support at Carvana. âWe do a lot of build outs so we knew we needed to automate.âÂ

As Carvana's business grew in the years leading up to and during the pandemic, this manual provisioning process threatened to put the brakes on growth. Unfortunately, with up to 15 hours per week devoted to manual deployments, the team just didn't have the time to

focus on automation. This put the infrastructure team under added pressure. But then, as car shortages and rising interest rates caused a drop in car sales in 2022, Carvana had to move from speeding infrastructure deployments to quickly trimming excessive infrastructure costs.Â

Solution

First, Carvanaâs infrastructure team set out to automate their highly manual provisioning processes with a project they termed TURBO, The Universal Request to Build Orchestrator. The team was using Jira to accept incoming infrastructure requests and wanted to add Terraform to automate the provisioning. It initially considered using a third-party to develop a custom solution, but soon realized that would be a painful, slow process and an ongoing maintenance nightmare. Then they found Harness.Â

âHarness was open enough to orchestrate Carvanaâs custom build process,â Adams explained. âWe wouldnât have to wait for something to be built. Harness could orchestrate our capabilities right away and get our infrastructure automation up and running as quickly as possible.â

After a quick proof of concept, Carvana had an automated provisioning process in just 4 months. With Harness doing the orchestration, engineers could make a provisioning request in Jira and virtual machines would be automatically built using Terraform. It was just in time, too. As Carvanaâs business boomed in 2021âthe company went from 7,000 to 21,000 employeesâthe team had to provision thousands of new machines for those now working from home due to COVID restrictions.

âWe just have to approve a build and it happens,â added Richard Taylor, Senior Cloud Infrastructure Engineer at Carvana. âAutomation became crucial for us not to become paralyzed by sheer growth.â

As the post-pandemic economy shifted, however, vehicles were in short supply and rising interest rates further dampened the used car market. Carvana pivoted to now focusing on costs. For the infrastructure team, however, with instances spread across cloud environments and managing teams, the view was murky at best.Â

âWe needed to understand our cloud costs because we were getting a lot of questions,â said Adams. âHarness Cloud Cost Management helped a lot. The anomaly detection saved a lot of time for me by pinpointing what was happening and how our bill changed month to month. The product is pretty awesome.â

Cloud Cost Management also helps Carvana answer cost-related questions related to its different vendors, services, and environments by providing monthly or even daily insights upon request. Itâs also looking to expand the use of Intelligent Cloud AutoStopping to power down more systems when theyâre not in use and to expand its use deeper into Azure and GCP.

Results

Harness helped Carvana decrease manual effort and increase scalability during a huge growth period. Then, as Carvana needed to cut costs, Harness helped Carvana build on those automation gains while providing deeper and more granular insights into its cloud infrastructure costs.Â

Harness orchestration has helped Carvana cut 92% from its infrastructure provisioning time. What used to take typically 2 to 3 hours and as long as 4 hours can now be completed automatically in as little as 20 minutes. At 5 instances per week, Carvana can redirect approximately 700 annual hours of expensive infrastructure engineering resources.

Harness Cloud Cost Management has also helped Carvana cut a significant portion of their cloud spend, with still more to go.Â

âWe're currently on a downward trend for our GCP spend and, for the last three months, we've been able to cut the bill down as much as we can,â said Adams. âWeâve saved about \$120,000 per month, which is \$1.4 million per year.â

Carvana expects even more savings as it expands its use of Intelligent Cloud AutoStopping to power down unused instances and even incorporate it into its automated provisioning system. The teamâs goal is to automatically add AutoStopping when a new machine is built so it can optimize costs from day one.

âCloud Cost Management was something we didn't know we needed at the time,â said Adams. âNow, especially with other departments using it, we can't live without it.â

Harness has helped Carvana remove the roadblocks associated with infrastructure provisioning. When the company needs to spin up new instances, it can now do so quickly and with little manual effort. And, it has granular cloud cost insights so it can provide and maintain instances without breaking the bank.Â

âWe are all-in on Harness because the tool works very well for every cloud environment and we're able to identify everything, which is amazing,â concluded Adams. âHaving control of our billing is key with this product.â

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/accelerate-kubernetes-migration>

Casting Networks Accelerates Kubernetes Migration by Four Months

Without Harness, Casting Networks would be 4 months behind their Kubernetes migration schedule.

About

Casting Networks, LLC is a cutting-edge entertainment technology company that helps performers find great roles and industry professionals find great talent.

Migrating to Kubernetes

Jeremy Malara, Director of DevOps at Casting Networks hit roadblocks while migrating Casting Networks legacy monolith applications to Kubernetes microservices. The purpose of this initiative was to increase developer velocity, cut costs and increase application stability. However, as this initiative kicked off it was soon apparent that the current deployment pipelines in place weren't suited well for this migration.

On-boarding new Kubernetes microservices quickly became a bottleneck. Casting Network's Jenkins pipelines were made of hundreds of thousands of lines of bash script. This complexity meant it took an entire day to onboard a single microservice. said Jeremy.

Knowledge silos and tribal knowledge created another migration barrier. Development teams at Casting Networks lacked experience and skills in building microservice pipelines using scripts and Jenkins pipelines. Jeremy's team of three SREs were the only people capable of conducting this migration.

Jeremy understood that his approach with Jenkins Pipelines was brittle and not scalable, and thus, would hinder his team's ability to complete their Kubernetes migration. This prompted him to evaluate other Continuous Delivery tools he could use to standardize Casting Network's deployment process for Kubernetes and Microservices.

Standardizing and Scaling Microservice Pipelines

After a thorough evaluation of commercial and open-source tools, Jeremy soon realized that Harness Continuous Delivery as-a-Service could transform his Kubernetes migration.

Harness pipeline templates would enable standard and repeatable deployment steps, removing the need for developers or a dedicated team of SREs to script each pipeline. Each developer would have the ability to instantiate a pipeline template, and trigger deployments themselves using a self-service model. The simplicity of the Harness platform would help drive adoption and allow teams to easily onboard their microservices on their own creating fewer bottlenecks and increasing developer velocity.

Once Harness deployment pipeline templates were created, Harness Continuous Verification could be embedded into these pipelines to automate QA and canary deployments so code could be pushed straight to production with no human intervention. Casting Network's

engineers âwouldnât even think about deployments, they would just let Harness take care of the work.â

In addition, a centralized secret store within Harness allowed teams to manage and reference secrets across their deployment pipelines in a secure and scalable manner while ensuring compliance.

Jeremy was convinced he needed Harness to jump start his Kubernetes migration initiative.

Reducing Deployment and Onboarding time by 95%

Jeremyâs team did their first deployment using Harness on Day 1. The time it took to deploy an artifact was reduced by 95% from 12 minutes to 30 seconds. The time it took to onboard a single microservice was reduced 97% from 1 day to just 10 minutes. As a result of implementing Continuous Verification, change failure rate was reduced by 90% from 20% to just 2%.

Most importantly, âWithout Harness, Casting Networks wouldâve been 4 months behind on their migration.â Harness helped them achieve their business goal faster.

â

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/case-studies/automated-ci-cd-rollback>

Build.com Automated CI/CD Rollbacks to 30 Seconds

Build.com removed most of their verification effort. Find out how you can too.

Prospects and analysts will sometimes ask âYeah, I get Harness â but do you REALLY have customers that automatically deploy, verify and rollback in production?â Yes, actually, we do. Automation sounds scary, but then again so does picking up your mission-critical apps and moving them to a large bookshop in the cloud.

Before I go on, I just want to give a shout out to Ed, Tim and the team at Build.com who were early believers in Harness â and more importantly, a big believer in CI/CD and the use of machine learning to automate the verification and rollback of production deployments.

Here is the story of how theyâve managed to automate their deployment pipeline end to end in just a few weeks.

The Deployment Pipeline

The first screenshot below shows Build.com's actual deployment pipeline, composed of 7 different stages and workflows:

- Stage 1 & 2 â Dev and Test workflows (execute in parallel)
- Stages 3 thru 5 â Edu/Sandbox/Staging workflows (execute in parallel)
- Stage 6 â Manual Approval
- Stage 7 â Production workflow

You can see that the deployment pipeline looked solid right up until Stage 7 where the production workflow failed.

The Failed Production Workflow

Below we can see the failed production workflow in more detail.

The workflow starts off with a canary deployment where phase 1 upgrades 20% of the production environment. Next, Harness marks this new deployment in New Relic and then instantly connects to both New Relic and Sumo Logic to verify the performance and quality of the service/application.

During this verification process, the Harness unsupervised machine learning algorithms start to analyze, compare and flag anomalies/regressions from the thousands of log entries and time-series metrics that both tools capture from the application.

You can see from the screenshot above that application performance was verified successfully but the application quality verification failed. This is normally a sign that Harness observed something unique and unexpected â typically a new event, error, or exception that has been introduced to the service/app.

Continuous Verification with Machine Learning

Prior to Harness, 6 to 7 team leads would spend 60 minutes verifying every production deployment. Now, one engineer can do this job in a matter of minutes.

Below is a screenshot that confirms why the above application quality verification step failed. By analyzing the application log data in Sumo Logic, Harness's machine learning algorithms were able to detect four new quality regressions. Specifically, Harness detected 4 new exceptions that have never been observed before in any previous deployments.

The grey dots you see in the below chart represent âbaseline eventsâ or âclustersâ â these are events that Harness has learned over time and are classified as ânormalâ because they are observed frequently during deployments. The red dots represent unknown events or events that have an unexpected frequency. These are typically the things that bite you in the ass during a production deployment.

Within seconds of detecting these regressions, Harness performed a âSmart Rollback,â taking the service/application back to the last working version (artifact & run-time configuration). It's worth mentioning that a Smart Rollback can be either fully automated as part of the workflow or controlled via manual intervention (a human).

The Smart Rollback (in 32 seconds)

If we zoom in at the top of the above workflow, you'll see visual confirmation that a rollback actually occurred after Phase 1 of the canary deployment failed:

Perhaps the most pleasing aspect of this deployment failure was the time it took to automatically roll back to the previous working versionâ|.just 32 seconds. The last time I chatted with Ed, Tim, and the team back in October, they told me that on average it took them 32 minutes to manually rollback a production deployment due to the number of scripts, dependencies, and configuration. Build.com is second only to Home Depot with well over \$500m of eCommerce revenue so every minute of downtime and rollback counts.

So there you go â proof that today it's possible to automate your entire CI/CD process end to end. In Build.com's case, they use Jenkins for Continuous Integration and Harness for Continuous Delivery and Continuous Verification of New Relic and Sumo Logic data. â

Better still, no one was hurt or replaced with the use of machine learning in this movie.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

[Sensormatic optimizes retail operations with Harness SEI](#)

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/blog-categories/saas>

SaaS

Campspot Reduces Outage Risk by 78%

Campspot turned to Harness for reliable deployments with less outage risk.

Carvana Reduces Resource Creation Time by 92% With Harness

Learn how Harness helped Carvana decrease manual effort during a growth period. A process that used to take 4 hrs now takes 20-30 mins!

Single Digits Saves \$108k in Deployment Effort

Single Digits made their migration easy with Harness. See how you can too.

Burst SMS Removes Outage Risk for Their Customers

Harness delivers better DevOps practices to all our customers. Find out how Burst SMS used Harness to reduce outages.

How Zeotap Increased Deployment Velocity 36x

From Downtime Windows to Deployment Nirvana: Learn How Zeotap Increased Deployment Velocity 36x.

Masters Microservices CI/CD

Learn how GoSpotCheck creates true Continuous Delivery pipelines using Harness.

Achieves Global Continuous Delivery with Harness

Linedata saved \$500k in developer toil. Find out how you can too.

Reduced Kubernetes Cloud Costs by Millions in 5 Months

Learn how Relativity democratized cloud cost data across engineering and product teams with Harness Cloud Cost Management.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/blog-categories/continuous-error-tracking>

Continuous Error Tracking

The Modern Software Delivery Platform®

Need more info? Contact Sales

Source URL: <https://www.harness.io/blog-categories/efficiency>

Efficiency

Optimizing Facebookâs Proxygen Project Build Time with Harness CI: A Case Study

In this article, we build Facebook's Proxygen project on the Harness CI module. Additionally, we conduct a comprehensive analysis, comparing the build time obtained from Harness CI with that of Github Actions, which happens to be Proxygen's default CI tool.

The Top 4 Deployment Patterns to Drive Efficiency in CI/CD

In this blog, we'll dive into a brief overview of the top deployment patterns to maximize developer efficiency in production.

Priceline Accelerates Move to GCP and Canary Deployments Using Harness

Harness helped Priceline quickly and easily build deployment pipelines from scratch for its migration to GCP and GKE. Find out how.

Carvana Reduces Resource Creation Time by 92% With Harness

Learn how Harness helped Carvana decrease manual effort during a growth period. A process that used to take 4 hrs now takes 20-30 mins!

Discover Dollar Saves 70% On Their Cloud Bill

Discover Dollar made idle cost management easy with Harness. See how you can too.

Reduced Kubernetes Cloud Costs by Millions in 5 Months

Learn how Relativity democratized cloud cost data across engineering and product teams with Harness Cloud Cost Management.

The Modern Software Delivery Platform®

Need more info? Contact Sales

Source URL: <https://www.harness.io/blog-categories/security-testing-orchestration>

Security Testing Orchestration

Dynamic Application Security Testing Best Practices

This blog delves into the ins and outs of Dynamic Application Security Testing (DAST), the steps involved in implementing it, and the tools and techniques that can help you stay ahead of security threats.

Meeting Federal Zero Trust Requirements With Harness

In this blog, we'll explore the new Federal Zero Trust Strategy as part of the Executive Order on Improving the Nationâs Cybersecurity and how Harness can help support it.Â

The Modern Software Delivery Platform®

Need more info? Contact Sales

Cloud Cost Management

Architecting for Cost Savings at Tyler Technologies

Your cloud architecture plays a significant role in your ability to control cloud costs as your business grows. Tyler Technologies realized this, and took action to optimize the organization of the shared architecture their clients relied on in order to maximize Cloud AutoStopping savings.

Harness Cloud Asset Governance powered by AIDA®: The Path to a Well Managed Cloud

Harness Cloud Asset Governance helps customers find and eliminate cloud waste automatically, while also providing a policy engine to ensure cloud resources are in compliance with corporate standards. By leveraging policy-as-code, it automates resource optimization, security, and compliance tasks, freeing your engineers to focus on creating innovative products and services that drive your revenue.

What Puts Harness Among 10 Top Providers in Cloud Cost Management

Harness Cloud Cost Management has made its debut appearance in The Forrester Wave™: Cloud Cost Management And Optimization, Q3 2022, and we're thrilled to be recognized as one of the top vendors in the industry.

Attributing Costs Using Harness Cloud Cost Management

There are quite a few people/teams involved in cloud costs. From devs and budget owners to finance, everyone needs to be able to look at areas of wastage. Attributing costs and cloud cost optimization are paramount to success. Let's see how we can do that with Harness CCM!

Kubernetes Cost Management Strategies: Cost Savings

The second Kubernetes cost management strategy we'll go over is cost savings. This means downsizing, rightsizing, and autoscaling to help in effective cloud cost management.

Log4Shell Response

Harness' response to the recent log4shell exploit. Find out how we remediated the situation and made sure you're safe.

The Harness Tooltip Framework

See how we selected our tooltip framework, how we implemented it, how we benefit from tooltips, and see examples of nicely-formatted tooltips.

Discover Dollar Saves 70% On Their Cloud Bill

Discover Dollar made idle cost management easy with Harness. See how you can too.

Top 8 Cloud Cost Management Tools

Managing cloud costs manually isn't easy nor scalable. The good news is that tools exist to remove the headache. This blog post explores some of the most popular tools that make cloud cost management easier.

A Developer's Guide to Reduce Cloud Costs

Cloud computing is not all that we expect when it comes to cost. This blog post shares helpful, applicable tips for developers to lower the costs of applications living in the cloud.

Gain New Perspectives on Your Cloud Bill

Harness just released a way to build and save custom dashboards and reports on cloud spend. Check out this new finops and cloud cost management tool.

Reduced Kubernetes Cloud Costs by Millions in 5 Months

Learn how Relativity democratized cloud cost data across engineering and product teams with Harness Cloud Cost Management.

The Modern Software Delivery Platform®

Need more info? Contact Sales

Source URL: <https://www.harness.io/blog-categories/feature-flags>

Feature Flags

Release Management with Feature Flags

Follow these tips on how to enhance your release management using feature flags.

Using Feature Flags for Effective Incident Management

Any application is prone to an incident, no matter its resilience. Learn how to use Feature Flags for effective incident management.

The Journey to Using Feature Flags at Their Full Potential

Feature flags may seem straightforward but actually require some strategy to use at their full potential. Follow these 4 steps to get your team mastering feature flags.

Leveraging Feature Flags for Zero-Downtime Database Migrations

Learn how to avoid downtime during your database migration with Harness Feature Flags.

How We Use Our Own Feature Flags at Harness

Let's take a look at how Feature Flags are used by our very own teams here at Harness with a case study on our Customer Success Management team.

Safe Testing in Production with Harness Feature Flags

Testing in production may seem like a risky task, but in reality, it can be the safest way to ensure a working environment. Learn when and how to test in production using Harness Feature Flags.

Elevating Reliability in Software: Harnessing Smart Feature Flags for Velocity and Performance

Software reliability is crucial for a brand's reputation and overall business success in today's digital age. Every software glitch, downtime, or inconsistent user experience could translate to lost customers, negative reviews, and reduced revenue.

Introducing Flag Archiving and Restoration for Smart Feature Flags

You now can archive and restore smart feature flags giving developers more flexibility to manage and address issues with the deletion of a flag.

A/B Testing For Feature Flags, What It Is And What It Shouldn't Be

Feature Flags and A/B testing often go hand in hand - but they probably shouldn't!

How Do I Start Using Feature Flags?

Let's dive into how your team can start implementing feature flags into your software delivery lifecycle.

Erlang and Elixir SDK Now Available in Harness Feature Flags

We are proud to announce the GA release of our Erlang and Elixir software development kit (SDK).

How Embedding Feature Flags in Engineering Culture Prevents Bad Habits

In this blog, we'll discuss the importance of incorporating feature flags as a cultural shift in your development teams and some common misconceptions about their implementation.

The Modern Software Delivery Platform®

Need more info? Contact Sales

Source URL: <https://www.harness.io/products/cloud-cost-management/savings-calculator>

Savings Calculator

Calculate how much you can potentially save on your cloud costs

Select Cloud Provider :

Enter monthly EC2 spend

Enter monthly ASG spend

Enter monthly EKS spend

Enter monthly RDS spend

Enter monthly ECS spend

What % of that total spend is covered by reserved instances or savings plans?

30%

What % of that total is production spend?

60%

Additional Coverage from Commitment Orchestrator

10%

Savings percent from Commitment Orchestrator

30%

Monthly Compute Engine VM Spend?

Monthly GKE Spend?

What percent of all that spend is covered by RIs / saving plans ?

30%

What percent of all that spend is production ?

60%

Additional Coverage from Commitment Orchestrator

10%

Savings percent from Commitment Orchestrator

30%

Monthly Azure VM Spend?

Monthly AKS Spend?

What percent of all that spend is covered by RIs / saving plans ?

30%

What percent of all that spend is production ?

60%

Additional Coverage from Commitment Orchestrator

10%

Savings percent from Commitment Orchestrator

30%

Potential Monthly Saving

\$ 39,000

AutoStopping

\$0

Commitment Orchestrator

\$10,000

Additional savings can be realized by right sizing over-provisioned resources and cleaning up unused resources using our Cloud Asset Governance feature and Recommendations for any cloud resource including Kubernetes workloads and nodes

Assumptions

The savings calculator assumes that you could gain 10% additional coverage through our automated purchasing and management of Reserved Instances (RIs) and savings plans using the Commitment Orchestrator feature, to a maximum of 80%. It also accounts for 30% savings from the additional coverage using 1-year convertible RIs and compute savings plans. Please note that the actual additional coverage, maximum and savings will depend on your historical compute spend patterns. Additionally, the calculator estimates a 70% reduction in non-production compute spend, based on the average savings our customers have achieved with our Cloud AutoStopping feature.

DISCLAIMER

* The cloud costs savings calculations shown above are for informational purposes only and based on some assumptions. Actual savings may vary due to a number of factors, including but not limited to usage, applicable discounts, promotions, contract terms, market conditions, the complexity of your resources, and unforeseen developments with cloud providers or vendors. Cloud cost pricing across all three cloud vendors is based on publicly available data.

Get started with smart Cloud Cost Management

Source URL: <https://www.harness.io/open-source>

Harness âœï, Open Source

As supporters of the open source community, we believe it is important to be clear when our software is free and open source, and when it isn't. Below is a breakdown of the licensing of our free and open source software (FOSS) and our free and source-available software.

Free and Open

Gitness is free and open-source software (FOSS) licensed under the Apache 2.0 license.

Community Edition

The Harness CI Community Edition (aka Drone), is free and open-source software (FOSS) licensed under the Apache 2.0 license.

Team Edition

The Harness CI Team Edition (powered by Drone) is made available under a source-available license called the Polyform Small Business License, which makes it free of charge for most users.

â

â¢ Free for individuals, students, startups, nonprofits and small businesses (under \$1 million US dollars).

â

â¢ Organizations who don't qualify for free may use the limitedÂ free trial.

Open Source Version

LitmusChaos is free and open-source software (FOSS) licensed under the Apache 2.0 license. It is currently at Incubation Level at CNCF (Cloud Native Computing Foundation).

Our Community

Join our virtual and in-person events; engage with our teams in our Community Forum and Slack workspace; follow our source code repos; and collaborate in the open with our engineers and product teams around new ideas and features.

Attend a Virtual Meetup

Follow us onÂ Github

Visit our Community

Join us on Slack

Benefits Of Open Source And Source-Available Code

For Engineers

You now have frictionless access to a globally-adopted and category-leading CI/CD platform, one with AI-powered intelligence that automates deployments and rollbacks. Get back your nights and weekends for more important things than fighting toil. And the best part, you can collaborate with a global community of users just like you.

Open Licenses

The open and accessible licenses give you the right to use, modify and distribute the code, with straightforward limits on using our code to compete with us. We chose this approach to empower our users with elements that they love from licenses like Apache that give them free and open access to the code, while assuring Harness is able to continue to provide industry-leading enterprise support to our customers.

For Enterprise

As an enterprise user, you can collaborate with our engineering and product teams around new designs. You can submit pull requests while simultaneously leveraging open source-code access.

Organizations We Sponsor

We're a Supporting Member of the Open Source Initiative.

We're a General Member of the Continuous Delivery Foundation.

We're a Silver Member of the CNCF.

Source URL: <https://www.harness.io/blog-categories/health>

Health

The Modern Software Delivery PlatformÂ®

Need more info? Contact Sales

Source URL: <https://www.harness.io/blog-categories/finance>

Finance

Unlock the Secrets of Cloud Cost Forecasting: Dimensions Finance Canât Ignore

Tips for what finance needs to include in a cloud cost forecast. It's time to stop feeling like your forecast is at the mercy of out of control cloud costs.

Carvana Reduces Resource Creation Time by 92% With Harness

Learn how Harness helped Carvana decrease manual effort during a growth period. A process that used to take 4 hrs now takes 20-30 mins!

Discover Dollar Saves 70% On Their Cloud Bill

Discover Dollar made idle cost management easy with Harness. See how you can too.

Raisin Runs from Jenkins Nightmare - Saves 525k per Year

Raisin removed their CD maintenance migraine, find out how you can too!

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/blog-categories/it>

IT

Metrikus Shortens Commit-to-Production Time by 66%

Metrikus made feature flag management easy with Harness. See how you can too.

Single Digits Saves \$108k in Deployment Effort

Single Digits made their migration easy with Harness. See how you can too.

Burst SMS Removes Outage Risk for Their Customers

Harness delivers better DevOps practices to all our customers. Find out how Burst SMS used Harness to reduce outages.

How Zeotap Increased Deployment Velocity 36x

From Downtime Windows to Deployment Nirvana: Learn How Zeotap Increased Deployment Velocity 36x.

intelliflo Repurposed 2 FTEs + Increased Deployment Velocity 20x

Find out how intelliflo reduced their time spent on Jenkins maintenance.

Ancestry Cuts Downtime by 50% Using Harness Governance

Find out how Ancestry secured their platform using Harness.

Empowers Developers with Harness

Vuclip found that Harness was the only solution that fully integrated with their GCP, Kubernetes, APM, and Log stack.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: https://www.harness.io/company/contact-sales?utm_source=Website&utm_medium=internal&utm_content=Contact-Sales-CD-Pricing

The Software Delivery Platform Business Service Teams Prefer

Want to see a custom demo or get help finding the right plan? We'd love to chat. Lean into AI/ML withÂ Harness - the simple, scalable CI/CD platform.

- Fully Integrated Modules
 - Test Intelligence
 - Enterprise-Grade Security
 - Fine-Grained RBACÂ Security
 - Complete Flexibility
-
- AI/ML DrivenÂ Workflows
 - Smart Feature Rollouts
 - Multi-Cloud Cost Management
 - Comprehensive Auditability

Contact Sales

By signing up, you agree to our Privacy Policy and our Terms of Use.

Nick Willson, TechOps Lead, GoSpotCheck

Take Harness for a spin

Source URL: <https://www.harness.io/company/jobs>

Thinking about pursing a career opportunity at Harness? Join us in taking software delivery into the future. We're always on the lookout for exceptional talent to join our fast growing global team!

Life at Harness

We are a company that strives to be the best in the industry, while remaining an enjoyable place to work. We are fortunate to have the opportunity to come together virtually and in-person across the globe. Life at Harness embraces all our employees with global holiday and milestone celebrations, TGIF off program, and morale boosting activities all while Getting Ship Done!

Source URL: https://www.harness.io/company/contact-sales?utm_source=Website&utm_medium=internal&utm_content=Contact-Sales-STO-Pricing

The Software Delivery Platform Business Service Teams Prefer

Want to see a custom demo or get help finding the right plan? We'd love to chat. Lean into AI/ML withÂ Harness - the simple, scalable CI/CD platform.

- Fully Integrated Modules
 - Test Intelligence
 - Enterprise-Grade Security
 - Fine-Grained RBACÂ Security
 - Complete Flexibility
-
- AI/ML DrivenÂ Workflows
 - Smart Feature Rollouts
 - Multi-Cloud Cost Management
 - Comprehensive Auditability

Contact Sales

By signing up, you agree to our Privacy Policy and our Terms of Use.

Nick Willson, TechOps Lead, GoSpotCheck

Take Harness for a spin

Source URL: https://www.harness.io/company/contact-sales?utm_source=Website&utm_medium=internal&utm_content=Contact-Sales-CI-Pricing&utm_term=Self-Managed

The Software Delivery Platform Business Service Teams Prefer

Want to see a custom demo or get help finding the right plan? We'd love to chat. Lean into AI/ML withÂ Harness - the simple, scalable CI/CD platform.

- Fully Integrated Modules
 - Test Intelligence
 - Enterprise-Grade Security
 - Fine-Grained RBACÂ Security
 - Complete Flexibility
-
- AI/ML DrivenÂ Workflows
 - Smart Feature Rollouts
 - Multi-Cloud Cost Management
 - Comprehensive Auditability

Contact Sales

By signing up, you agree to our Privacy Policy and our Terms of Use.

Nick Willson, TechOps Lead, GoSpotCheck

Take Harness for a spin

Source URL: https://www.harness.io/company/contact-sales?utm_source=Website&utm_medium=internal&utm_content=Contact-Sales-Cl-Pricing&utm_term=SaaS

The Software Delivery Platform Business Service Teams Prefer

Want to see a custom demo or get help finding the right plan? We'd love to chat. Lean into AI/ML withÂ Harness - the simple, scalable CI/CD platform.

- Fully Integrated Modules
 - Test Intelligence
 - Enterprise-Grade Security
 - Fine-Grained RBACÂ Security
 - Complete Flexibility
-
- AI/ML DrivenÂ Workflows
 - Smart Feature Rollouts
 - Multi-Cloud Cost Management
 - Comprehensive Auditability

Contact Sales

By signing up, you agree to our Privacy Policy and our Terms of Use.

Nick Willson, TechOps Lead, GoSpotCheck

Take Harness for a spin

Source URL: https://www.harness.io/company/contact-sales?utm_source=Website&utm_medium=internal&utm_content=Contact-Sales-Cl-Pricing

The Software Delivery Platform Business Service Teams Prefer

Want to see a custom demo or get help finding the right plan? We'd love to chat. Lean into AI/ML withÂ Harness - the simple, scalable CI/CD platform.

- Fully Integrated Modules
- Test Intelligence
- Enterprise-Grade Security
- Fine-Grained RBACÂ Security
- Complete Flexibility

- AI/ML Driven Workflows
- Smart Feature Rollouts
- Multi-Cloud Cost Management
- Comprehensive Auditability

Contact Sales

By signing up, you agree to our Privacy Policy and our Terms of Use.

Nick Willson, TechOps Lead, GoSpotCheck

Take Harness for a spin

Source URL: <https://www.harness.io/products/cloud-cost-management/features>

Get Control of FinOps with Automated Cloud Cost Management

Cost Reporting

Cost Perspectives

Visualize complex multi-cloud and Kubernetes cluster cost information for cost showback and chargeback, organized in the business context (teams, workloads, applications, etc.) that your teams need.

Cost Categories

Capture all costs for a team/application/workload into well defined cost buckets to give your teams accurate and detailed cloud costs, grouped in ways most meaningful to you.

Anomaly Detection

Our intelligent cloud cost anomaly detection automatically detects and alerts your affected teams of abnormal cost spikes as they happen, stopping bill shock in its tracks.

Custom BI Dashboards

Dig deeper into your cloud costs, monitor KPIs, and understand your cloud usage landscape with our powerful and flexible business intelligence and analytics platform.

Cost Optimization

Cloud AutoStopping™

Save up to 70% on non-production cloud costs when you automatically detect, shutdown and auto-start idle cloud resources, all with a simple one-time setup that doesn't need maintaining.

Recommendations

Optimize your cloud resources with relevant and actionable recommendations across AWS/Azure/GCP for a host of resources like K8s clusters, VMs, ECS, EBS and RDS.

Commitment Orchestrator

Never miss an RI/SP contract renewal again. Automate long-term commitment management for cost savings and optimized compute coverage that always meets coverage targets.

Cost Governance

Cloud Asset Governance

Real-time governance policy enforcement and auto-remediation with Governance-as-Code. AI-powered YAML policy generation using Harness AIDA for effective cost, security, and compliance management across cloud assets.

Hierarchical Budgets

Set and track daily, monthly, quarterly, and annual budgets, with real-time forecasting, alerts, and cascading budgets across your organization's hierarchy.

Get started with smart Cloud Cost Management

Source URL: https://www.harness.io/company/contact-sales?utm_source=Website&utm_medium=internal&utm_content=Contact-Sales-FF-Pricing

The Software Delivery Platform Business Service Teams Prefer

Want to see a custom demo or get help finding the right plan? We'd love to chat. Lean into AI/ML with Harness - the simple, scalable CI/CD platform.

- Fully Integrated Modules
- Test Intelligence
- Enterprise-Grade Security
- Fine-Grained RBACÂ Security
- Complete Flexibility
- AI/ML DrivenÂ Workflows
- Smart Feature Rollouts
- Multi-Cloud Cost Management
- Comprehensive Auditability

Contact Sales

By signing up, you agree to our Privacy Policy and our Terms of Use.

Nick Willson, TechOps Lead, GoSpotCheck

Take Harness for a spin

Source URL: https://www.harness.io/company/contact-sales?utm_source=Website&utm_medium=internal&utm_content=Contact-Sales-CCM-Pricing

The Software Delivery Platform Business Service Teams Prefer

Want to see a custom demo or get help finding the right plan? We'd love to chat. Lean into AI/ML with Harness - the simple, scalable CI/CD platform.

- Fully Integrated Modules
- Test Intelligence
- Enterprise-Grade Security
- Fine-Grained RBACÂ Security
- Complete Flexibility
- AI/ML DrivenÂ Workflows
- Smart Feature Rollouts
- Multi-Cloud Cost Management
- Comprehensive Auditability

Contact Sales

By signing up, you agree to our Privacy Policy and our Terms of Use.

Nick Willson, TechOps Lead, GoSpotCheck

Take Harness for a spin

Source URL: https://www.harness.io/company/contact-sales?utm_source=Website&utm_medium=internal&utm_content=Contact-Sales-FF-Pricing

The Software Delivery Platform Business Service Teams Prefer

Want to see a custom demo or get help finding the right plan? We'd love to chat. Lean into AI/ML with Harness - the simple, scalable CI/CD platform.

- Fully Integrated Modules
- Test Intelligence
- Enterprise-Grade Security
- Fine-Grained RBAC Security
- Complete Flexibility
- AI/ML Driven Workflows
- Smart Feature Rollouts
- Multi-Cloud Cost Management
- Comprehensive Auditability

Contact Sales

By signing up, you agree to our Privacy Policy and our Terms of Use.

Nick Willson, TechOps Lead, GoSpotCheck

Take Harness for a spin

Source URL: <https://www.harness.io/blog-categories/velocity>

Velocity

Campspot Reduces Outage Risk by 78%

Campspot turned to Harness for reliable deployments with less outage risk.

Priceline Accelerates Move to GCP and Canary Deployments Using Harness

Harness helped Priceline quickly and easily build deployment pipelines from scratch for its migration to GCP and GKE. Find out how.

Carvana Reduces Resource Creation Time by 92% With Harness

Learn how Harness helped Carvana decrease manual effort during a growth period. A process that used to take 4 hrs now takes 20-30 mins!

Choice Hotels Deploys 85% Faster Using Harness

Harness helps us capitalize on the investments we've made in automation tools by streamlining everything into a single pipeline.

SMU Reduces Rollback & Deployment Time by Over 80%

SMU becomes a deployment powerhouse by leveraging Harness.

Metrikus Shortens Commit-to-Production Time by 66%

Metrikus made feature flag management easy with Harness. See how you can too.

Single Digits Saves \$108k in Deployment Effort

Single Digits made their migration easy with Harness. See how you can too.

Burst SMS Removes Outage Risk for Their Customers

Harness delivers better DevOps practices to all our customers. Find out how Burst SMS used Harness to reduce outages.

How Zeotap Increased Deployment Velocity 36x

From Downtime Windows to Deployment Nirvana: Learn How Zeotap Increased Deployment Velocity 36x.

intelliflo Repurposed 2 FTEs + Increased Deployment Velocity 20x

Find out how intelliflo reduced their time spent on Jenkins maintenance.

Ancestry Cuts Downtime by 50% Using Harness Governance

Find out how Ancestry secured their platform using Harness.

Raisin Runs from Jenkins Nightmare - Saves 525k per Year

Raisin removed their CD maintenance migraine, find out how you can too!

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/blog-categories/service>

Service

Campspot Reduces Outage Risk by 78%

Campspot turned to Harness for reliable deployments with less outage risk.

Priceline Accelerates Move to GCP and Canary Deployments Using Harness

Harness helped Priceline quickly and easily build deployment pipelines from scratch for its migration to GCP and GKE. Find out how.

Choice Hotels Deploys 85% Faster Using Harness

Harness helps us capitalize on the investments we've made in automation tools by streamlining everything into a single pipeline.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/products/service-reliability-management/automated-slo-tracking>

Automated SLO Tracking

SLOs, SLIs and Error Budgets

Enable your teams to create and track SLOs and Error Budgets with automation using SLOs as Code, templates, and APIs. Stay ahead of critical issues that can impact your business by continuously monitoring the health of your service and its dependencies in real time.

Advanced SLO configuration

Track your entire user journey and get a holistic view of the overall reliability performance with Composite SLOs. Give your teams more control over how they manage error budgets. Use SLO Downtime to minimize error budget impact and meet compliance requirements.

SLO reporting

Eliminate hours of manual work with automated SLO and Error Budget Reports. Use historical SLO performance data to confidently set SLAs that are aligned with your business goals. Get actionable insights from SLO Reports to help you make better strategic decisions

and improve team accountability.

Try Service Reliability Management today

Deploy faster with better reliability

Source URL: https://www.harness.io/company/contact-sales?utm_source=Website&utm_medium=internal&utm_content=Contact-Sales-SRM-Pricing

The Software Delivery Platform Business Service Teams Prefer

Want to see a custom demo or get help finding the right plan? We'd love to chat. Lean into AI/ML with Harness - the simple, scalable CI/CD platform.

- Fully Integrated Modules
- Test Intelligence
- Enterprise-Grade Security
- Fine-Grained RBAC Security
- Complete Flexibility
- AI/ML Driven Workflows
- Smart Feature Rollouts
- Multi-Cloud Cost Management
- Comprehensive Auditability

Contact Sales

By signing up, you agree to our Privacy Policy and our Terms of Use.

Nick Willson, TechOps Lead, GoSpotCheck

Take Harness for a spin

Source URL: <https://www.harness.io/products/service-reliability-management/ci-cd-integrations>

CI/CD Integration

Automated pipeline governance

Use Policy as Code to create and enforce reliability guardrails in your pipelines. Provide flexibility to teams to reset error budget with complete audit trail and accountability.

Deployment impact analysis

Easily configure your pipelines to monitor the health of your service and its dependencies after each deployment. Get immediately notified on service health and SLO violations. View detailed report summarizing the SLO performance and changes for the analysis period.

SLO as code

Use code repositories as the single source of truth for your SLO definitions. Automate your SLO workflows with the Harness Terraform Provider, so you can easily create, manage, and maintain your SLOs as code. Make SLOs a team sport with SLOs as Code, so everyone can see how their work is impacting SLO experiments.

Try Service Reliability Management today

Deploy faster with better reliability

Source URL: <https://www.harness.io/blog-categories/continuous-integration>

Continuous Integration

Push Helm Charts to Container Registries with Harness CI

Learn how to publish OCI Helm charts to a Container registry

Run Ruby Builds Up To 4x Faster Using Harness CI Test Intelligence

Announcing Ruby Test Intelligence preview for Harness CI! Run only the subset of tests relevant to your Ruby code changes and skip the rest.

Dreading the upcoming Bitbucket Server support deadline? Try Gitness.

Learn how Gitness can replace Bitbucket Server before Atlassian ends support on February 2nd, 2024

Unit Testing vs. Integration Testing

Explore the key differences between integration tests and unit tests in the realm of software development. Learn how these testing methods contribute to building robust and reliable software systems. Discover their unique roles, benefits, and how they integrate into the CI/CD pipeline for delivering top-quality software

Optimizing Facebook's Proxygen Project Build Time with Harness CI: A Case Study

In this article, we build Facebook's Proxygen project on the Harness CI module. Additionally, we conduct a comprehensive analysis, comparing the build time obtained from Harness CI with that of Github Actions, which happens to be Proxygen's default CI tool.

Easily Migrate Continuous Integration Pipelines to Harness

Explore Harness CI Migration Utility, our latest innovation for effortless CI migration

Thousands of community plugins at your fingertips with Harness CI

Tap into the collective wisdom of the developer community by running Drone plugins, Bitrise steps, and GitHub Actions from your Harness CI pipeline

Introducing Remote Debugger: A Powerful Tool for Real-Time Builds Debugging

Learn how the Remote Debugger simplifies troubleshooting, transforming this time-consuming exercise in guesswork into a targeted, real-time effective investigation.

Accelerate iOS and MacOS Build Pipelines with Apple M1 and Harness

Apple M1 support is available for CI pipelines! Learn how to optimize your Apple platforms app development by building on Apple Silicon M1 machines with Harness Cloud.

The Top 4 Deployment Patterns to Drive Efficiency in CI/CD

In this blog, we'll dive into a brief overview of the top deployment patterns to maximize developer efficiency in production.

Setting Up PagerDuty Notifications for Incident Management Using Harness

Learn how to integrate PagerDuty incident management tool with Harness to get a clear view of your CI/CD pipeline events.

Using Ephemeral Environments to Test and Scale Your Deployments

In this tutorial, we will discuss what ephemeral environments are, how they work, and how to configure them.

The Modern Software Delivery Platform®

Need more info? Contact Sales

Software Engineering Insights

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

Discover how Harness Software Engineering Insights (SEI) revolutionizes code quality and productivity. Explore the Trellis Framework, quality metrics, and structured engineering improvements for unparalleled software development. Learn more in our comprehensive guide.

Hygiene in SDLC: A Key to Engineering Efficiency

Explore the importance of hygiene in software development with our blog. Discover how it ensures top-notch software quality and insights. Get practical tips for improvement and join us in fostering a culture of continuous excellence. Prioritize quality and efficiency for success in the software world!

How the size of a Pull Request can impact developer experience?

Learn how the size of a Pull Request can significantly impact developer experience, and discover how Harness SEI helps in optimizing PRs to improve code quality, reduce review times, and enhance collaboration for your engineering team.

Unlocking Efficiency: Exploring Churn Rate with Harness Software Engineering Insights (SEI)

Explore the dynamic world of Churn Rate and its role in revolutionizing software delivery with Harness Software Engineering Insights (SEI). Uncover actionable insights, optimize workflows, and navigate the seas of development challenges with precision.

What are Engineering Metrics?

In this article, we will explore what engineering metrics are and how they can help organizations deliver better results, enhance productivity, and improve processes.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: https://www.harness.io/company/contact-sales/?utm_source=website&utm_content=Contact-Sales-CD-Pricing

The Software Delivery Platform Business Service Teams Prefer

Want to see a custom demo or get help finding the right plan? We'd love to chat. Lean into AI/ML with Harness - the simple, scalable CI/CD platform.

- Fully Integrated Modules
- Test Intelligence
- Enterprise-Grade Security
- Fine-Grained RBAC[®] Security
- Complete Flexibility
- AI/ML Driven[®] Workflows
- Smart Feature Rollouts
- Multi-Cloud Cost Management
- Comprehensive Auditability

Contact Sales

By signing up, you agree to our Privacy Policy and our Terms of Use.

Nick Willson, TechOps Lead, GoSpotCheck

Take Harness for a spin

Continuous Delivery

Harness Acquires Armory Assets

We're excited to announce that Harness has acquired key intellectual property and technology from Armory.

What is serverless?

Understand how Serverless works and how it's reshaping software development, driving efficiency, and reducing costs

Verifying Your Kubernetes Deployments Made Easy

Harness (the modern continuous delivery platform) has introduced a feature called continuous verification to help DevOps professionals to verify their deployments through any monitoring tool of their choice.Â

How Harness Implemented CDN for Its UI Services

In this blog, we'll walk through what a CDN is and how Harness implemented it into our environment.

How To Implement CI/CD Efficiently

Let's explore how you can adopt CI/CD to improve your organization's efficiency and delivery pipeline in Harness.

Why GitOps is Necessary for Cloud-Native Enterprises

Let's talk about GitOps and its benefits then integrate Argo with a Kubernetes cluster to deliver changes.

Deploying a To-Do Application to Kubernetes with Harness

This tutorial shows how we can easily deploy a Kubernetes to-do application using Harness CI/CD.

How to Build and Deploy a Node.js Microservice with Harness

When it comes to modern software development, microservices are one of the hottest trends. This blog post will explain what microservices are, why you should use them, their benefits, and how to deploy them using Harness.

How to Create Multi-stage Docker Builds with Harness Continuous Delivery

This tutorial explains what multi-stage Docker builds are and how they can help speed up your development process.

CI/CD Challenges and How to Solve Them

This blog post will share four common CI/CD challenges and how to solve them.

10 Signs You Don't Do Continuous Delivery

While many of these development practices may feel commonplace, if you're doing these, you're not actually doing continuous delivery at all.

Now Generally Available: Harness GitOps-as-a-ServiceÂ Delivers Scalability, Control, and Security to Enterprises

As GitOps evolves into a new industry standard for continuous software delivery, Harness is proud to introduce generally available enterprise capabilities for GitOps deployments with governance, reliability, and visibility at scale as a part of the most comprehensive software delivery platform on the market.

The Modern Software Delivery PlatformÂ®

Need more info? Contact Sales

Governance

Harness Cloud Asset Governance powered by AIDAâ€¢: The Path to a Well Managed Cloud

Harness Cloud Asset Governance helps customers find and eliminate cloud waste automatically, while also providing a policy engine to ensure cloud resources are in compliance with corporate standards. By leveraging policy-as-code, it automates resource optimization, security, and compliance tasks, freeing your engineers to focus on creating innovative products and services that drive your revenue.

Harness Named a Market Fast Mover in GigaOm Radar Report for Cloud FinOps

Harness is proud to be included in the latest GigaOm Radar for Cloud FinOps as a Fast Moving Challenger in the Innovative/Platform Play quadrant of the report.

Meeting Federal Zero Trust Requirements With Harness

In this blog, we'll explore the new Federal Zero Trust Strategy as part of the Executive Order on Improving the Nation's Cybersecurity and how Harness can help support it.Â

Five Key Technologies Shaping the Future of Software Delivery

If you're ready to take your software delivery to the next level, consider looking into these five key concepts that are quickly becoming essential to modern software delivery.Â

Meeting Federal Compliance Requirements with Policy-as-Code and Automated Change Management

With a broad number of compliance policies, how do federal IT shops ultimately implement, govern, and enforce them?

Balancing Developer Freedom and Governance with OPAÂ

Besides good beer, the next biggest component in developer happiness is to have access to the tools and processes to get their job done quickly and effectively.

Feature Flags Policy Governance Tutorial

In this post, we go in depth to teach you all about the Feature Flagsâ policy governance: architecture, use cases, data available to utilize, and more.

Introducing Harness Policy as Code, Powered by OPA

It's here: Harness Policy as Code helps organizations create and enforce policies on deployments, infrastructure, and more, providing developer velocity without sacrificing compliance and standards.

Introducing Harness GitOps: Enterprise Security, Governance, and Scale for GitOps Deployments

Harness GitOps delivers scalability, control, and security - a new deployment strategy with less risk. Try it today!

Campspot Reduces Outage Risk by 78%

Campspot turned to Harness for reliable deployments with less outage risk.

Metrikus Shortens Commit-to-Production Time by 66%

Metrikus made feature flag management easy with Harness. See how you can too.

Federal Government: DevOps, DevSecOps, and Pipelines

To say that I have been around the block with respect to working in the Federal Government is an understatement. I have had the privilege of working with - and for - some of the most secretive projects, and quite honestly, the most brilliant people in the industry.

The Modern Software Delivery Platform®

Need more info? Contact Sales

Source URL: <https://www.harness.io/blog-categories/chaos-engineering>

Chaos Engineering

Harness Chaos Engineering Faults Landscape

Harness Chaos Engineering, powered by LitmusChaos, provides a wide range of chaos faults that can be used to verify resilience of your system systematically, incrementally and continuously. This article covers the anatomy of a Harness Chaos Engineering experiment and types of chaos faults that the product supports.

Harness Announces the 4th Edition of Chaos Engineering Conference focusing on Software Resilience

Save the date for January 24th-25th, 2024, as Harness brings back Chaos Carnival, the annual worldwide conference on Chaos Engineering.

Chaos Experiments in Harness CD Pipelines

Chaos Engineering delivers fantastic benefits to the developers, development teams, and overall DevOps when chaos experiments are automated into the regular CD pipelines. Read this blog to know the benefits and how to use chaos experiments in Harness CD pipelines.

Chaos Engineering with Jenkins

Jenkins pipelines can deliver resilience verification by adding chaos experiments into their pipelines. Adding chaos experiments into CD pipelines have significant benefits such as improving developer productivity, reducing the operational debt and catching the resilience issues at the earliest.

Achieving Continuous Resilience with Harness Chaos Engineering

Introducing the continuous resilience approach. Understand the modern approach to practicing chaos engineering where the efforts to build resilience are inserted into all stages of your software development life cycle through automated chaos experiments. Harness Chaos Engineering or Harness CE comes with all the building blocks required for achieving Continuous Resilience in DevOps.

Highlights from LitmusChaos at KubeCon + CloudNativeCon Europe 2023

Here's what we saw from LitmusChaos at the Cloud Native Computing Foundation's (CNCF's) flagship conference in Europe.

Disaster Recovery 101

In this blog, we'll discuss disaster recovery plans and best practices for creating and executing one.

What We Learned at Chaos Carnival 2023

We're thrilled to report that our third annual chaos engineering conference, Chaos Carnival 2023, was a success. Read the recap!

How Chaos Engineering Strengthens Your Disaster Recovery Plan

Avoid becoming the next cautionary tale of major service outages making the news.

5 Reasons to Attend Chaos Carnival 2023

Harness is excited to host the third annual Chaos Carnival, a FREE two-day virtual conference on March 15-16, 2023. So, have you registered yet? Here are five reasons why you should register today!

Five Key Technologies Shaping the Future of Software Delivery

If you're ready to take your software delivery to the next level, consider looking into these five key concepts that are quickly becoming essential to modern software delivery.

Harness Expands Chaos Engineering Resiliency Features with Integrated Continuous Delivery

Harness Chaos Engineering expands its chaos fault library and delivers native integration into Harness Continuous Delivery pipelines to build and validate resilience.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/products/service-reliability-management/change-impact-analysis>

Change Impact Analysis

Change Impact dashboard

Gain full visibility into every change made to your system, from deployments and infrastructure changes to feature flags and chaos experiments. Identify and triage issues quickly and confidently with our AI/ML-powered anomaly detection. Get alerts when your SLOs are not being met or changes are observed. Real time.

Accelerated incident triage

Accelerate incident triage and boost efficiency with deeper context into SLO performance and system changes right in your incident channels. Connect the dots between incidents and other change sources, such as Kubernetes infrastructure, to quickly identify the root cause.

Extensive Change Catalog

Rich set of Integrations with change sources including tools used for deployments, infrastructure, feature flags, chaos engineering, incident management and more. Deliver features progressively by carefully evaluating the impact of each feature flag toggle on the service reliability metrics. Validate that your services are meeting SLOs and error budgets before and after running chaos experiments.

Try Service Reliability Management today

Deploy faster with better reliability

Source URL: https://www.harness.io/company/contact-sales?utm_source=Website&utm_medium=internal&utm_content=Contact-Sales-CHE-Pricing

The Software Delivery Platform Business Service Teams Prefer

Want to see a custom demo or get help finding the right plan? We'd love to chat. Lean into AI/ML with Harness - the simple, scalable CI/CD platform.

- Fully Integrated Modules
- Test Intelligence
- Enterprise-Grade Security
- Fine-Grained RBAC[®] Security
- Complete Flexibility
- AI/ML Driven[®] Workflows
- Smart Feature Rollouts
- Multi-Cloud Cost Management
- Comprehensive Auditability

Contact Sales

By signing up, you agree to our Privacy Policy and our Terms of Use.

Nick Willson, TechOps Lead, GoSpotCheck

Take Harness for a spin

Source URL: <https://www.harness.io/blog-categories/sto>

STO

What is DevSecOps? Strategies to Secure Applications

This blog post will cover every important aspect you need to know about security testing and DevSecOps.

Top 10 Best Practices for DevSecOps

DevSecOps: A methodology where engineering teams run security scans throughout the SDLC to find and fix vulnerabilities before they make it to the end user. Let's learn some DevSecOps best practices.

The Modern Software Delivery Platform®

Need more info? Contact Sales

Source URL: <https://www.harness.io/legal/subscription-terms>

Subscription Terms

As of Sep 21, 2023

These Harness Subscription Terms (collectively, the "Agreement") between Harness Inc., a Delaware corporation, with its principal place of business at 55 Stockton St., 8th Floor, San Francisco, CA 94108, U.S.A. ("Harness", "we", "us" or "our") and you ("Customer", "you" or "your") applies to your use of the Harness Platform (as defined below). By clicking on the designated button, entering an Order Form (as defined below), or by downloading, installing, accessing or using the Harness Platform, you agree to the terms of this Agreement. If you are entering into this Agreement on behalf of your organization or entity, you represent that you have the authority to bind such organization or entity to the Agreement, and the terms "Customer", "you" and "your" will refer to such organization or entity. If you do not agree to the terms of this Agreement, or if you are not authorized to accept this Agreement on behalf of your organization or entity, do not download, install, access or use the Harness Platform. "Harness Platform" means the downloadable and/or online software products that are specified in the applicable Order Form, or otherwise accessed by you, and subsequent updates made generally available by Harness under this Agreement, inclusive of the Delegate. The Harness Platform excludes Third Party Products, Extensions, and Beta Services.

1. Harness Platform

1.1. License Grant. Subject to payment of the applicable fees, Harness's receipt of a purchase order number from Customer (if needed), and Customer's ongoing compliance with the terms of this Agreement, Harness grants to Customer a limited, non-exclusive, non-transferable, non-sublicensable right and license to access and use the Software for internal business purposes in accordance with the Documentation (as defined below) during the applicable License Term, only for the number of License Units (as defined below) as specifically authorized by the applicable Order Form. "License Term" means the duration of your subscription or license to the applicable Harness Platform beginning and ending on the start and end dates, respectively, specified in the applicable Order Form, which include the Initial Term (as defined in Section 5), and all Renewal Terms (as defined in Section 5), as applicable. "License Unit" means a specific license type or metric, and a numeric quantity thereof, identified in an Order Form, to establish the extent and amount of Customer's license or right to use the Harness Platform. All Order Forms are hereby incorporated into, supplement, and form a part of this Agreement. "Order Form" means an ordering document or online order that sets forth the applicable Harness Platform or Services (as defined below), fees and payment terms and start and end dates (as applicable) and is entered into between Customer and Harness, or Customer and an authorized reseller.

1.2. Restrictions on Use. Except as otherwise expressly provided in this Agreement, Customer shall not (and shall not permit any third party to): (a) sublicense, sell, resell, transfer, assign, distribute, share, lease, rent, make any external commercial use of, outsource, use on a timeshare or service bureau basis, or use in an application service provider or managed service provider environment, or otherwise generate income from the Harness Platform or Extensions; (b) copy the Harness Platform or Extensions onto any public or distributed network, except for an internal and secure cloud computing environment; (c) cause or permit the decompiling, disassembly, or reverse engineering of any portion of the Harness Platform or Extensions, or attempt to discover any source code or underlying algorithms or other operational mechanisms of the Harness Platform or Extensions (except where such restriction is expressly prohibited by law without the possibility of waiver, and then only upon prior written notice to Harness); (d) modify, adapt, translate or create derivative works based on all or any part of the Harness Platform or Extensions; (e) violate the Acceptable Use Policy ("AUP") located at

<https://harness.io/legal/aup>; or (f) use free or trial accounts for certain products after having purchased License Units for the same products; (g) attempt to probe, scan or test the vulnerability of the Harness Platform (excluding the Delegate); (h) breach the security or authentication measures of the Harness Platform without proper authorization or willfully render any part of the Harness Platform or Extensions unusable; or (i) otherwise use the Harness Platform in violation of applicable law (including any export law) or outside the scope expressly permitted hereunder and in the applicable Order Form. Additionally, Customer shall not export or re-export, directly or indirectly, any Harness Platform or technical data or any copy, portions or direct product thereof (i) in violation of any applicable laws and regulations, (ii) to any country for which the United States or any other government, or any agency thereof, at the time of export requires an export license or other governmental approval, including Cuba, Iran, North Korea, Syria, the Crimea region of Ukraine, and the so-called Donetsk People's Republic, and Luhansk People's Republic regions of Ukraine, or any other Group D:1 or E:2 country (or to a national or resident thereof) specified in the then current Supplement No. 1 to part 740 of the U.S. Export Administration Regulations (or any successor supplement or regulations, without first obtaining such license or approval) or (ii) to anyone on the U.S. Treasury Department's list of Specially Designated Nationals or the U.S. Commerce Department's Table of Denial Orders. Customer shall, at its own expense, obtain all necessary customs, import, or other governmental authorizations and approvals.

âDelegateâ means the software agent provided by Harness to Customer which facilitates Customer's use of the Harness Platform. For the purposes of this Agreement, the Delegate is not considered an Extension.

1.3. Free Access. If the Harness Platform is provided to Customer on a limited trial, free, or beta basis (âFree Accessâ), Customer agrees that such use and access of the Harness Platform is governed by this Agreement. Harness shall have the right to downgrade, limit or otherwise modify the Harness Platform provided for a Free Access account at any time without notice to Customer, and Harness disclaims all liability arising from Free Access, and shall have no liability, nor warranty, indemnity, maintenance, support, or security obligations to Customer with respect to any such Free Access. Customer may only use the number and type of License Units for the specified duration indicated by Harness prior to Customer downloading or accessing the Harness Platform with respect to any such Free Access. Harness may immediately revoke and terminate any Free Access at any time and without any notice to Customer. Customer agrees to provide feedback related to the Harness Platform as reasonably requested by Harness with respect to any Free Access. Customer agrees that Free Access is not a guarantee of any future Harness Platform or Harness product features. Customer will not utilize a Free Access account for the same products it has purchased.

1.4. Unauthorized Use. Customer shall notify Harness promptly of any unauthorized use or access of the Harness Platform (including unauthorized users or unauthorized disclosure of any password or account), or any other known or suspected breach of security or misuse of the Harness Platform. Customer is responsible for use of the Harness Platform (and all other acts or omissions) by its employees, contractors, Affiliates or other users that it allows to use or access the Harness Platform.

1.5. Support. During the License Term, Harness shall provide support to Customer in accordance with Harness's then-current support policy, and as identified in an Order Form. In the event that the level of support is not identified in the Order Form, Customer shall receive a âstandardâ level of support that is included with the Harness Platform at no additional cost. For any support tier above standard support, the applicable support fees will be a percentage of all of Customer's Harness Platform-based fees, and will be prorated for mid-year expansions based on the remaining months in the then current Initial Term or Renewal Term. Further, Customer agrees to facilitate any connections and access necessary for Harness to (i) deliver, deploy and provide the Harness Platform as provided hereunder and (ii) to perform its obligations hereunder (including any support obligations). Notwithstanding anything to the contrary in this Agreement, Harness has no warranty, indemnity or other obligation or liability with respect to modifications made to the Harness Platform or Documentation (as defined below) by Customer or on Customer's behalf other than the generally available updates provided by Harness.

1.6. Purchasing Through Authorized Resellers. If you purchase a subscription to the Harness Platform or any Services through a Harness authorized reseller (âPartnerâ), this Agreement and any agreed upon usage limitations will govern the use of such Harness Platform and Services unless otherwise agreed by Harness and Customer. You also agree that Harness is an express third party beneficiary of your agreement with any authorized reseller. Your payment obligations for the Harness Platform and Services will be with the authorized reseller as further described in Section 2.6 below, not Harness, and you will have no direct fee payment obligations to Harness, provided that Harness may terminate this Agreement if you breach any of your payment obligations to such authorized reseller for the Harness Platform and Services. Any terms agreed to between you and the authorized reseller that are in addition to or inconsistent with this Agreement are solely between you and the authorized reseller. No agreement between you and an authorized reseller is binding on Harness, nor will it have any force or effect with respect to the rights in, or the operation, use or provision of, the Harness Platform or Services.

1.7. Contractors and Third Party Providers. You may permit your authorized consultants, contractors, and agents (âThird Party Providersâ) to access and use the Harness Platform, but only on your behalf in connection with providing the Harness Platform to you, and subject to the terms and conditions of this Agreement. Any access or use by a Third Party Provider will be subject to the same limitations and restrictions that apply to you under this Agreement, and you will be responsible for any Third Party Provider's actions or omissions relating to its use of the Harness Platform. The aggregate use by you and all of your Third Party Providers must not exceed the allotted License Units (without paying the overage fees set forth in Section 2.2), and nothing in this Section is intended to or will be deemed to increase such License Units.

1.8. Services. Harness will use commercially reasonable efforts to provide the Services as described in an applicable Order Form (or statement of work referencing this Agreement entered into between the parties (âSOWâ)), if any. Harness will retain all right, title and interest in and to the deliverables and other results of the Services under this Agreement, and, subject to payment of the applicable fees and compliance with the terms of this Agreement, Harness hereby grants to Customer a limited, non-exclusive, non-transferable, non-sublicensable right and license to use such deliverables and results solely for Customer's internal business purposes and only in connection with Customer's permitted use of the Harness Platform. The Services (and any deliverables or other results) are not subject to any acceptance procedure. âServicesâ mean any training, enablement, consulting, installation and/or other professional services described in the applicable Order Form or SOW.

1.9. Customer Affiliates. Customer Affiliates may purchase and use the Harness Platform and Services subject to the terms of this

Agreement by executing Order Forms or SOWs hereunder that incorporate by reference the terms of this Agreement. In each such case, all references in this Agreement to Customer shall be deemed to refer to such Customer Affiliate for purposes of such Order Form(s) or SOW(s), and Customer Affiliate agrees to be bound by this Agreement. "Affiliate" means, with respect to Harness or Customer, any entity that directly or indirectly controls, is controlled by, or is under common control with Harness or Customer, respectively. "Control," for purposes of this definition, means direct or indirect ownership or control of more than 50% of the voting interests of the subject entity.

1.10 Security. Harness will establish and maintain appropriate administrative, technical, and physical safeguards and controls to: (i) help ensure the ongoing confidentiality, integrity, availability, and resiliency of the Harness Platform and Customer Data, and (ii) have in place a process for regularly testing, assessing and evaluating the effectiveness of technical and organizational measures to help ensure the security of the Harness Platform's processing.

1.11 Extensions. Customer may use the Extensions solely in connection with the applicable Harness Platform subject to the Documentation, open source licenses, the applicable terms within this Agreement (including with respect to the Term), and the payment of any Fees associated with the Extensions. "Extension" means any separately downloadable or accessible suite, agent, configuration file, add-on, technical add-on, plug-in, example module, command, function, playbook, content or application that enables or extends the features or functionality of the applicable Harness Platform.

1.12 Third Party Products. This Agreement does not govern Customer's use of Third Party Products used in connection with the Harness Platform. Third Party Products are governed solely by the terms and conditions between Customer and the Third Party Product developer. Harness does not make any commitments or claims regarding security, confidentiality, or performance of any Third Party Products, and specifically disclaims any liability regarding Third Party Products. Customer acknowledges and accepts that Third Party Products: (i) are activated and used at the sole risk of Customer; (ii) are not warranted, supported, or endorsed by Harness; and (iii) may degrade the performance of the Harness Platform beyond Harness's reasonable control. To the extent any Third Party Product accesses, processes, or gathers personal data, the applicable third party is Customer's direct data processor, and is not acting as a data sub-processor of Harness. "Third Party Product(s)" means any product, software, application, platform, or service (i) selected by Customer (ii) not developed by Harness, and (iii) which integrates, interacts, or interoperates with, or adds functionality to, the Harness Platform.

2. Fees

2.1. Fees. You agree to pay all fees specified in the Order Form and/or SOW, or as otherwise agreed upon by the parties. Fees are non-cancelable, non-refundable, and due and payable within thirty (30) days from the date of the invoice, or as otherwise specified in the Order Form. Unless otherwise agreed to by the parties in writing, all fees hereunder are payable in United States dollars. Additionally, Customer must provide all purchase order numbers (if applicable) to Harness by no later than the applicable Start Date as specified in the Order Form. All payments shall be made through automated clearing house (ACH) transfers, or wire transfers, to Harness's designated account, unless otherwise agreed by Harness. Fees do not include any customizations of the Harness Platform (nor support for any such customizations, unless otherwise agreed in writing).

2.2. Excess Usage. If Customer's use of the Harness Platform exceeds the number of License Units set forth in the Order Form, Customer will be billed for and Customer will pay those overages at a prorated amount for the remainder of the applicable License Term under the Order Form, based on the pricing specified in the applicable Order Form. If Harness believes in good faith that Customer's use of the Harness Platform exceeds the number of License Units set forth on the Order Form, Harness may (i) audit Customer's use of the Harness Platform (not more frequently than twice per calendar year), upon at least twenty-four (24) hours' notice, and (ii) require that Customer provide Harness with all relevant records within five (5) business days of such request in order to determine if Customer's use of the Harness Platform exceeds the number of authorized License Units.

2.3. Payment Terms. Customer acknowledges that purchases made under this Agreement are neither contingent on the delivery of any future functionality or features of the Harness Platform nor dependent on any oral or written public comments made by Harness regarding future functionality or features of the Harness Platform. All fees shall be fixed during the Initial Term (as defined in Section 5) unless Customer purchases additional License Units. If the number of purchased License Units does not increase upon renewal, then all Harness Platform and support fees will increase by the percentage specified in the Order Form at the start of each applicable Renewal Term (as defined in Section 5). If Customer is overdue on any payment, then Harness may (i) require that Customer pay a late fee equal to the lesser of 1.5% of the then-outstanding unpaid balance per month or the maximum amount allowable by law, and/or (ii) suspend Customer's use of and access to the Harness Platform and/or Services associated with Customer's account until such non-payment is corrected. Customer represents and warrants that the billing and contact information provided to Harness is complete and accurate, and Harness shall have no responsibility for any invoices that are not received due to inaccurate or missing information provided by Customer. All amounts invoiced under this Agreement and any Order Form shall be paid by Customer in full without any set-off, counterclaim, deduction, or withholding (excluding any tax withholding deductions as required by law).

2.4 Credit Cards. If Harness authorizes you to pay by credit or debit card in writing, you: (i) will provide Harness or its designated third-party payment processor with valid credit or debit card information; and (ii) hereby authorize Harness or its designated third-party payment processor to charge such credit or debit card for all items listed in the applicable Order Form or SOW or as otherwise agreed by the parties. Such charges must be paid in advance or in accordance with any different billing frequency stated in the applicable Order Form or SOW (if applicable). You are responsible for providing complete and accurate billing and contact information and notifying Harness in a timely manner of any changes to such information.

2.5 Taxes. The fees paid by Customer are exclusive of all taxes, levies, or duties ("Taxes") imposed by taxing authorities, if any, and Customer shall be responsible for payment of all such Taxes, excluding taxes based on Harness's income. Customer represents and warrants that the billing and contact information provided to Harness is complete and accurate, and Harness shall have no responsibility for any invoices that are not received due to inaccurate or missing information provided by Customer.

2.6 Payment Through Partner. Notwithstanding anything herein to the contrary, if Customer has licensed the Harness Platform from a Harness Partner, then Customer shall make its payments for the Harness Platform directly to such Partner, and the payment terms agreed

by Customer and such Partner shall supersede and govern to the extent anything in this Section 2 conflicts with such payment terms.

3. Confidentiality

3.1. Confidential Information and Restrictions. âConfidential Informationâ means all information disclosed by a party (âDisclosing Partyâ) to the other party (âReceiving Partyâ), whether orally or in writing, that is designated as âconfidentialâ or âproprietary,â or that, given the nature of the information or circumstances surrounding its disclosure, should reasonably be understood to be confidential. âConfidential Informationâ does not include any information that: (i) is or becomes generally known to the public without breach of any obligation owed to the Disclosing Party, (ii) was known to the Receiving Party prior to its disclosure by the Disclosing Party without breach of any obligation owed to the Disclosing Party, (iii) is received from a third party without breach of any obligation owed to the Disclosing Party, or (iv) was independently developed by the Receiving Party. The Receiving Party will: (i) not use the Disclosing Partyâs Confidential Information for any purpose outside of this Agreement; (ii) not disclose such Confidential Information to any person or entity, other than its Affiliates, employees, consultants, subcontractors, subprocessors, agents and professional advisers (âRepresentativesâ) who have a âneed to knowâ for the Receiving Party to exercise its rights or perform its obligations hereunder, provided that such employees, consultants, and agents are bound by agreements or, in the case of professional advisers, ethical duties respecting such Confidential Information in accordance with the terms of this Section 3; and (iii) use reasonable measures to protect the confidentiality of such Confidential Information. The Receiving Party shall be liable for any breach of this section by its Representatives. If the Receiving Party is required by applicable law or court order to make any disclosure of such Confidential Information, it will first give written notice of such requirement to the Disclosing Party, and, to the extent within its control, permit the Disclosing Party to intervene in any relevant proceedings to protect its interests in its Confidential Information, and provide full cooperation to the Disclosing Party in seeking to obtain such protection. Further, this Section 3 will not apply to information that the Receiving Party can document: (i) was rightfully in its possession or known to it prior to receipt without any restriction on its disclosure; (ii) is or has become public knowledge or publicly available through no fault of the Receiving Party; (iii) is rightfully obtained by the Receiving Party from a third party without breach of any confidentiality obligation; or (iv) is independently developed by employees of the Receiving Party who had no access to such information.

3.2. Equitable Relief. The Receiving Party acknowledges that unauthorized disclosure of the Disclosing Partyâs Confidential Information could cause substantial harm to the Disclosing Party for which damages alone might not be a sufficient remedy and, therefore, that upon any such disclosure by the Receiving Party the Disclosing Party will be entitled to seek appropriate equitable relief in addition to whatever other remedies it might have at law or equity.

3.3. Feedback. Customer acknowledges and agrees that (a) any questions, comments, suggestions, ideas, feedback or other information about Harness, the Harness Platform, Extensions, the Services, the Documentation or other materials provided by Harness (collectively, âFeedbackâ) provided by Customer are non-confidential, (b) Harness will have full discretion to determine whether or not to proceed with the development of any requested enhancements, new features or functionality, and (c) Harness will have the full, unencumbered right, without any obligation to compensate or reimburse Customer, to use, incorporate and otherwise fully exercise and exploit any such Feedback in connection with its products and services.

4. Proprietary Rights

4.1 Ownership. Harness owns and shall retain all proprietary rights, including all copyright, patent, trade secret, trademark and all other intellectual property rights, in and to the Harness Platform (and all derivatives, improvements or enhancements thereof), System Data, Delegate, Extensions, Documentation, the results generated by the Harness Platform, and any Services, or any materials generated by Harness.Â

4.2 FOSS; Third Party Components. Customer acknowledges that the rights granted under this Agreement do not provide Customer with title to or ownership of the Harness Platform, in whole or in part. Certain âfreeâ or âopen sourceâ based software (the âFOSS Softwareâ) and third party software included with the Harness Platform (the âThird Party Softwareâ) is shipped with the Harness Platform but is not considered part of the Harness Platform hereunder. Use, reproduction, and distribution of FOSS Software is governed by the terms of the applicable open source software license and not this Agreement. Harness will provide Customer with a list of the FOSS Software and Third Party Software embedded in the Harness Platform upon request. With respect to Third Party Software included with the Harness Platform, such Third Party Software suppliers are third party beneficiaries of this Agreement. The Harness Platform and Third Party Software may only be used and accessed by Customer as prescribed by the instructions, code samples, on-line help files and technical documentation made publicly available by Harness for the Harness Platform, as may be updated from time to time by Harness (the âDocumentationâ). Harness will not be responsible for any act or omission of any third party, including the third partyâs access to or use of any Customer data or the performance of the Harness Platform in combination with such Third Party Software.

5. Term and Termination

This Agreement will be in full force and effect beginning on the earlier of the start date of the first Order Form entered into hereunder and the date you first access or otherwise use the Harness Platform, and will remain in effect until this Agreement is terminated pursuant to this Section. Termination of a specific Order Form will not affect the effectiveness of this Agreement or any other Order Form. Unless indicated otherwise in an Order Form, each Order Form shall be valid from the earliest start date therein through the initial end date therein (the âInitial Termâ), and shall automatically renew for additional successive twelve (12) month terms (each, a âRenewal Termâ), unless either party provides notice of non-renewal no less than thirty (30) days prior to the end of then-current Initial Term or Renewal Term, as applicable. If either party commits a material breach of this Agreement, and such breach has not been cured within thirty (30) days after receipt of written notice thereof, the non-breaching party may terminate this Agreement, except that Harness may immediately terminate this Agreement and/or terminate or suspend Customerâs use of and access to the Harness Platform associated with Customerâs account upon Customerâs breach of Section 1.2 (Restrictions on Use) or Section 2.1 (Fees). Additionally, Harness may temporarily suspend access to the Harness Platform if Customerâs use poses a security risk or adversely impacts Harnessâs business. Either party may also terminate this Agreement upon written notice if (a) the other party suspends payment of its debts or experiences any other

insolvency or bankruptcy-type event or (b) there are no Order Forms or SOWs then in effect. Upon expiration or termination of an Order Form, for any reason, all rights granted to Customer with respect to such Order Form shall terminate and Customer shall destroy any copies of the Harness Platform and Documentation provided under such Order Form within Customer's possession and control. Upon any termination of this Agreement, each Receiving Party will return or destroy, at the Disclosing Party's option, the Disclosing Party's Confidential Information in the Receiving Party's possession or control. All payment obligations that have accrued as of such expiration or termination, any other rights or obligations that by their nature should survive, along with Sections 1.2, 1.3, 1.4, 1.6, 1.7, 2, 3, 4, 5, 6.2 and 7 through 10, will survive any expiration or termination hereof.

6. Warranties

6.1. Harness Platform Warranty. Harness warrants that during the first thirty (30) days after the beginning of a License Term under the applicable Order Form, the Harness Platform will, in all material respects, conform to the functionality described in the then-current Documentation for the applicable version of the Harness Platform. Harness's sole and exclusive obligation, and Customer's sole and exclusive remedy, for a breach of this warranty shall be that Harness will use commercially reasonable efforts to repair or replace the Harness Platform to conform in all material respects to the Documentation, and if Harness is unable to materially restore such functionality within thirty (30) days from the date of written notice of breach of this warranty by Customer, Customer shall be entitled to terminate the applicable Order Form upon written notice to Harness, and Harness shall promptly provide a pro-rata refund of the subscription fees under such Order Form that have been paid in advance for the remainder of the License Term under such Order Form (beginning on the date of termination). To be eligible for the foregoing remedy, Customer must notify Harness in writing of any warranty breaches within such warranty period, and Customer must have installed (if applicable), used and configured the Harness Platform in accordance with this Agreement and the Documentation.

6.2. Disclaimer. EXCEPT AS EXPRESSLY PROVIDED IN THIS SECTION 6, THE HARNESS PLATFORM, DOCUMENTATION, SERVICES, MAINTENANCE AND SUPPORT ARE PROVIDED "AS IS," AND HARNESS AND ITS SUPPLIERS EXPRESSLY DISCLAIM ANY AND ALL OTHER REPRESENTATIONS AND WARRANTIES, EITHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT THERETO, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, NON-INFRINGEMENT, OR THE CONTINUOUS, UNINTERRUPTED, ERROR-FREE, VIRUS-FREE, OR SECURE ACCESS TO OR OPERATION OF THE HARNESS PLATFORM. HARNESS EXPRESSLY DISCLAIMS ANY WARRANTY AS TO THE ACCURACY OR COMPLETENESS OF ANY INFORMATION OR DATA ACCESSED OR USED IN CONNECTION WITH THE HARNESS PLATFORM, DOCUMENTATION, SERVICES, MAINTENANCE OR SUPPORT. Additionally, Harness is not responsible for any delays, delivery failures, or any other loss or damage resulting from the transfer of data over communications networks and facilities, including the Internet, and Customer acknowledges that the Harness Platform, Delegate, Extensions, Services and Documentation may be subject to limitations, delays and other problems inherent in the use of such communications facilities. The Harness Platform is not fault-tolerant and is not designed or intended for use in hazardous environments, including without limitation, in the operation of aircraft or other modes of human mass transportation, nuclear or chemical facilities, life support systems, implantable medical equipment, motor vehicles or weaponry systems, or any other application in which failure of the Harness Platform could lead to death or serious bodily injury of a person, or to severe physical or environmental damage (each, a "High Risk Use"). Harness expressly disclaims any express or implied warranty or representation of fitness for High Risk Use. Harness shall not be liable to Customer for any loss, damage or harm suffered by Customer that is directly or indirectly caused by Customer's unauthorized use of the Harness Platform to process Prohibited Data. Prohibited Data means: (a) special categories of data enumerated in European Union Regulation 2016/679, Article 9(1) or any successor legislation; (b) patient, medical, or other protected health information regulated by the Health Insurance Portability and Accountability Act (as amended and supplemented) ("HIPAA"); (c) credit, debit, or other payment card data or financial account information, including bank account numbers or other personally identifiable financial information; (d) social security numbers, driver's license numbers, or other government identification numbers; (e) other information subject to regulation or protection under specific laws such as the Children's Online Privacy Protection Act or Gramm-Leach-Bliley Act ("GLBA") (or related rules or regulations); or (f) any data similar to the above protected under foreign or domestic laws.

6.3. Mutual Warranty. Each party hereby represents and warrants to the other that: (a) such party has the right, power, and authority to enter into this Agreement and to fully perform all of its obligations hereunder, (b) entering into this Agreement does not and will not violate any agreement or obligation existing between such party and any third party, and (c) this Agreement, when executed and delivered, will constitute a valid and binding obligation of such party and will be enforceable against such party in accordance with its terms.

6.4. Beta Software. From time to time, Customer may have the option to participate in a program with Harness where Customer is given access to alpha or beta software, services, products, features and documentation (collectively, "Beta Software") offered by Harness. The Beta Software is not generally available and may contain bugs, errors, defects or harmful components. Accordingly, Harness is providing the Beta Software to Customer "as is." Notwithstanding anything to the contrary in this Agreement, Harness makes no warranties of any kind with respect to the Beta Software, whether express, implied, statutory or otherwise, including any implied warranties of merchantability, fitness for a particular purpose, or non-infringement, and has no indemnity or other obligation or liability with respect to Beta Software. Harness does not warrant that the Beta Software will meet any specified service level, or will operate without interruptions or downtime.

7. Indemnification

7.1. By Harness. Harness agrees to defend, at its expense, Customer against (or, at Harness's sole option, settle) any third party claim to the extent such claim alleges that the Harness Platform infringes or misappropriates any patent, copyright, trademark or trade secret of a third party, and Harness shall pay all costs and damages finally awarded against Customer by a court of competent jurisdiction as a result of any such claim.

7.2. Remedies. In the event that the use of the Harness Platform is, or in Harness's sole opinion is likely to become, subject to such a claim, Harness, at its option and expense, may (a) replace the applicable Harness Platform with functionally equivalent non-infringing

technology, (b) obtain a license for Customerâs continued use of the applicable Harness Platform, or (c) terminate the applicable Order Form and provide a pro-rata refund of the subscription fees under such Order Form that have been paid in advance for the remainder of the License Term under such Order Form (beginning on the date of termination).

7.3. Limitations. The foregoing indemnification obligation of Harness will not apply: (1) if the Harness Platform is or has been modified by Customer or its agent; (2) if the Harness Platform is combined with other non-Harness products, applications, or processes, but solely to the extent the alleged infringement is caused by such combination; (3) to any unauthorized use of the Harness Platform or breach of this Agreement; or (4) if Customer fails to install or use any functionally equivalent non-infringing aspect of the Harness Platform that would have avoided the alleged infringement. The foregoing shall be Customerâs sole remedy with respect to any claim of infringement of third party intellectual property rights.

7.4 By Customer. Customer agrees to defend, at its expense, Harness and its Affiliates, its suppliers and its resellers against any third party claim to the extent such claim alleges, arises from, or is made in connection with Customerâs breach of Section 1 (Harness Platform) or Section 10 (Data Use), any High Risk Use, or Customerâs negligence or willful misconduct. Customer shall pay all costs and damages finally awarded against Harness by a court of competent jurisdiction as a result of any such claim.

7.5 Indemnification Requirements. In connection with any claim for indemnification under this Section 7, the indemnified party must promptly provide the indemnifying party with notice of any claim that the indemnified party believes is within the scope of the obligation to indemnify, provided, however, that the failure to provide such notice shall not relieve the indemnifying party of its obligations under this Section 7, except to the extent that such failure materially prejudices the indemnifying partyâs defense of such claim. The indemnified party may, at its own expense, assist in the defense if it so chooses, but the indemnifying party shall control the defense and all negotiations related to the settlement of any such claim. Any such settlement intended to bind either party shall not be final without the other partyâs written consent, which consent shall not be unreasonably withheld, conditioned or delayed; provided, however, that Customerâs consent shall not be required when Harness is the indemnifying party if the settlement involves only the payment of money by Harness.

8. Limitation of Liability

The limits below will not apply to the extent prohibited by applicable law. EXCEPT FOR LIABILITY ARISING FROM VIOLATIONS OF SECTION 1.2 (RESTRICTIONS ON USE), CUSTOMERâS PAYMENT OBLIGATIONS, OR A PARTYâS INFRINGEMENT OR MISAPPROPRIATION OF THE OTHER PARTYâS INTELLECTUAL PROPERTY RIGHTS, UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER IN TORT, CONTRACT, OR OTHERWISE, WILL EITHER PARTY BE LIABLE TO THE OTHER PARTY UNDER THIS AGREEMENT FOR ANY (A) INDIRECT, SPECIAL, INCIDENTAL, CONSEQUENTIAL OR PUNITIVE DAMAGES OF ANY CHARACTER, INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, LOST PROFITS, LOST SALES OR BUSINESS, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, LOST DATA, OR FOR ANY AND ALL OTHER INDIRECT DAMAGES OR LOSSES, EVEN IF SUCH PARTY HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY OF SUCH DAMAGES, OR (B) DIRECT DAMAGES, COSTS, OR LIABILITIES IN EXCESS OF THE AMOUNTS PAID BY CUSTOMER DURING THE TWELVE (12) MONTHS IMMEDIATELY PRECEDING THE INCIDENT OR CLAIM UNDER THE APPLICABLE ORDER FORM OR SOW. THESE LIMITATIONS SHALL APPLY NOTWITHSTANDING ANY FAILURE OF ESSENTIAL PURPOSE OF ANY REMEDY.

9. Data Use

9.1 System Data. Harness shall have the right collect and analyze System Data (including, without limitation, information concerning Customer and data derived therefrom), and Harness will be free (during and after the term hereof) to (i) use such information and data to improve and enhance the Harness Platform and for other development, diagnostic and corrective purposes in connection with the Harness Platform and other Harness offerings, and (ii) disclose such data solely in aggregate or other de-identified form in connection with its business. âSystem Dataâ means data collected by Harness regarding the Harness Platform that may be used to generate logs, statistics or reports regarding the performance, availability, usage, integrity or security of the Harness Platform.

9.2 Customer Data and License to Customer Data. âCustomer Dataâ means electronic data submitted by Customer through the Harness Platform. As between the parties, Customer exclusively owns and reserves all rights, title, and interest in and to the Customer Data and Customerâs software applications (and all derivatives, modifications, improvements, or enhancements related to any of the foregoing), which includes all copyright, patent, trade secret, trademark and all other intellectual property rights related to any of the foregoing. Customer hereby grants to Harness and its Affiliates a non-exclusive right to access, use, and process Customer Data, as necessary to provide the Harness Platform and the Services in accordance with this Agreement, and to improve the functioning and usability of the Harness Platform. Customer is solely responsible for the quality and integrity of Customer Data. Customer represents and warrants that it has obtained all the necessary consents to provide Harness with the foregoing license to Customer Data.

9.3 Personal Identifiable Information. If Customer provides Harness with any personally identifiable information (âPersonal Dataâ), Customer represents and warrants that such Personal Information is not Prohibited Data, has been collected by Customer in accordance with the provisions of all applicable data protection laws and regulations, and that Customer has all right and consents necessary to provide such Personal Data to Harness.

10. Miscellaneous

10.1 Governing Law; Venue. This Agreement shall be governed by and construed under the laws of the State of California, U.S.A, as if performed wholly within the state and without giving effect to the principles of conflict of law. The parties consent to the exclusive jurisdiction and venue of the courts located in and serving San Francisco, California. The Uniform Computer Information Transactions Act (UCITA) nor the United Nations Convention for the International Sale of Goods will apply to this Agreement.

10.2 No Waiver. Failure by either party to exercise any of its rights under, or to enforce any provision of, this Agreement will not be

deemed a waiver or forfeiture of such rights or ability to enforce such provision. If any provision of this Agreement is held by a court of competent jurisdiction to be illegal, invalid or unenforceable, such provision will be amended to achieve as nearly as possible the same economic effect of the original provision and the remainder of this Agreement will remain in full force and effect.Â

10.3 Entire Agreement. This Agreement (including each Order Form and SOW) represents the entire agreement between the parties and supersedes any previous or contemporaneous oral or written agreements or communications regarding the subject matter of this Agreement.Â

10.4 Order of Precedence. This Agreement shall control over additional or different terms of any purchase order, confirmation, invoice, statement of work or similar document (other than the Order Form or SOW, which will take precedence), even if accepted in writing by both parties, and waivers and amendments to this Agreement shall be effective only if made by non-pre-printed agreements clearly understood by both parties to be an amendment or waiver to this Agreement.Â

10.5 Interpretation. All capitalized terms used but not defined in an Order Form or SOW shall have the meanings provided to them in the Agreement. For purposes of this Agreement, âincludingâ means âincluding without limitation.âÂ

10.6 Severability. If any provision of this Agreement is held by a court of competent jurisdiction to be illegal, invalid or unenforceable, such provision will be amended to achieve as nearly as possible the same economic effect of the original provision and the remainder of this Agreement will remain in full force and effect.

10.7 Cumulative Remedy. The rights and remedies of the parties hereunder will be deemed cumulative and not exclusive of any other right or remedy conferred by this Agreement or by law or equity.Â

10.8 Independent Contractors. No joint venture, partnership, employment, or agency relationship exists between the parties as a result of this Agreement or use of the Harness Platform.Â

10.9 Assignment and Delegation. Customer may not assign this Agreement without the prior written consent of Harness, and any purported assignment in violation of this Section 10.9 shall be void. Harness may assign, transfer or subcontract this Agreement in whole or in part without Customerâs consent. Upon any assignment of this Agreement by Customer that is approved by Harness or other corporate transaction involving Customer that would materially increase its Licensee Unit usage, if the Order Form contains a subscription for an âunlimitedâ amount of Licensee Units, such subscription will, with respect to Customer or the successor entity, as applicable, be capped at the monthly average of authorized Licensee Units used by Customer under such Order Form during the three full calendar months prior to such assignment (or if the Harness Platform has been used for fewer than three full calendar months, then the monthly average based on a pro rata calculation of such use). Harness reserves the right to perform its obligations from locations and/or through use of Affiliates, contractors and subcontractors, worldwide, provided that Harness will be responsible for such parties.

10.10 Force Majeure. With the exception of Customerâs payment obligations, no delay, failure, or default, other than a failure to pay fees when due, will constitute a breach of this Agreement to the extent caused by epidemics, acts of war, terrorism, hurricanes, earthquakes, cyberattacks, other acts of God or of nature, strikes or other labor disputes, riots or other acts of civil disorder, embargoes, government orders responding to any of the foregoing, or other causes beyond the performing partyâs reasonable control.

10.11 Publicity. Customer agrees that Harness may refer to Customer by its trade name and logo, and may briefly describe Customerâs business, in Harnessâs marketing materials and website. Additionally, Customer and Harness shall collaborate in good faith for the purpose of executing various co-marketing activities (e.g., customer testimonial videos, case study write ups, conference speaking slots, serving as a referenceable customer, etc.). The parties agree that all co-marketing activities will be contingent on a successful deployment of the Harness Platform.

10.12 Notice. Harness may give notice to Customer by electronic mail to Customerâs email address as provided by Customer on the Order Form or on record in Customerâs account information, or by written communication sent by first class mail or pre-paid post to Customerâs address as provided by Customer on the Order Form or on record in Customerâs account information. Customer may give notice to Harness at any time by any letter delivered by nationally recognized overnight delivery service or first class postage prepaid mail to Harness at the following address or such other address as may be notified to Customer from time to time: Harness, 55 Stockton St., 8th Floor, San Francisco, CA 94108, Attn.: Legal Department. Notice under this Agreement shall be deemed given when received, if personally delivered; when receipt is electronically confirmed, if transmitted by email; the day after it is sent, if sent for next-day delivery by a recognized overnight delivery service; and upon receipt, if sent by certified or registered mail, return receipt requested.Â

10.13 Updates. Harness may update this Agreement from time to time by providing you with prior written notice of material updates at least thirty (30) days in advance of the effective date. Such notice will be given in accordance with this section. Except as otherwise specified by Harness, (a) updates will be effective upon the effective date indicated at the top of this Agreement or in such notice, (b) your continued access or use of the Harness Platform or Services on or after the effective date of such updates constitutes your acceptance of such updates and (c) if you do not agree to such updates, you should stop using the Harness Platform and Services. However, if you have paid for a subscription to the Harness Platform, and we update this Agreement during your License Term, the updates with respect to that subscription will be effective upon your next Renewal Term, if applicable, and in this case, if you object to the updates, as your sole and exclusive remedy, you may choose not to renew, in accordance with the terms hereof. The updated version of the Agreement will supersede all prior versions.

Source URL: <https://www.harness.io/blog-categories/quality>

Quality

Hygiene in SDLC: A Key to Engineering Efficiency

Explore the importance of hygiene in software development with our blog. Discover how it ensures top-notch software quality and insights. Get practical tips for improvement and join us in fostering a culture of continuous excellence. Prioritize quality and efficiency for success in the software world!

Choice Hotels Deploys 85% Faster Using Harness

Harness helps us capitalize on the investments we've made in automation tools by streamlining everything into a single pipeline.

Metrikus Shortens Commit-to-Production Time by 66%

Metrikus made feature flag management easy with Harness. See how you can too.

Single Digits Saves \$108k in Deployment Effort

Single Digits made their migration easy with Harness. See how you can too.

Burst SMS Removes Outage Risk for Their Customers

Harness delivers better DevOps practices to all our customers. Find out how Burst SMS used Harness to reduce outages.

How Zeotap Increased Deployment Velocity 36x

From Downtime Windows to Deployment Nirvana: Learn How Zeotap Increased Deployment Velocity 36x.

Ancestry Cuts Downtime by 50% Using Harness Governance

Find out how Ancestry secured their platform using Harness.

Automated CI CD Rollback to 30 Seconds

Build.com removed most of their verification effort. Find out how you can too.

Empowers Developers with Harness

Vuclip found that Harness was the only solution that fully integrated with their GCP, Kubernetes, APM, and Log stack.

The Modern Software Delivery Platform[®]

Need more info? Contact Sales

Source URL: <https://www.harness.io/legal/harness-social-media-sweepstakes>

Harness Social Media Sweepstakes

As of

â

Sweepstakes Name: Harness Pipeline Challenge

Prize: Win one year subscription to ChatGPT Plus!

Official Rules

NO PURCHASE NECESSARY TO ENTER OR WIN.

A PURCHASE OR PAYMENT WILL NOT IMPROVE YOUR CHANCES OF WINNING.

â

Eligibility

This Harness Pipeline Challenge ("Sweepstakes") is open to all natural persons who are legal residents of the 50 United States, District of Columbia, the United Kingdom, Canada (excluding Quebec), Chile, Ecuador, India, Indonesia, Japan, New Zealand, Nigeria, Singapore, South Africa, Switzerland, and Taiwan who are at least 18 years old at time of entry. Void outside the countries and states

listed above, in Puerto Rico, the U.S. Virgin Islands, and other U.S. territories and possessions. The Sweepstakes is subject to all applicable federal, state, and local laws and regulations and is void where prohibited by law.

â

Sponsorship

The Sweepstakes sponsor is Harness Inc. (âSponsorâ), located at 55 Stockton St. 8th Floor, San Francisco, CA 94108. Sponsor and its respective affiliates, subsidiaries, successors, assigns, agents, representatives, officers, directors, shareholders, and employees, and any entity involved in the development, production, implementation, administration, judging or fulfillment of the Sweepstakes, including without limitation, the immediate family members of such individuals, are not eligible to participate. Sponsor will conduct the Sweepstakes substantially as described in these Official Rules.

â

Sweepstakes Period

This Sweepstakes begins on April 17, 2023 at 12:01 AM PST and ends on April 30, 2023 at 11:59 PM PST (âSweepstakes Periodâ).

â

How to Enter

During the Sweepstakes Period, visit the product sign-up page located on

Sponsorâs website at <https://app.harness.io/auth/#/signup> and follow the instructions to electronically complete and submit the Harness freemium account form. You will be asked to provide basic contact information, including your work email address. (NOTE: You must complete all required fields of the entry form with information that is valid and accurate as of the date of your submission to be eligible). Once submitted, you will receive a verification email and once your identity is verified, you will be granted access to the Harness platform. Participants must complete the following minimum requirements to enter:

This Sweepstakes is in no way sponsored, endorsed or administered by, or associated with, any of the social media companies that the Sponsor used to promote the Sweepstakes, nor by any of their subsidiaries or affiliates.

â

Limitations on Deployments

LIMIT 1 CI BUILD OR CD PIPELINE SUBMISSION PER PERSON PER EMAIL ADDRESS. Only your first CI build or CD pipeline that is submitted to Sponsor as specified above will count towards the Sweepstakes (a âQualifying Submissionâ). No third party entry or entry through any Sweepstakes service is permitted. Any person who attempts or otherwise encourages the entry of multiple or false contact information under multiple identities, email addresses, etc., or uses any device or artifice to enter or encourage multiple or false entries, as determined by Sponsor in its sole discretion, will be disqualified. Use of robotic entry devices is strictly prohibited.

â

Limitations on Time to Compete

Participants have the length of the Sweepstakes Period to submit a Qualifying Submission. All Qualifying Submissions received after the expiration of the Sweepstakes Period will be excluded from the Sweepstakes and will not count.

â

Alternative Method of Entry

If you do not wish to submit your CI build or CD pipeline, you may enter the Sweepstakes by sending a 3x5 inch card with your full name, complete mailing address, email address, and telephone number to the Sponsor at 55 Stockton St., 8th Floor, San Francisco, CA 94108. Such alternative method of entry shall count as a Qualifying Submission. Entries must be postmarked and received by the end of the Sweepstakes Period. Limit one mailed entry per person. All entries become the property of the Sponsor and will not be returned. The Sponsor is not responsible for lost, late, illegible, incomplete, or misdirected entries.

â

Winning Prize

One (1) winner will be selected at random and awarded a gift card equal to \$240, equivalent to the current US listed price for a yearly subscription to ChatGPT Plus (the âPrizeâ). All other expenses not specifically mentioned herein are solely the winnerâs responsibility. The winner understands there are inherent risks involved in the use of generative AI. The winner agrees to accept the prize âas isâ and entrants hereby acknowledge that Sponsor is not in any manner responsible or liable for any warranty, representation, or guarantee related to the Prize, including warranties provided exclusively by Open AI. Additional terms and conditions may apply to Chat GPT Plus for which Sponsor shall not be responsible or liable in any way for any losses or damages whatsoever.

- The winner will be contacted via Twitter Direct Message from @Harnessio or have their LinkedIn post commented on by the Harness handle.
- The winner must affirm acceptance of all terms and conditions, and upon request, provide proof of residence or other eligibility requirements outlined in the Sweepstakes rules.
- Winners must complete the above actions within 7 days of notification or Sponsor will consider the prize forfeited and select an alternate winner using the same random selection process mentioned above.
- Sponsor will:
 - â Arrange for delivery of the Prize to the winner
 - â Ensure there is a signature or other proof of delivery

Winner is solely responsible for any federal, state, and local taxes. Upon completion of the Sweepstakes and fulfillment of the Prize, Sponsor will announce the giveaway winner publicly on the Sponsor's social media handles.

â

Odds

The odds of winning will depend on the number of Qualifying Submissions received.

â

WARNING:

âANY ATTEMPT BY ANY PERSON, WHETHER OR NOT AN ENTRANT, TO DELIBERATELY DAMAGE, DESTROY, TAMPER WITH OR VANDALIZE ANY SPONSOR-OWNED WEBSITE OR APPLICATION OR RELATED SOCIAL NETWORKING SITE, THE ENTRY PROCESS, OR OTHERWISE INTERFERE WITH OR UNDERMINE THE LEGITIMATE OPERATION OF THE SWEEPSTAKES, MAY BE A VIOLATION OF CRIMINAL AND CIVIL LAWS AND SPONSOR RESERVES THE RIGHT TO SEEK DAMAGES AND DILIGENTLY PURSUE ALL REMEDIES AGAINST ANY SUCH PERSON TO THE FULLEST EXTENT PERMITTED BY LAW.

â

General Terms

âBy entering this Sweepstakes, you promise to abide by the Official Rules and decisions of Sponsor, which will be final and binding in all respects. Sponsor reserves the right, at its sole discretion, to refuse, disqualify or withdraw any entry at any time. Sponsor will not be responsible for any injury, damage or loss of any kind arising out of your participation in the Sweepstakes or from the Prize. YOU AGREE TO RELEASE, DISCHARGE, AND HOLD HARMLESS SPONSOR, ITS AFFILIATES, SUBSIDIARIES, EMPLOYEES, OFFICERS, DIRECTORS, AGENTS AND ASSIGNS FROM AND AGAINST ANY OR ALL CLAIMS, LOSSES, INJURIES, OR DAMAGES RESULTING FROM PARTICIPATION IN THE SWEEPSTAKES OR FROM THE PRIZE.â Except where prohibited by law, by accepting the Prize, winner grants Sponsor a perpetual, worldwide, royalty-free license to use winner's name, photograph, voice, and/or likeness without further authorization, compensation, or remuneration of any kind for advertising, promotion and other publicity purposes in any and all media now or hereafter known throughout the world.

â

Questions?

If you have any questions drop a note to corp.marketing@harness.io.

Source URL: <https://www.harness.io/legal/aida-terms>

Harness AI Development Assistant (AIDA) Terms

As of

These Harness AI Development Assistant Terms ("AIDA Terms") is a legal agreement between you (referred to as "User" or "you") and Harness Inc. (referred to as "Harness" or "we"), that governs your use of AIDA. By accessing or using AIDA, you agree to be bound by the AIDA Terms, along with the Underlying Terms and Privacy Policy (collectively, the âAgreementâ).â

â

Capitalized terms not defined here have their respective meanings given in the Underlying Terms.â

â

Updated June 2023

Source URL: <https://www.harness.io/blog/fastest-ci-tool>

The Data is In: Harness CI is up to 4X faster than Other Solutions

When I started Drone over 10 years ago as an open source project, I was frustrated by how much time it took to ship code with existing CI tools. It made for a bad developer experience, and I wanted to solve that problem. My vision was to create a CI tool that was fast, simple, open, and secure. Iâm excited to have realized that goal now with Harness CI.

On the heels of announcing several new speed enhancements for the Harness Continuous Integration (CI) module, Iâm excited to share our test data showing that **Harness CI builds up to four times faster** than other leading CI solutions. These fast build times shown in our test results validate the impact of three important new feature enhancements: Cache Intelligence, Test Intelligenceâ€¢ (a technology exclusive to Harness), and Hosted Builds â and, more importantly, they help to create a great developer experience.

At Harness, we understand that developers are frustrated with slow, flaky builds. Our focus is to remove friction points for developers, so they can focus on writing code and shipping it quickly, efficiently, reliably, and securely. Harness CIâs latest speed-enhancing features are a testament to this commitment.

You can read more about these new feature enhancements in our blog announcement, or dive deeper into the technical details on our CI product features page. In this blog, weâll take a closer look at this impressive new data, as well as how Harness CI can save businesses up to 300% in infrastructure costs as a result of these recent innovations.Â

The Data Behind Four Times FasterÂ

As we looked at the initial test results from our latest CI feature enhancements, we knew we were seeing unprecedentedly fast build times. We also knew immediately that we had to show developers just how much faster we could help them build.Â

We designed our tests to run the same builds with Harness CI, GitHub Actions, and another popular CI tool across the Apache Kafka, Apache RocketMQ, and Apache Zookeeper repositories. The chart below shows those results in minutes. (*Note, the Harness âbest caseâ results were achieved where the code changes were minimal and there was no test case run (e.g. this was an actual code change in the open source repo where very few test cases were identified to run).*)

The median test results show Harness CI is two to five times faster than the nearest competitor and three to four times faster than GitHub Actions. In some cases, Harness was 19 to 75 times faster than GitHub Actions.Â Â Â

Apache Kafka Build Times

Apache Kafka is a popular distributed event streaming platform with over 23,500 stars on GitHub. Itâs also used by 80% of the Fortune 100 companies for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications.

Below, you can see the build time results we recorded with the Apache Kafka repository.

Over the course of 50+ pipeline runs:

- GitHub Actions completed the Kafka build in 19 minutes
- CI Vendor 2 completed the Kafka build in 15 minutes
- Harness averaged five minutes to complete the build with a best case build time of 24 seconds

Since the Apache Kafka repo is a very large project, we also used it to demonstrate the substantial time savings gained from our Test Intelligence feature, a unique offering in the market. Using machine learning (ML), Test Intelligence determines which tests are necessary based on the code changes. In our demonstration, Test Intelligence effectively reduced the number of unit tests from 16,000 to 700. But thatâs not all â Test Intelligence split the necessary tests and ran them in parallel.Â

Weâll take a closer look at how Test Intelligence works and those results after we detail the other tests and the methodology we used.

Apache RocketMQ

Apache RocketMQ is a distributed messaging and streaming platform with over 18,000 stars on GitHub that makes it easier to build event-driven applications.

The chart below shows the build time test results from the Apache RocketMQ repository tests.

Over the course of 50+ pipeline runs:

- GitHub Actions completed the RocketMQ build in 19 minutes
- The CI Vendor 2 completed the RocketMQ build in 15 minutes
- Harness averaged three minutes and 18 seconds to complete the build with a best case build time of one minute

Apache Zookeeper

Apache Zookeeper has over 10,000 stars on GitHub and aims to make maintaining and coordinating distributed systems easier.

You can see our build time test results from the Apache Zookeeper repository in the chart below.Â

Over the course of 50+ pipeline runs:

- GitHub Actions completed the Zookeeper build in 19 minutes
- The CI Vendor 2 completed the Zookeeper build in 13 minutes
- Harness averaged six minutes to complete the build with a best case build time of 15 seconds

The Test Methodology

We chose to test these three projects because of their popularity among the open source community and the high rate of adoption by development teams at enterprise organizations, including Goldman Sachs, Target, Intuit, Cisco, and more. Since we used open source projects, we also had the ability to replicate the same test scenario for each build.Â

We ran builds from these projects using the free offerings from GitHub Actions, another popular CI vendor, and Harness. The performance numbers were calculated by building the over 50 pull requests from each repository. This allowed us to test using actual code commitsÂ to the open source repositories in order to simulate a real use case that developers encounter every day. We used the out-of-the-box default configuration for each build run rather than fine-tuning the queries, because we know that developers have a better experience when they're not required to make extra modifications.Â

You can see a screenshot of the actual pull request from the tests run in the Kafka repository below:Â

Caching and Testing Speed Data

The speed results that Harness CI demonstrated in these tests were achieved by our latest feature enhancements. Let's take a closer look at how each of these differentiating features impacted the test results.Â

Cache Intelligence Time Savings

Software development is inherently complex with various dependencies across build tools, package managers, and more. All this complexity increases build times. Developers don't always remember to manually turn on caching or specify all of the necessary paths to store the right data in for each directory. By automatically caching well-known directories for Java and Node.js, Cache Intelligence helps developers spend less time waiting on builds to complete, which means more time coding.Â

Test Intelligenceâ€¢ Time Savings

Harness CI's exclusive, ML-powered Test Intelligenceâ€¢ delivers even more time savings now with the addition of test concurrency. Test Intelligenceâ€¢ determines which subset of tests is necessary based on code changes while providing visibility into which tests are selected and why. After the required tests are identified, these tests run concurrently to further accelerate build time without compromising quality.Â

The chart below details the performance enhancement with Test Intelligenceâ€¢ when 10 files were changed in the same Kafka pull request. Test Intelligenceâ€¢ split and ran tests for just those 10 files in parallel, rather than running them sequentially. As a result, Harness ran approximately 700 tests that were related to impacts from the code change, whereas GitHub Actions and the other CI tool ran the entire suite of approximately 16,000 tests.Â

You can see how Test Intelligenceâ€¢ saves a significant amount of time by only running the necessary tests and running them in parallel. With Test Intelligenceâ€¢, Harness CI employs a unique approach to accelerating build times without sacrificing quality.Â

Time and Cost Savings with Harness CI

Gone are the days of âit just worksâ being good enough for CI in a market that demands that companies go faster to remain competitive.

We've talked about the recent improvements that enhance speed, but there are also cost savings when it comes to building faster. With Hosted Builds, companies don't just alleviate the management burden of hosting infrastructure, they can also do more for less. Faster builds mean less build minutes consumed and a smaller footprint for resource utilization.Â

Combining Cache Intelligence and Test Intelligenceâ€¢ with Hosted Builds ultimately saves businesses 200-300% annually on infrastructure costs. This savings is shown in the chart below using a Harness customer's real usage data calculated to indicate how much time and money they would spend with Harness and each of the other CI vendors tested.Â

Testing Methodology

Here is the methodology we used to determine up to 300% cost savings based on real customer data.

Our customer has 90 software engineers who build 19,524 times a month using Harness CI. In keeping that pace, they build 234,288 times a year. Currently, the average time it takes to build is 15 minutes. We estimated that the average number of builds per developer per week was 11.Â All estimates assume builds are run on Linux. Based on this customer, this comes to a total of 292,860 build minutes a month and 3,514,320 build minutes a year.Â

Next, we compared these figures against two CI vendors. GitHub Actions costs 0.008 per build minute, and CI Vendor 2 costs 0.0120 per build minute. Harness CI is competitively priced relative to GitHub Actions and priced lower than CI Vendor 2.Â

With a very conservative estimate of two times faster than both vendors, Harness CI would cost \$1,171 a month and \$14,057 a year. Respectively, CI Vendor 2 cost \$2,343 a month and \$28,115 a year. GitHub Actions cost \$3,514 a month and \$42,172 a year. That

yielded 200% annual savings over CI Vendor 2 and 300% savings over GitHub Actions.

See For Yourself How Fast Harness CI Is

On a personal note, I am extremely excited to share these updates. The last few years have left much to be desired with CI innovation, but today, with the ability to run builds up to four times faster, the game changes completely. It was my dream to make a CI system that would be simple, open, fast, secure and now this dream has become a reality. I invite you to try this sample to reproduce the results and see for yourself.Â

Ready to start building with the fastest CI platform on the planet? Get started for free with Harness CI.

â

Bradley Rydzewski is Senior Director of Product at Harness and founder of Drone.Â

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/software-development-life-cycle>

Understanding the Phases of the Software Development Life Cycle

While the technologies, methods, and perspectives about building high-performance and scalable software services have changed, the responsibilities and actions have not. The Software Development Life Cycle (SDLC) is a series of important phases defined for teams producing and delivering high-quality software. This blog post will discuss the SDLC and its stages in greater detail.

What is the Software Development Life Cycle?

The Software Development Life Cycle refers to the phases of work involved in producing software applications. Each phase corresponds to a role or responsibility that contributors to the software must understand, manage, and optimize to deliver their software services with speed and performance. These stages of work include:

- Requirements Gathering,Â

- Software Design,Â
- Software Development,Â
- Test and Integration,
- Deployment,Â
- Operationalization and Maintenance.

Letâs discuss each stage of work in greater detail.Â

The Phases of the SDLC

Requirements Gathering

In this stage of work, the team identifies, gathers, and defines current problems, requirements, requests, and customer expectations related to the software application or service.

Some activities related to the requirements gathering phase can involve creating software specifications, creating a detailed plan, documentation, issue tracking, and project or product planning, including allocating the correct resources.

Defining software or product requirements gives teams the foresight and context needed to deliver and produce their software solutions.

Software Design

In this design phase of work, the team makes software design decisions regarding the architecture and make of the software solution. This can involve creating design documents, coding guidelines, and discussing the tools, practices, runtimes, or frameworks that will help the team meet the software requirement specification and goals defined in the requirements gathering phase.Â

Software Development

In this stage of work, teams build the software solutions based on the design decisions made. Here, teams meet the goals and outcomes set during the software requirements gathering phase by implementing the solution.

The development process may involve teams of people, new technologies, and unexpected challenges; however, the development teams typically work with tech leads and product or project managers to unblock the process, make decisions, or provide support. This stage of work ends once teams have packaged and built their code.

Test and Integration

In this phase of work, a software implementation is packaged and tested to assure quality. Testing or quality assurance ensures the solutions implemented pass the standard for quality and performance. This can involve unit testing, performing integration and end-to-end tests, verification/validation, and reporting or identifying bugs or defects in the software solution.

Deployment

In this stage of work, the software is deployed into a production environment. The work gathered, designed, developed, and tested is shared with the consumers and users of the software service. This process involves provisioning infrastructure within an on-premise or cloud provider and defining a software deployment strategy for delivering the changes to a customer.

If you want to learn more, we break down deployment strategies and discuss when and why to use each type.

Operationalization and MaintenanceÂ

In this stage of work, the software is operationalized to ensure there are no issues or incidents related to the deployment. This stage of work can involve reviewing, understanding, and monitoring network settings, infrastructure configurations, and performance of application services in production. This process can involve incident resolution or management in the course of any issues or changes made to impact a customer or user base.Â

Why Does the SDLC Matter for Software Delivery?

Now that we know more about the work associated with each part of the SDLC, we can discuss why it matters and how it applies to how we deliver software today. For many organizations, a challenge is delivering better software faster. Understanding the SDLC allows teams to understand what it takes to deliver features or code changes to customers.Â

The reality for many developers is the need to wait months or years to see code changes make it out to users, coupled with a lack of visibility, communication, and collaboration during the process. Organizations and teams that have the capability to deploy on-demand and in a self-service fashion empower their teams to continue doing their best work.

Our Continuous Delivery 2020 Insights report found that engineering teams spend on average \$109,000 annually to deploy and deliver their software applications. Production deployment efforts result, on average, to 25 hours of engineering effort.Â

The SDLC offers perspective into the distinct work phases needed to produce software. Understanding this work allows teams to avoid

the delivery issues by creating and owning checks and balances early on in our development and delivery life cycle.

This is also about incorporating feedback and insights during the software development process to continuously deliver value in a repeatable, quick, and sustained fashion.Â

Understanding the SDLC allows teams to also improve their DevOps performance, which can be measured through DORA metrics.

Examples of a Software Development Life Cycle Model

Computer scientists, software development practitioners, and leaders have always aimed to deliver better software faster. Over time, several models (such as waterfall, spiral, Agile) emerged to describe and represent the SDLC processes and manage the level of development complexity as demand, tools, processes, and mindsets changed.Â Â

Waterfall Model

Perhaps one of the earliest models used to represent the process for delivering software is the waterfall model, developed in 1956. In this model, a chain of linear sequential phases represents the activities for delivering software.

Each phase depends on the delivery and execution of the previous phase, where each phase contained a set of tasks. This model originated from the manufacturing and construction industries and was adopted for knowledge or project-based creative work.Â

The drawback of this model is its dependencies. Since progress flows in one direction, there was little room to adjust to newly-discovered constraints, requirements, and problems once design decisions were made and implementation began.

Delivering all the software would also lead to increased costs as changes in requirements would lead to major redesigns, redevelopment, and retesting. These drawbacks lead to modified waterfall models, such as the Sashimi (Waterfall with Overlapping Phases), Waterfall with Subprojects, and Waterfall with Risk Reduction.

Iterative and Incremental Model

In response to the perceived problems with the waterfall model, organizations such as the United States Department of Defense released statements, such as the MIL-STD-498, encouraging âIterative and Incremental Development.â This led to the term that would combine both iterative design and incremental development patterns.

In this model, the software is developed and delivered through repeated cycles of smaller portions of work. This model allows for software teams to take advantage of learnings and insights made earlier on in the process from developing and using the software system. Teams at each iteration of work make the necessary design modifications and additional functional capabilities.

NASAâs Project Mercury is an example of the early usage of the Iterative and Incremental Development model. The success of the project later led to further adoption as Project Mercuryâs engineers took to other teams and projects.

Although the origins of the iterative model stem from the software industry, many hardware and embedded software development efforts are now using iterative and incremental techniques (for example, companies such as SpaceX and Rocket Lab).

The Evolution of Process Models

Following the success of Iterative and Incremental software development methods, other software development methods emerged to leverage more project management principles and development practices.

The spiral model is one risk-driven development model that encourages project teams to deliver based on unique project risks, leveraging one or many elements of other delivery methodologies. In the 1990s, the Agile manifesto led to the adoption and popularity of the Agile model and subsequent Agile methodologies.Â

Today, we have the DevOps Life Cycle, representing the SDLC and our goals to continuously deliver software value as a cross-functional team.

How you develop or deliver is up to you. Whatâs important is knowing what is involved in the process. This blog post discussed the software development lifecycle and the SDLC models that emerged to allow us to build and share our software services today. Good luck to everyone who manages or is a part of a project life cycle!Â

Expand Your Knowledge of SDLC Tools

Domain experts, architects, systems developers, engineers, and leaders all have a stake in delivering great software. If youâd like to learn more about delivering software value, we recommend getting more in tune with the state of the Kubernetes ecosystem.

We have created an in-depth eBook that you can download **for free**, which you will learn about Kubernetes architecture, options for running Kubernetes across a host of environments, key open source projects in the Kubernetes ecosystem, adoption patterns of cloud-native infrastructure and tools, and more.

Download your free copy of Kubernetes eBook now.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/what-is-argo-cd>

What is Argo CD?Â

At its very simplest, Argo CD is an open-source GitOps continuous delivery tool. It monitors your cluster and your declaratively-defined infrastructure stored in a Git repository and resolves differences between the two â effectively automating an application deployment.Â

You may have also heard of it as a Kubernetes controller. Kubernetes helped change how infrastructure is managed, but it also added complexity for developers and system administrators by introducing new concepts like kubernetes manifests, kubernetes resources, and other various Kuberentes specific concepts. As the GitOps movement starts to walk back some of that complexity and abstract it under a layer of automation, Argo CD (âCDâ of course being shorthand for âcontinuous deliveryâ) is one of the primary tools that makes this automation possible.Â

Without Argo CD, GitOps remains âClickOps.â Your CI/CD tool may be able to roll out infrastructure changes, but it canât monitor diffs, and thus infrastructure remains mostly manual.

In this article, we explain what Argo CD does, how it works, and an example of how to get started with it.

What Does Argo CD Do?

Argo CD is a declarative, GitOps continuous delivery tool built by the team at the financial software giant Intuit. After moving to the public cloud, having to manage containerization tools and infrastructure interfered with Intuitâs goal to realize the maximum benefits of the cloud. The Intuit team developed Argo CD to achieve maximum release speed and velocity gains. They wanted to apply a pane of automation that would eliminate some of that manual work, so they created the controller â Argo CD â and they open-sourced it.Â

Argo CD is now maintained by the community as a part of the Argo Project, and embedded in lots of GitOps tools that youâre probably familiar with as a delivery tool for Kubernetes native continuous deployment. (Harnessâ own GitOps solution is built on top of it.)Â

Why Argo CD?

As a GitOps continuous delivery tool, Argo CD continuously monitors your running infrastructure (the actual state) to compare it to declaratively-defined code (the desired state or target state) to determine whether they are out of sync, which helps to remediate configuration drift.

Argo CD automatically deploys new configurations and new version code to the target environment. Depending on how you've configured Argo CD to work, it'll either notify you that things are out of sync after a new git commit, or take action. If you've set it up to automatically enforce changes, it'll overwrite the production configuration with what's stored in your immutable, versioned Git repository. The tool is great for complex application rollouts.

Furthermore, Argo CD is essential when your developer team is working in version-controlled environments. By automating lifecycle management and application deployment, Argo CD proactively monitors your application configuration for any potential syncing issues before it reaches your production environment.

Argo CD Core Concepts

To set up and begin using Argo CD and Kubernetes, you'll need a working familiarity with:

- **Containers**, virtual machines, and probably, the containerization tool Docker
- **A container orchestration system** like Kubernetes (though there are alternatives)
- **Continuous deployment and integration tools** like Harness
- **(Possibly) Managing Kubernetes clusters** and manifests in YAML, Helm Charts, or Kustomize

If you're familiar with the items above, you know that to reduce the amount of manual effort needed to provision infrastructure, it's helpful to deploy in containers, orchestrate those deployments, and abstract those deployments into code. That's the core premise of GitOps being declarative and version controlled — moving everything into a Git repo where you can bring the full force of source control and automation to bear.

Argo CD offers the actual monitoring and syncing between the Git-defined infrastructure and your container orchestration tools. It has five primary components:

1. The Argo CD user interface (UI)

Within the web UI, you can create applications, though advanced folks interested in automation will want more control and will do this declaratively in Git. You can also manage connected Git repositories, certificates for access to those repositories, your clusters, and projects directly in the web UI. (Projects let you structure your applications to create useful silos around each team's work.)

One of the more useful components of the Argo CD interface is you can view a visual pipeline of your infrastructure deployments, and your various applications' services and clusters. If an application is in a crash loop, this lets you see all the events, as well as the manifest and configuration values, so you can visualize deployment issues and debug. (It will also hide your secrets.)

2. Both API and command line interface (CLI)

Users need not touch the Argocd login page because you can either manage Argo CD through APIs or create YAML resource definitions (or use Kustomize or Helm Charts to help manage Kubernetes resources). Once set up, developers don't need to understand all the intricacies of the infrastructure they're deploying. They just need the application definitions to include in their pull requests or merge requests.

3. Custom resource definitions (CRD)

Argo CD creates its own namespace within your Kubernetes (or similar) cluster (`kubectl create namespace argocd`). There, it stores the Argo CD CRDs.

4. Repository service

Argo CD caches your Git repo locally and stores application manifest files.

5. Application controller

Once you download the Argo CD application controller onto your repository server, it can invoke hooks defined by software development lifecycle events. (E.g. PreSync, Sync, PostSync.)

To download Argo CD, visit the project's "Getting started" page.

An Example of Deploying Infrastructure Updates Using Argo CD

Once you've installed Argo CD and have configured your certificates and clusters, you can set it to automatically "deploy" infrastructure by enforcing the Git-defined version.

For example, the following becomes possible:

This also works in reverse, and that's the real advantage of a Kubernetes controller. If the Kubernetes cluster resources fall out of sync, Argo CD will detect that, and reconcile the changes by applying what's in Git. If you were simply using a CD tool or Jenkins, the sync issue would go undetected and unremediated.

Things to Consider Before Adopting Argo CD

Argo CD is an intuitive solution to deploy Kubernetes applications. But before you go all in on the open source project, you should be aware of some of its shortcomings. Governance is not a strong suite of ArgoCD. There is limited role based access control and there is virtually no audit trails. If you have a strict deployment process and you need to make sure applications are correctly deployed, GitOps and ArgoCD may not be able to help you enforce stringent rules. Advanced deployment strategies like blue green or canary will be difficult to implement if you're just using ArgoCD. Multi cluster management can also present challenges at scale.

Harness offers ArgoCD-as-a-Service to address these gaps and make ArgoCD scalable for enterprise organizations.Â

Argo CD: Turning âClickOpsâ into GitOps

Argo CD is the controller (among other things) that makes GitOps go. Developed by Intuit and maintained by the community, it's increasingly the standard for reconciling changes between declaratively-defined infrastructure and production clusters.Â

In this article, we've reviewed why Argo CD is important, what it can do, core concepts, and an example of application deployments with it.Â

If you're looking to make good on your investment in CI/CD, and want infrastructure to mostly manage itself, this is a great place to start to support complex application rollouts.

Are you looking to choose Argo CD? Request a demo of Harness GitOps as a Service.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/announcing-speed-enhancements-and-hosted-builds-for-harness-ci>

Announcing Speed Enhancements and Hosted Builds for Harness Continuous Integration

We are excited to announce feature enhancements for the Harness Continuous Integration (CI) module designed to deliver the fastest CI

builds available in the market, along with hosted builds as a fully managed cloud service offering. New Cache Intelligence, enhanced Test Intelligence, and Starter Templates deliver additional time savings for both Harness CI's SaaS and on-premise offerings.

Harness's new CI features demonstrate Harness's commitment to delivering a world-class developer experience for our customers. These new CI features deliver both speed and simplicity – all built on the open source backbone of Drone, renowned for its vibrant community of more than 50,000 active users.

CI Architected for Speed, Simplicity, and Scale

Harness CI hosted builds enable developers to create and run a pipeline in as little as a minute, without having to take on any of the burden of the necessary infrastructure to do so. Removing onerous processes, such as provisioning server build farms and manually turning off test systems in the cloud, enables developers to spend more time writing code and less time dealing with the ingredients necessary to complete their builds quickly.

Harness CI hosted builds are available as a fully managed cloud service offering, with high availability and built-in backup and recovery to ensure business continuity. Enterprise organizations further benefit from CI's ability to define and scale processes, security, and compliance across teams and repositories using Policy as Code and pre-built templates. With these features, it's easier to create and enforce consistent processes across the organization.

Hosted builds are offered for all subscription levels: Free, Team, and Enterprise. In this new pricing model, credits are used to pay for build execution time, depending on the type of target environment used (OS type, machine resources, etc.). New users can take advantage of 2,000 free credits at signup, which equates to approximately 2,000 build minutes on Linux systems.

Efficiencies Driven by Machine Learning

Time spent waiting for builds to run is a common and painful issue impacting developers in software delivery. Valuable developer time would obviously be better spent creating differentiating features for your customers than waiting for a build to complete. Harness CI's proprietary Cache Intelligence reduces pipeline execution time by using machine learning (ML) to automatically cache well-known directories for Java and Node.js to avoid overloading compute processing power, dramatically decreasing build times. With Cache Intelligence, developers can spend less time waiting for builds to finish and more time coding or debugging. Harness leverages intelligence within Harness CI so developers get build speed benefits without effort.

Harness CI's exclusive, ML-powered Test Intelligence feature delivers even more time savings now with the addition of test concurrency, which splits the necessary tests so that they can be run simultaneously. Harness CI Test Intelligence determines which subset of tests are necessary based on code changes while providing visibility into which tests were selected and why. After the required tests are identified, these tests are run concurrently to further accelerate build time without compromising quality. Test Intelligence reduces build cycles by up to 90%, so developers are no longer wasting valuable time waiting around for builds to finish.

Get Started Building and Running Pipelines Faster

As teams have become more geographically distributed and development environments more complex, onboarding new developers can take as long as three to four weeks. Harness CI has made developer onboarding faster with starter templates for most programming languages, including .Net, Java, Go, Node.js, Python, Ruby, PHP, Rust, and more. With language-specific templates, developers don't have to hunt for dependencies or create their own environments which delay getting started. With Starter Templates, developers can start building and running pipelines faster, and spend less time learning how to use new tools and systems. Customers can further speed up the onboarding process with Harness CI's SaaS offering, as hosting on premises can also delay onboarding when having to build and maintain server build farms.

In addition Harness provides an intelligent YAML editor with schema validation and auto-complete recommendations to expedite the configuration experience. The Harness visual pipeline editor also provides a guided experience for building, debugging, and running pipelines quickly. Developers of all levels will be able to reduce toil and spend their time doing what they enjoy doing: writing code.

Lastly, developers can now run their pipelines anywhere as Harness CI has added a Docker runner and expanded operating system support to Mac OS, in addition to Windows and Linux.

Try Out Harness Continuous Integration Today

See for yourself just how fast Harness CI can be. Not only does Harness CI help optimize performance, it's all designed with the developer experience in mind. Get started with a free trial today.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/best-practices-software-delivery>

Modern Software Delivery Best Practices & Software Delivery Management

Marc Andreessen famously said "software is eating the world," in that software is no longer just static websites and a few programs running on your computer. Modern software runs your car, makes healthcare better, and even cleans up after my cats. Yeah, that last one wasn't the future predicted in movies, but here we are. I digress. This article will have two main parts: modern software delivery, and software delivery best practices. Let's dive right in.

Modern Software Delivery

With software as a critical component of daily life, the release cycle must be faster, software quality must be higher, and security (especially data security) can't be an afterthought. The news is full of failures on each of these fronts on a weekly basis, and the businesses that can adapt will thrive in this new world.

These demands on modern software are non-negotiable, and it's forcing a shift in how business is done. Longstanding technical organizations are completely redesigning their software development life cycle, companies where IT was a cost center to the business are investing in software, and hundreds of startup hopefuls are flooding the market looking to bring software up against some of the longest standing challenges in everything from healthcare to space exploration.

That's a long way of saying that everyone needs to improve how they deliver software. Best practices from as little as five years ago are already outdated. Technological shifts, innovation, and market shifts have teams on their toes at all times to just keep up with demand. Movement to the cloud, digital transformation, DevOps, DevSecOps, chaos engineering - That's just a shortlist of what's dominated the attention of development teams in recent years. Let's talk about how we can tackle these challenges and make the world a better place.

Start With Goals

Before you dive into data, or best practices, you need to know what success looks like. Every organization has different ambitions based on the needs of their customer base. At Harness, we speak to hundreds of companies every year. What we've found is that most common goals generally break down into one or more of these four categories. And often, they hit all of them.

Velocity

Delivering software faster is front of mind for almost every organization I talk to. The market is moving fast and they can't wait weeks or months to get changes out into the hands of their customers. Moreover, long delivery cycles mean more time and money is being invested in new features that aren't getting real world feedback.

Governance

Most organizations have audits to deal with, whether it's security audits, or industry-specific regulation. Including considerations for traceability, documentation, and logging in the design of your software delivery process reduces headaches down the road when you

realize compiling all your audit data by hand is an approach that doesn't scale with growth and velocity.Â

Quality

Not every organization is obsessed with speed, is highly regulated, or has a workload that demands optimization in order to scale. But, quality is the one goal that crosses all industries and practices. People remember bad experiences more than they do good ones, so quality has to be baked into every aspect of your software development, not something done right before your end users see your latest feature.

Efficiency

Efficiency often goes hand in hand with speed, but as you add other considerations mentioned above, lack of efficiency can often cut into speed. Processes that get features to market rapidly often involve trade offs, and scaling these processes can amplify the worst effects of that balancing act. These challenges can include manual toil, cost considerations in your cloud spend, maintenance, and more.Â

Software Delivery Best Practices

Take a Data-Driven Approach

Once you know what you're looking to achieve, start looking at ways to quantify that success. There's a whole litany of literature on effective software delivery and metrics, with the choice of this author being Accelerate by Jez Humble, Gene Kim, and Nicole Forsgren. A quick summary of Accelerate: High performing organizations have a lot of things in common, and the data proves it out. As an aside, we also have an article on Accelerate metrics that may be of interest!

A quantitative approach to manufacturing in the real world unlocked higher levels of productivity, and software is no different. While there is a fierce debate of what metric shows how productive a software team is, there is no dispute over measuring this data and working to improve processes.

Use the Agile Approach

The same principles you use to evaluate and improve your software development process should be applied to your software. Manageable, measurable goals that you iterate quickly make you more responsive to your customers. Delivering software is a constant process and feedback is not something that should be reserved for the end.Â

Delivering new features starts with planning deliverables that you can provide feedback on quickly. Spending months building out a product before you solicit feedback leads to waste as requirements are not always met as intended, corner cases don't get surfaced, and technical debt can be created by bad tradeoffs that were unnecessary.

Depending on the data you look at, somewhere around 2/3s of organizations are practicing agile. With that, there's a good chance you already are too. So how can you level this up? For one, there's a whole lot of well-researched articles on this topic.Â

What else? I'm glad you asked. Enter the humble feature flag. Homegrown systems have existed for years, but in recent years it has gone from a wild west hack to a full-blown ecosystem of tools with a user base that spans developers, DevOps teams, and product management. Feature flagging tools allow for you to iterate faster, merge your code in smaller commits, and gather feedback on alpha and beta features in production, without the risks inherent in testing in production.

Ditch the Monolith

Microservices hit peak buzzword a few years ago, but the hype is real. Having distinct services that are loosely coupled allows teams to deliver software faster and with better quality (in fact, I talked about the journey from monolith to microservices on this very blog). In a microservices architecture your service is responsible for fulfilling a role, no more, no less. Other services and the rest of the organization is not concerned with how you achieve the ends they need, just that you fulfill it in a timely manner, correct manner.Â

This approach frees up software development teams from commitments to specific languages, frameworks, deployment strategies, and other patterns that don't fit right for every single service.Â

Source Control Process

Your code is your product. Everything downstream of source control will be affected by decisions you make in your branching process, code reviews, and merge process. The more software developers are touching a given codebase, the likelihood of merge conflicts and other logistical issues goes up.

Review process - Be deliberate in establishing a culture of peer review. Validating changes and providing feedback is not a chore, it's a quality gate. It provides a learning mechanism for newer developers.

Merge frequently - Avoid long lived branches. Merge conflicts and lack of proper integration slow down delivery velocity, and can deter quality. Consider using trunk-based development as a strategy to minimize your lead time to production.

Testing

It goes without saying that robust testing is the cornerstone of delivering quality software. In the current year, it's still not uncommon

that the bulk of testing applications happens in the QA phase of delivery. As a part of the greater shift left mentality, writing and running tests is just as important to software engineering as writing your application source code.Â

Frequent production pushes rely on a rigorous testing regimen. During development, you build out unit tests, integration tests as you pull the different pieces together, and functional tests to validate the end-user experience is as expected.Â

The benefits of automated testing are obvious, though a new challenge has emerged from its adoption at scale. Mainly, it takes up increasing amounts of resources and time. Harness introduced Test Intelligence to address this exact problem.

Finally, testing in production is no longer a third rail. The aforementioned feature flag has changed the paradigm in the level of control when trying out new things. Seeing how a feature works in the wild allows for greater refinement ahead of general release.

Continuous Integration

Continuous Integration processes allow for multiple developers to contribute to one codebase, while maintaining a consistent, well-tested end product. A shortlist of what your build process should include is unit testing, security scans, dependency management, compilation/build, and packaging.

Continuous Deployment / Continuous Delivery / Continuous Verification

Software delivery has long been plagued by manual toil, scripting, and wasted time in getting your code to production. The release process will generally involve a long list of tasks such as provisioning infrastructure, vetting in the test environment, sign-offs/approvals by QA, and ultimately deploying to the production environment. Automated software delivery makes your process repeatable and sustainable while avoiding liabilities come audit time. It's 2022, so it's time to ditch the scripts for delivery. Modern software delivery tools provide extensive tooling, integrations, and options to automate your entire delivery process.Â

To achieve both your speed and quality goals, automated verification of deployments is a must. This means more than just validating your app starts, or that your health endpoint gave you a quick HTTP 200. Verification is tying your metrics into your pipeline and evaluating the health of your app for a period of time to make sure you don't miss adverse events that happen soon after deployment. This is in addition to your normal monitoring regimens, though Harness Continuous Verification provides you with a way to use those existing tools in your pipelines, with AI/ML analysis.

It's worth noting that while it cooperates with CI, CD is a distinct problem being solved and we have spilled much ink on this lovely blog explaining the finer nuances of what is CI and CD.

Security

Just as testing is not the sole responsibility of the QA team, security is the responsibility of everyone. Security begins at developing source code, is tracked throughout the process including artifact management, and ultimately through each and every deployment to environments.

Basic measures any organization can take include requiring security scans in CI/CD pipelines, code analysis, managing SBOMs (software bill of materials), and well designed standards enforced throughout the release process.

Shift Left

With the agile approach, feedback and responsiveness are paramount. The idea of shifting left is to move control over every key aspect of the end product into the hands of the software team.Â With security as an example, think about how much more secure your end product is likely to be if teams are made aware of potentially unsafe code, or dependencies while building the software product, instead of well after the fact when they may have moved on to another initiative.

If teams spend a small amount of time upfront to manage security, testing, and infrastructure as code, the business saves resources in the long run with happier customers and better agility to meet business goals.

Efficient Management of Software Development

Tools

The acceleration of software development has resulted in a multitude of great tools to meet the challenges. Open-source projects such as Jenkins paved the way for modern CI/CD solutions, products such as GitLab brought a first class CI experience to version control, the Jira Board is a staple in software delivery, and lest we forget Harness bringing the first modern CD solution to market.Â

Modern software solutions for software development range anywhere from minimum viable product (MVP) solutions to white glove solutions, and full product suites. There is a constant push and pull in arguments over costs, build versus buy, developer experience, and more. The best organizations invest in appropriate tools because every developer working on a tooling problem is not working on a customer problem.

Culture

Software is still a people business, but this can often get lost in the storm of technologies, tools, and trends. Software is written by human beings with families and interests outside of work. To manage a productive team there must be a culture of respect for people and their

time. Moreover, with the rapid expansion in jobs and opportunities, many developers will leave a job than deal with a toxic culture, a bad manager, or poor development tooling/processes. Invest time and resources into career development, team building that isn't just dropping a ping pong table and a keg in the middle of the office, and listening to feedback.

Continuous Improvement

Perhaps the most important point here is that this is a journey. No organization has solved everything, regardless of how many smart and creative people they hire. Measure your success against the industry at large, against your peers, and most importantly against yourself. Your software development process should be better today than it was yesterday.

Closing Thoughts

At Harness, we partner with a diverse array of customers; everyone from the biggest banks in the world to early stage startups. When we partner with these organizations a big part of what we are doing is helping them find the right ways to optimize their development process and bring it in line with their delivery goals. Successful software delivery is one part planning, one part culture, and one part tools.

Do your tools support your software delivery process, or do they slow you down? Come have a chat with us about our software delivery platform, whether you're looking at Continuous Integration, Continuous Delivery, feature flags, or something else.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

[Documentation](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[Case studies](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[We want to hear from you](#)

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

[Sign up for our monthly newsletter](#)

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/legal/subprocessors>

Harness Subprocessors

As of

In order to provide its services, Harness may engage third parties to carry out data-processing activities that involve access to customer data. These organizations, called "subprocessors," are identified below with their locations and the types of services they provide to Harness. Harness performs due diligence on the technical and organizational security measures of all subprocessors, and requires each to

commit to obligations regarding their security controls and applicable regulations for the protection of personal data.

Name	Description of processing	Location
Amazon Web Services, Inc.	Hosting Services	USA
Microsoft Azure	Hosting Services	USA
Google LLC	Hosting Services/ Analytics	USA
Salesforce, Inc.	Customer Relationship Management	USA
Zendesk, Inc.	Customer Service Management	USA
Marketo, Inc.	Customer Engagement	USA
MongoDB, Inc.	Database Management	USA
Amplitude, Inc.	Analytics	USA
Segment.io, Inc.	Analytics	USA
SendGrid, Inc.	Email Notification Service	USA
Stripe, Inc.	Payment Processor	USA

Source URL: <https://www.harness.io/blog/harness-policy-as-code>

Introducing Harness Policy as Code, Powered by OPA

We're excited to announce Harness Policy as Code, powered by Open Policy Agent (OPA), a centralized policy management and rules service that empowers enterprises to centrally define and monitor policies that are enforced across all delivery pipelines and processes. Harness Policy as Code helps organizations create and enforce policies on deployments, infrastructure, and more, providing developer velocity without sacrificing compliance and standards.

Harness Policy as Code is based on OPA, an easy-to-use, extensible solution for creating and enforcing policies across the entire stack. OPA is an open source project accepted by the Cloud Native Computing Foundation (CNCF) with wide adoption across numerous software delivery use cases. Policies are written as declarative code, so they are easy to understand and modify—from simple to complex use cases.

Harness Policy as Code integrates with CI, CD, and Feature Flags enforcing automated approvals, denials, and other advanced pipeline functionality. Check out our technical documentation to learn more.

Why We Need Policy Management in Software Delivery

As DevOps is adopted within an enterprise, typically one team creates and maintains software delivery and processes. That team has full control and visibility, as they are the "creators" of DevOps processes within the company. As more business units adopt DevOps within the company, that originating team's manual processes can create a bottleneck, which hampers innovation by limiting team autonomy and slowing down software delivery.

Â In an effort to remove the bottleneck and increase velocity, companies can give development teams more autonomy by allowing them to drive their own DevOps processes. That decentralization of process control can lead to more risks for the company.

When governance is decentralized, development teams can miss quality checks or approvals, introduce vulnerabilities, or break compliance. Organizations need to balance autonomy *and* governance, so they can empower teams with the confidence that they are adhering to all compliance standards and security policies â all without slowing down innovation.

Compliance becomes even more critical in regulated industries, such as financial services and healthcare—not only with enterprise standards, but with third-party regulations, like SOC2, PCI, and FedRamp. It is imperative that all software delivery pipelines meet compliance standards with full auditability; otherwise, the organization is at risk of failed audits, heavy fines, and reputational damage.Â

Centralized management and governance of policies across DevOps processes allow enterprises to define standards for the entire organization while enforcing compliance with regulations. Policies enable individual teams to have autonomy over their processes with oversight and guardrails in place to prevent them from straying from standards, ensuring secure and compliant software delivery.Â

Harness Policy as Code Features

Harness Policy as Code is a centralized policy management and rules service that leverages OPA to meet compliance requirements across software delivery. HPE enables organizations to centrally define and monitor policies that are enforced across all delivery pipelines and processes.Â

Policy as Code features for writing and enforcing policies include:

- A Policy Editor that enables developers to start writing policies-as-code quickly. With a library of policies to start from and a testing terminal, developers can try out policies on real inputs during development before enabling them.
- Policies that are configured to be automatically enforced on Harness processes (e.g. on Pipeline Run, on Feature Flag save).
- The ability to set severity, so a policy violation can issue a warning or throw an error to stop processes from continuing.
- An audit trail that can maintain a full history of policy evaluations with detailed outputs for audit and compliance.

With the release of Policy as Code, policies can now be enforced on CI and CD pipelines and Feature Flags.

Pipeline policies govern the requirements of delivery pipelines, and they can be automatically enforced when the pipeline is saved or triggered, or even in the middle of pipeline execution. Policies can enforce specific pipeline configuration, advanced access control use cases, runtime validation, and more. Here are some examples of what the Policy as Code can do:

- Require an approval step before deployment to production.
- Forbid use of Shell scripts in the pipeline.
- Only allow deployment to approved namespace.
- Only allow deployments from approved container registries.
- Validate test step outcome meets minimum threshold before allowing the pipeline to continue.

Policies for Feature Flags are enforced when the flag is updated or toggled on/off, enabling policies for adhering to standards, flag process, and hygiene. This includes:

- Only allowing creation of boolean flags.
- Enforcing flag naming conventions.
- Enforcing when creating a flag the default on and off values must both be false.
- Requiring a Feature Flag be enabled in QA before it can be turned on in Production.

Policy as Code centralizes and standardizes policy management across software delivery, allowing engineering leaders to empower dev teams to own their tools and practices while ensuring that everyone is following company standards for compliance and security. With guardrails in place, security vulnerabilities won't be introduced as development teams are writing their pipelines. Leaders can rest assured that compliance standards are being met, with full auditability of policies and failures, and they can find and report breaches as early as possible with shift-left governance.

Get Started

Check out our platform governance page to learn more about how Harness' modern approach to software delivery governance empowers teams with stable processes that don't slow down delivery, or request your personalized demo today.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

What Is GitOps? Learn About Benefits, Challenges, and More

In this article, we will cover what GitOps is, what benefits an organization gains from GitOps, and how to scale GitOps for the enterprise.

At Harness, we offer **GitOps-as-a-Service**, built on the popular Cloud Native Computing Foundation (CNCF)-incubated open source Argo CD project. Harness GitOps-as-a-Service not only delivers the lightning fast deployments and lightweight operation that developers love, but it does so with enterprise-grade security and governance at scale.Â

If you want to see Harness GitOps-as-a-Service in Action, request a demo.Â

What Is GitOps? How does GitOps work?

GitOps is an approach for companies looking to simplify deployments of cloud-native applications. Whether itâs adding a firewall rule, defining a VPC, or fixing a UI bug, all of it should come from the central plane of source control. Below we detail the principles that make a GitOps workflow enticing for developers and their teams that are looking to deliver software in a faster, more reliable manner.Â

GitOps Principles

The Entire System is Described Declaratively

GitOps focuses on the what instead of the how. Kubernetes is the most popular platform at the moment where all application components are descriptions, not directions on how to accomplish building them out. In the declarative paradigm, a developer describes their desired state, and the system they are interacting with determines when, how, and where to place applications in a way that meets the requirements described.

The key benefits of declarative infrastructure and declarative configuration are they allow software development teams to focus on their application first, not the logistics of deployment and runtimes.Â

The State of the System Lives in Version Control

In GitOps, the Git repository serves as the authority on the desired state of the application. Every change or rollback is funneled through Git pull requests, Git reverts, and actions oriented around the version control system.Â

Automatically Apply Approved Changes

Within GitOps, the process to apply changes is driven entirely from the Git repository with a pull request or merge request. After working on a feature branch, a developer submits a pull request, any applicable GitOps Continuous Integration pipelines run, and the change is merged once the required process is fully satisfied. On merge, with the main branch as the source of truth, the deployment process is run by the GitOps operator in the deployed infrastructure, with no additional steps required post-merge.

Drift Consolidation

Much of the interest around GitOps has been on the push-based side of development. As a change is made, it is applied to the cluster, and regardless of the mechanics of how this is accomplished, it centers around changes being deployed by developers. The other half of GitOps is the pull action, where an application is self-healing, correcting itself to align with the desired state.Â

Kubernetes-based infrastructure already restarts failed application instances (pods) and aligns with a desired state. GitOps extends this principle to every aspect of the application. When a misalignment is spotted from the application side in configuration, images deployed, etc., the GitOps operator treats source control as the final authority and aligns the running state to the state in Git.

Why GitOps?

Life Before GitOps

In the not too distant past, the author of this article owned deployments for an ecommerce application. The process involved a deployment pipeline, but also many manual processes, and requests to various internal service providers. Every six weeks, new changes were deployed, concluding a process that spanned months when you include the software development process. Something as simple as a few application code changes, or modifications to configuration files, were a large production. At the first sign of trouble, it was a requirement to roll back and conduct another six week test cycle.Â

That six-week run-up consisted of deploying, testing, and retesting in each pre-production environment. If a change required firewall updates, additional server resources, or anything outside of the scope of deploying the codebase, that required a minimum two week leadup for each environmentâoften longer. No change could be initiated for production prior to testing in every lower environment, and infrastructure changes across multiple environments could not be completed in a single six-week cycle.Â

This isnât an unusual story; itâs actually very common. Companies are implementing GitOps because of all these problems. The pace of change slows to a crawl because of fragile, unrepeatable, non-scalable approaches to software development and deployments. The benefits can be seen in the GitOps model below.Â

Benefits of the GitOps Approach

Single Source of Truth

Anyone who has had to onboard into an engineering organization is likely very familiar with the headache of dealing with multiple authoritative systems. One system for firewalls, another system for DNS, a few more systems for provisioning compute infrastructure if you're on-prem, as well as in the cloud, a version control system, and finally, a CI/CD pipeline tool. Unless your organization has an exceptional cultural commitment to documenting everything, it's as much of a full time job to know where to find the answer as it is to parse it.

One GitOps definition is that GitOps allows developers to have their version control system as the authoritative source to look for answers about every configuration they care about. The benefits of this starts with streamlining onboarding, providing an enhanced developer experience, all the way up to avoiding the dreaded context switching that cuts into productive time.

Version Control

The changes made to every aspect of the application live in one place, and are versioned as each change is proposed by pull requests and implemented. With everything in code, any change can be reviewed and there is a full paper trail, and the ability to roll back changes.

Democratizing Infrastructure Management

Historically, every system has a different set of gatekeepers. Network teams, security teams, ops teams, and more. In the pre-GitOps world, achieving Continuous Deployment automation was at the mercy of at least a half dozen different groups and whatever red tape they put up. Any update required a long request process to the owner of a given system, with carefully documented changes.

Developers provisioning cloud resources was the first major deviation from how companies manage infrastructure. Then came infrastructure as code, and ultimately, the GitOps pipeline, to close the loop from idea to implementation.

Standardizing/Ease of Use/Simplifying

Part of the reason for change management and gatekeeping has been to assure that what ends up in a production environment maps to lower environments. With many layers of applications, configurations, and infrastructure, standing up a new environment has often been a combination of checking many sources of truth, change tickets, and ultimately trial and error.

With all infrastructure, configuration, and code defined in a Git repository (or VC of choice), the ability to build out a new environment or port configurations across environments isn't a multi-sprint project.

Velocity

Organizations are increasingly standardizing on metrics to measure the effectiveness of software delivery, with the DORA metrics being the most popular. A key metric measured is lead time, or in practical terms, how long it takes from a code commit to that change arriving in production. The GitOps automation approach shortens the Continuous Delivery cycle time to the minimum required to push new changes and features. GitOps continuous delivery processes can reduce the amount of time needed to get new code to production.

Shorter Feedback Loop

Velocity provides two benefits in one. Getting features to your end users faster is valuable in and of itself. Then, add on top the quick feedback on changes development teams get. It's not uncommon that the time from pull request to user feedback is expressed in months. Revisiting, refining, and pushing changes long after a development team has moved onto another project is more time-consuming and less efficient, especially when there are multiple iterations.

Harness GitOps-as-a-Service

At Harness, we've witnessed a large number of challenges as we've worked with our customers to implement GitOps principles at scale.

Promoting Releases Across Environments

Describing the desired state between test, QA, and production environments becomes increasingly difficult. As of this writing, practitioners are split among a few possible solutions, including using one branch per environment.

This introduces a litany of potential issues, not the least of which is code not being properly merged into all branches in the correct order. Go ahead and do the math on how many repositories you end up having across many environments and applications in that operating model.

Without a clear solution for this problem organizations either won't adopt, will adopt for select environments, or will disparate practices that become a form of technical debt.

Harness chose Argo CD for GitOps adoption as it provided the ability to use an app of apps pattern, resolving at least a portion of handling this issue at scale to manage infrastructure. When there's no longer a single Kubernetes cluster to manage but instead an

increasing amount of them as more teams implement GitOps workflows â issues scaling across an organization intensify. With a parent app in play, cascading changes across multiple applications becomes less of a barrier to entry and assists with cluster management.

GitOps-as-a-Service further solves this issue by offering pull request pipelines. The addition of pull request pipelines makes it easy to propagate changes across multiple services and environments without having to individually manage each deployment by adding a layer of pipeline orchestration on top of standard GitOps deployments.

Auditing

With GitOps, the Git log provides a definitive record of all changes. However, as an audit trail, it can become a burden to search for changes and tie them directly to business or regulatory implications. Within Git, all commits are recorded, but the implication of each commit takes time and effort to unearth â especially for non-technical audiences. An audit trail only goes so far without additional functionality to ease sorting through large numbers of pull requests or merge requests.

GitOps with Harness provides a full audit log of all the deployments and changes, tracked to allow operational and business level analysis, without requiring digging through Git commits. Searching through Git pull requests and files down to the code level is still an option on the table, but it is no longer the required starting point.

GitOps Covers One Part of the SDLC

A well understood limitation of the GitOps approach by itself is, there are many portions of the software development life cycle not covered, as itâs not a one-size-fits-all solution or magic bullet. Compiling code, running unit testing, integration testing, security scanning, and more require more tooling and processes to support the Continuous Deployment model of GitOps.

The Harness platform provides built-in GitOps as part of a holistic solution covering everything in the Continuous Integration and Continuous Deployment life cycle. A distinction is necessary here as GitOps follows an idea of pushing everything live immediately. Â In addition to GitOps capabilities, Harness provides a full delivery pipeline around your process including approvals, controlled feature releases, canary/blue/green rollout strategies, and Continuous Verification.

Scaling Up

With many applications and environments comes a set of challenges with scale. With the number of Git repositories skyrocketing, it quickly becomes hard to track environments and configurations. Centralizing versus true autonomy quickly becomes a real problem in that applying company-wide changes or policy becomes a burden when giving teams autonomy, contrasted with the tradeoffs of centralizing configurations and creating a large bottleneck. With the race for talent on and the necessity of a developer-centric experience, setting up GitOps at scale can become a liability.

With GitOps as a very developer-centric, decentralized approach to software delivery, the burden on DevOps teams is to provide the necessary tools without everything turning into the wild west. As with the benefits of a platform providing for other parts of the SDLC, Harness also provides a centralized pane of glass for managing GitOps while providing the needed autonomy to development teams.Â

Several key challenges the platform looks to solve are providing project-level spaces, robust role-based access control (RBAC), and Policy as Code powered by OPA. Combining the ability to segment spaces, provide proper access, and finally govern by way of policy described as code allows DevOps teams to fully control day-to-day operations without being overly hands-on or prescriptive in how teams build their software.

Adopting GitOps

Looking to adopt GitOps in your organization and want to learn more? Check out our webinar to dive deeper into GitOps, how to adopt, potential pitfalls, and how to extend GitOps to the next level.

Wherever you are on your GitOps journey, from exploration to seasoned professional, we hope this was helpful for you. If youâd like to see GitOps in action with Harness, come talk to us and weâll share our thoughts on opportunities with GitOps, as well as how weâre tackling the challenges.

Request your demo today!

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/legal/engineering-x-terms-of-use>

Engineering X Terms of Use

As of

In these Terms of Use (hereafter "Agreement" or "Terms"), "we," "us," "our," and "Company" will refer collectively to Harness Inc., located at 55 Stockton St., 8th Floor, San Francisco, CA 94108 and its affiliates. The terms "you," and "your" will refer to you, the user ("User"). By accessing or using the Website, or clicking "I Agree," "Sign Up" or "Get Access" you agree to be bound by these Terms. If you do not agree to be bound by these Terms, you should immediately leave this Website and not return.

When we refer to the "Website" in these Terms, we mean all articles, links, URLs, websites, blogs, information, data, and any other content provided by the Company from <https://www.engineeringx.org>.

â

You may only use the Website if you agree to form a binding contract with the Company and are not a person barred from using the Website under the laws of the applicable jurisdiction. In any case, you must be at least 18 years old to use the Website. If you are accepting these Terms and using the Website on behalf of a company, organization, government, or other legal entity, you represent and warrant that you are authorized to do so and have the authority to bind that entity to these Terms, in which case the words "you" and "your" as used in these Terms shall refer to that entity.

Our Privacy Statement (located at <https://www.harness.io/legal/privacy>) describes how we handle the information you provide to us when you use the Website. You understand that through your use of the Website you consent to the collection and use of this information, including the transfer of this information to the United States, and/or other countries for storage, processing, and use by Harness and its affiliates.

The Website and its content, including but not limited to text, graphics, images, logos, software, information, articles, data, code, blogs, articles, event descriptions, models, or any other materials or information posted via the Website or obtained by you through the Website (collectively, the "Content"), are the property of the Company or its licensors and are protected by intellectual property laws. You are granted a limited, non-exclusive, non-transferable right to access and use the Website for internal purposes in accordance with this Agreement. Unless otherwise agreed by you and the Company under other license terms, you agree not to reproduce, distribute, modify, sell, re-sell, or create derivative works of any Content from the Website without the prior written consent of the Company. Nothing in these Terms gives you any ownership rights in any of the Company's intellectual property including but not limited to the Company's Content.

â

Your use or reliance on any Content or Contributions (as defined below) posted via the Website or obtained by you through the Website is at your own risk. We do not endorse, support, represent or guarantee the completeness, truthfulness, accuracy, or reliability of any third party submitted content or communications posted via the Website or endorse any third party opinions expressed via the Website. You understand that by using the Website, you may be exposed to Contributions that might be offensive, harmful, inaccurate or otherwise inappropriate, or in some cases, postings that have been mislabeled or are otherwise deceptive.

We reserve the right to remove Contributions that violate these Terms, including for example, copyright or trademark violations or other intellectual property misappropriation, impersonation, unlawful conduct, or harassment.Â

If you believe that your Contribution has been copied in a way that constitutes copyright infringement, please report this by contacting our designated copyright agent at:

Harness Inc.
Attn: Copyright Agent
55 Stockton St. 8th Floor
San Francisco, CA 94108
Email: legalnotices@harness.io

â

- Registration: In order to become a member of the Website, you must create a user account ("Account"), which requires you to provide accurate, complete, and up-to-date information.Â You are solely responsible for maintaining the confidentiality of your Account credentials and for all activities that occur under your Account.Â You may close your Account at any time by emailing members@engineeringx.org. You represent and warrant that any information that you provide in connection with your use of the Website is and shall remain true, accurate, and complete, and that you will maintain and update such information regularly.Â You agree that if any information that you provide is or becomes false, inaccurate, obsolete or incomplete, Harness may terminate your use of the Website.Â You agree to be responsible for all action taken using your account, whether authorized by you or not.Â You agree to notify the Company immediately if you suspect your account has been compromised. You are responsible for safeguarding your account, so use a strong password and limit its use to this Account.Â The Company may restrict, suspend, or close your Account on the Website, or revoke membership privileges according to its policy for handling copyright-related takedown requests, or if the Company reasonably believes that you've violated any of these Terms.
- Access and Use: Upon becoming a member, you will have access to certain features and Content on the Website, as determined by the Company based on the level of membership granted. You acknowledge and agree that membership benefits are subject to change, modification, or termination at the Company's sole discretion without notice.
- Membership Tiers:Â Members will be classified into the following tiers and receive the following benefits, which may change over time:
- Use of Member Information and Likeness:Â By becoming a member, you grant the Company the right and license to use your name, contact information, photos, and likeness for the purposes of promotion, marketing, and advertising, including but not limited to online publication on the Website and social media platforms.Â Additionally, once you register your Account and become a member, your name, company and role will automatically be added to our Members page located at <https://engineeringx.org/members>. The Company may also add your picture from LinkedIn or other public websites to your information on the Members page.

â

You agree not to use, and not to encourage or allow any third party to use, the Website in the following prohibited ways:

- Encouraging (i) any illegal, fraudulent, or abusive activities or (ii) materially interfering with the business or activities of the Company.
- Buying, selling, or otherwise trading in user names or other unique identifiers on the Website.
- Sending advertisements, chain letters, or other solicitations through the Website, or use the Website to gather addresses or other personal data for commercial mailing lists or databases.
- Automating access to the Website, or monitoring the Website, such as with a web crawler, browser plug-in or add-on, or other computer program that is not a web browser.Â
- Using the Website to send emails to distribution lists, news groups, or group mail aliases.
- Contacting any Website members for commercial purposes.Â
- Falsely implying that you are affiliated with or endorsed by the Company.
- Hyperlinking to images or other non-hypertext content on the Website on other webpages.
- Removing any marks indicating proprietary ownership from materials you download from the Website.
- Showing any part of the Website on other websites with <iframe>.
- Disabling, avoiding, or circumventing any security mechanism or access restrictions of the Website.
- Straining infrastructure of the Website with an unreasonable volume of requests, or requests designed to impose an unreasonable load on information systems underlying the Website.
- Using the Website, or any files, data, software, or other materials received from the Website, in violation of any applicable license, law or regulation.
- Impersonating others through the Website.
- Encouraging or helping anyone to violate these Terms.
- Using or trying to use another's account on the Website without their specific permission.
- Reverse-engineering the Website in order to find limitations, vulnerabilities, or evade filtering capabilities.
- Launching or facilitating, whether intentionally or unintentionally, a denial of service attack on any of the Website or any other conduct that materially and adversely impacts the availability, reliability, or stability of the Website.
- Transmitting any material, data, or content that contains viruses, Trojan horses, spyware, worms or any other malicious, harmful, or deleterious programs.
- Violating or facilitating the violation of any applicable laws or regulations of any applicable jurisdiction.

- Using the Website to transmit any material or content that is, facilitates, or encourages libelous, defamatory, discriminatory, or otherwise malicious or harmful speech or acts to any person or entity, including but not limited to hate speech, and any other material or content that Company reasonably believes degrades, intimidates, incites violence against, or encourages prejudicial action against anyone based on age, gender, race, ethnicity, national origin, religion, sexual orientation, disability, geographic location or other protected category.
- Submitting content to the Website that violates the law, infringes anyone's intellectual property rights, violates anyone's privacy, or breaches agreements you have with others.
- Submitting content to the Website as a mere placeholder, to hold a particular address, user name, or other unique identifier; or
- Use the Website to disclose information that you don't have the right to disclose, like others' confidential or personal information.

â

The Company may investigate and prosecute violations of these Terms to the fullest legal extent. The Company may notify and cooperate with law enforcement authorities in prosecuting violations of the law and these Terms. The Company reserves the right to change, redact, and delete content on the Website for any reason. If you believe someone has submitted content to the Website in violation of these Terms, please contact us immediately.

You are responsible for your use of the Website and for any Contributions (defined below) you provide, including compliance with applicable laws, rules, and regulations. You should only provide Contributions that you are comfortable sharing with others.

Unless otherwise agreed to by you and Harness in writing or under separate terms, by uploading, e-mailing, posting, submitting, publishing or otherwise transmitting information, data, models, modifications, software, event types, tags, comments, code, suggestions, or other materials to the Website or Harness (each a "Contribution"), you hereby acknowledge that such Contribution is non-confidential and automatically grant (or warrant that the owner of such rights has expressly granted) to Harness a perpetual, irrevocable, world-wide, non-exclusive, sublicensable, fully paid-up and royalty-free license to use, make, have made, copy, distribute, perform, display (whether publicly or otherwise), modify, adapt, publish, and transmit such Contributions in any form, medium, or technology now known or later developed, and to grant to others rights to do any of the foregoing. Such use by Company, is made with no compensation paid to you with respect to the Contributions that you submit, post, transmit or otherwise make available through the Website as you agree that using the Website is sufficient compensation for any Contributions and grant of rights.Â

In addition, you represent and warrant that all so-called moral rights in the Contributions have been waived. For each Contribution, you represent and warrant that you have all rights necessary for you to grant the licenses granted in this Section, and that such Contribution, and your provision thereof to and through the Website, complies with all applicable laws, rules and regulations. Harness cannot and will not be liable for any loss or damage arising from your failure to comply with this Section. Harness will pre-screen, review, and approve Contributions, and Harness reserves the right to refuse or delete any Contributions in its discretion. You acknowledge and agree that Harness reserves the right (but has no obligation) to do one or more of the following in its discretion, without notice or attribution to you:

- (i) monitor and approve Contributions as well as your access to the Website;
- (ii) alter, remove, or refuse to post or allow to be posted any Contribution; and/or
- (iii) disclose any Contributions, and the circumstances surrounding their transmission, to any third party in order to operate the Website, in order to protect Harness, its suppliers or licensees and their respective employees, officers, directors, shareholders, affiliates, agents, representatives, and the Website's users and visitors; to comply with legal obligations or governmental requests; to enforce these Terms; or for any other reason or purpose.

Harness disclaims any responsibility for the Contributions displayed on its Website and assumes no responsibility for the timeliness, deletion, mis-delivery or failure to store any Contributions or other user information or personalization settings.

Between you and the Company, you remain solely responsible for the Contributions you submit to the Website. You agree not to wrongly imply that Contributions you submit to the Website are sponsored or approved by the Company. Additionally, these Terms do not obligate the Company to store, maintain, or provide copies of the Contributions you submit.

When Contributions you submit are removed from the Website, whether by you or by the Company, the Company's right to use the Contribution ends when the last copy of such Contribution disappears from the Company's backups, caches, and other systems. Other licenses or rights you apply to the Contribution you submit, such as Creative Commons or other licenses, may continue after your Contribution is removed. Those licenses may give others, or the Company itself, the right to subsequently share your Contribution through the Website.

As a registered member of this Website, you will receive access to the Engineering Excellence Model (the "Model"), a maturity model that codifies engineering best practices, which is licensed to you under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License ("License"). The License can be found here. Modifications to the Model can be submitted to members@engineeringx.org for consideration. By submitting modifications of the Model to the Company, you grant the Company and any other user a non-exclusive, worldwide, royalty-free license to use, modify and distribute your modifications under the terms of the License. Any approved modifications by the Company will be integrated into the Model.Â

Users are allowed to use, modify, and distribute the Model, provided users publish those modifications under the same License, and provide attribution to the Company. This ensures that improvements and enhancements to the Model benefit the entire community of users and contributors.Â

Additionally, while using the Model, you must comply with all applicable laws and regulations, and these Terms. Any use of the Model for illegal or unethical purposes, or in violation of these Terms is strictly prohibited. Failure to comply with this requirement may result in revocation of your access to the website, membership, and services.

You acknowledge that the Model is provided *as is* without warranties of any kind, and the Company and contributors are not liable for any damages or consequences resulting from the use of the Model.

â

You agree to hold harmless, defend, and indemnify the Company from all legal claims related to your Contributions, your breach of these Terms, your gross negligence or wilful misconduct, or breach of these Terms by others using your account. The Company agrees to notify you of any legal claims for which you might have to indemnify the Company as soon as possible. If the Company fails to notify you of a legal claim promptly, you won't have to indemnify the Company for damages that you could have defended against or mitigated with prompt notice. You agree to allow the Company to control the investigation, defense, and settlement of legal claims for which you're required to indemnify the Company, and to cooperate with those efforts. The Company agrees not to agree to any settlement that admits fault for you or imposes obligations on you without your prior agreement.

Your access to and use of the Website or any content are at your own risk. You understand and agree that the Website is provided to you on an *AS IS* and *AVAILABLE* basis. COMPANY DISCLAIMS ALL WARRANTIES AND CONDITIONS, WHETHER EXPRESS OR IMPLIED, OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. Company makes no warranty or representation and disclaims all responsibility and liability for: (i) the completeness, accuracy, availability, timeliness, security or reliability of the Website, the Model or any Content; (ii) any harm to your computer system, loss of data, or other harm that results from your access to or use of the Website or any of the content; (iii) the deletion of, or the failure to store or to transmit, any content and other communications maintained by the Website; and (iv) whether the Website will meet your requirements or be available on an uninterrupted, secure, or error-free basis. No advice or information, whether oral or written, obtained from the Company through the Website, will create any warranty or representation not expressly made herein.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THE COMPANY SHALL NOT BE LIABLE FOR ANY INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL OR PUNITIVE DAMAGES, OR ANY LOSS OF PROFITS OR REVENUES, WHETHER INCURRED DIRECTLY OR INDIRECTLY, OR ANY LOSS OF DATA, USE, GOODWILL, OR OTHER INTANGIBLE LOSSES, RESULTING FROM (i) YOUR ACCESS TO OR USE OF OR INABILITY TO ACCESS OR USE THE WEBSITE; (ii) ANY CONDUCT OR CONTENT OF ANY THIRD PARTY ON THE WEBSITE, INCLUDING WITHOUT LIMITATION, ANY DEFAMATORY, OFFENSIVE OR ILLEGAL CONDUCT OF OTHER USERS OR THIRD PARTIES; (iii) ANY CONTENT OBTAINED FROM THE WEBSITE; OR (iv) UNAUTHORIZED ACCESS, USE OR ALTERATION OF YOUR TRANSMISSIONS OR CONTENT. IN NO EVENT SHALL THE AGGREGATE LIABILITY OF THE COMPANY EXCEED THE GREATER OF ONE HUNDRED U.S. DOLLARS (U.S. \$100.00) OR THE AMOUNT YOU PAID COMPANY, IF ANY, IN THE PAST SIX MONTHS FOR ACCESS TO THE WEBSITE GIVING RISE TO THE CLAIM. THE LIMITATIONS OF THIS SUBSECTION SHALL APPLY TO ANY THEORY OF LIABILITY, WHETHER BASED ON WARRANTY, CONTRACT, STATUTE, TORT (INCLUDING NEGLIGENCE) OR OTHERWISE, AND WHETHER OR NOT THE COMPANY HAVE BEEN INFORMED OF THE POSSIBILITY OF ANY SUCH DAMAGE, AND EVEN IF A REMEDY SET FORTH HEREIN IS FOUND TO HAVE FAILED OF ITS ESSENTIAL PURPOSE.

The Company welcomes your Feedback and suggestions relating to the Website. See the Contact section below for ways to get in touch with us.

By providing Feedback, you acknowledge and agree that any questions, comments, suggestions, ideas, feedback or other information about the Website or content (*Feedback*) provided by you is non-confidential and shall become the sole property of the Company. The Company shall have exclusive rights to Feedback, including all intellectual property rights, and shall be entitled to the unrestricted use and dissemination of Feedback for any purpose, commercial or otherwise, without notice, acknowledgment or compensation to you.

Either you or the Company may terminate these Terms at any time. When our Agreement ends, your permission to use the Website and your membership also ends.

You may end your Agreement with Company at any time by deactivating your account(s) (if available) and discontinuing your use of the Website.

We may suspend or terminate your account or cease providing you with all or part of the Website at any time for any or no reason, including, but not limited to, if we reasonably believe: (i) you have violated these Terms, (ii) you create risk or possible legal exposure for us; (iii) your account should be removed due to unlawful conduct, (iv) your account should be removed due to prolonged inactivity; or (v) our provision of the Website to you is no longer commercially viable. We will make reasonable efforts to notify you by the email address associated with your account or the next time you attempt to access your account, depending on the circumstances. In all such cases, the Terms shall terminate, including, without limitation, your license to use the Website. For the avoidance of doubt, these Terms survive the deactivation or termination of your account.

Upon termination of these Terms, the terms of this Section 14, and the terms of the following Sections will survive (i.e. still apply): Section 2 (Privacy), Section 3 (Ownership), 4 (Content on the Website), Section 6 (Acceptable Use), Section 8 (Contributions),

Section 9 (Engineering Excellence Model, Modifications, and Licensing), Section 10 (Indemnification), Section 11 (Disclaimer), Section 12 (Limits on Liability), and Section 15 (General).^A A

These Terms will be governed by and interpreted according to the laws of the State of California without regard to conflicts of laws and principles that would cause the laws of another jurisdiction to apply. Any legal suit, action or proceeding arising out of or related to these Terms or the Website shall be instituted in either the state or federal courts of San Francisco, California, and we each consent to the personal jurisdiction of these courts.

In the event that any provision of these Terms is held to be invalid or unenforceable, then that provision will be limited or eliminated to the minimum extent necessary, and the remaining provisions of these Terms will remain in full force and effect. Company's failure to enforce any right or provision of these Terms will not be deemed a waiver of such right or provision.

You may not assign your Agreement with the Company. The Company may assign your Agreement to any affiliate or successor of the Company. Any attempted assignment against these terms has no legal effect.^A These Terms embody all the terms of agreement between you and the Company about use of the Website and supersede and replace all prior or contemporaneous communications, whether oral or written, between the parties.^A

For legal issues, you may notify the Company and send questions to the Company by sending an email to legalnotices@harness.io.^A For technical questions, feedback, or comments please send an email to members@engineeringx.org.^A

The Company may notify you under these Terms using the e-mail address you provide for your account on the Website, or by posting a message to the homepage of the Website or your account page.

We reserve the right to modify or update these Terms at any time, in our sole discretion. We will try to notify you of material revisions, for example via a Website notification or an email to the email associated with your account. Continued access or use of the Website after such modifications or after such notice constitute your acceptance of the updated Terms. If you do not agree to any modifications or updates, you should stop using the Website.

â

Effective: September 20, 2023

Source URL: https://www.harness.io/company/?45c010ef_page=2

Get to know Harness

Harness aims to enable every software engineering team in the world to deliver code reliably, efficiently and quickly to their users.

Our Mission

At the core of our mission lies a deep commitment to empowering software engineering teams worldwide. We are dedicated to providing the tools, knowledge, and resources necessary for these teams to excel in their work. Our ultimate goal is to revolutionize the way software is developed, ensuring that every team, regardless of their size or location, can deliver code with utmost reliability, efficiency, and speed.

Harness's mission is to enable the 30 million software developers in the world to deliver code to their users quickly, reliably and efficiently.

Meet our Executive Team

Jyoti Bansal is a serial entrepreneur and technology visionary who believes passionately in software's ability to change the world for the better. He co-founded Harness in 2017 to automate and simplify all software delivery processes, and serves as CEO. In 2018, he co-founded Traceable, the leading API security platform, and venture capital firm Unusual Ventures. Unusual Ventures is reinventing the VC engagement model by providing entrepreneurs with an unprecedented level of services. Unusual closed its third fund in 2022 and currently has over \$1B under management.

Carlos Delatorre is the Chief Revenue Officer (CRO) at Harness, overseeing the sales and go-to-market (GTM) functions. With a proven track record of enterprise sales leadership, he has consistently built high-performance teams that deliver strong results. Prior to Harness, Carlos was head of Sales at several companies, including Navan and MongoDB.

Currently, Carlos is a board member at Yalo and an investor/advisor to companies including Modern Treasury, Vartana, Exafunction, Starburst, and Outreach. He earned his undergraduate degrees from the University of Miami and Troy State University, and holds an MBA from Troy State University.

John Bonney oversees the Harness company's finance and accounting function. Prior to Harness, he oversaw rapid growth as the CFO at Mapbox as well as FinancialForce, the latter quadrupling revenues to over \$100M during his tenure. Prior to that, he was the Division CFO of the Cloud Business unit at SAP where he oversaw over \$1 billion of Cloud revenues across multiple business units, products, and geographies. Bonney joined SAP from Ariba as a senior finance executive who helped drive rapid expansion and ultimately the sale of Ariba to SAP for \$4.3 billion.

Previously, Luan has served as an advisor for companies like PlanGrid, Headhuntr.io, and 8A8 Inc. He also served as a Guest Lecturer at University of California, Berkeley for the "Organization and Management" course with the International Diploma Programs. Luan holds a degree in International Relations from the University of California, Davis. He has written articles on the "War for Talent"; and has been featured in several publications including Forbes, The Wall Street Journal, The Guardian, and San Francisco Business Times, specifically on the art and science behind recruiting.

â

Sri is responsible for leading the company's global engineering organization, including its R&D teams in United States, Bangalore (India), Latin America, and Europe. Sri previously spent five years scaling Zoom from less than 12 million ARR in the early days to post IPO growth, ultimately leading a team of more than 600 people, launching the company's marketplace, and significantly expanding its ecosystem. Sri has also held key engineering leadership roles at Saba, Plantronics, and Cisco.

Gleb Brichko brings over 20 years of B2B marketing experience across software, security, and infrastructure. Most recently, Gleb built and led the global growth and demand marketing team at Rubrik, where he was responsible for leading, planning and executing the company's go-to-market strategy, brand presence, demand generation campaigns and programs, digital and web, partner marketing, field marketing, customer advocacy, strategic events, content, and operations. His prior roles include marketing leadership positions at Nutanix, Fortinet, ForeScout, and Blue Coat Systems. Gleb holds a bachelor's degree in mass communications from UC Berkeley.

Parmeet Chaddha leads post-sales customer success and services function at Harness. With multiple patents to his name, Parmeet brings 30+ years of general management experience in engineering, business development and customer success. Prior to Harness, he led customer success at Securiti.ai. Previously, he held senior executive roles at Google Cloud, Nutanix, EMC, IBM and Oracle. He has been honored as "Top 50 Technology Executives" by InfoWorld and the "Leading Data Consultants for North America, 2022" by the CDO Magazine. Parmeet holds a B.S. and M.S. from Massachusetts Institute of Technology.

Investors

Harness was spun out of BIG Labs, a startup studio designed to solve hard technology problems and build enduring companies. Harness has raised \$425M of venture capital from top-tier investors.

Our offices around the globe

[Press & news](#)

[Partners](#)

[Contact us](#)

Source URL: https://www.harness.io/company/?45c01115_page=2

Get to know Harness

Harness aims to enable every software engineering team in the world to deliver code reliably, efficiently and quickly to their users.

Our Mission

At the core of our mission lies a deep commitment to empowering software engineering teams worldwide. We are dedicated to providing the tools, knowledge, and resources necessary for these teams to excel in their work. Our ultimate goal is to revolutionize the way software is developed, ensuring that every team, regardless of their size or location, can deliver code with utmost reliability, efficiency, and speed.

Harness's mission is to enable the 30 million software developers in the world to deliver code to their users quickly, reliably and efficiently.

Meet our Executive Team

Jyoti Bansal is a serial entrepreneur and technology visionary who believes passionately in software's ability to change the world for the better. He co-founded Harness in 2017 to automate and simplify all software delivery processes, and serves as CEO. In 2018, he co-founded Traceable, the leading API security platform, and venture capital firm Unusual Ventures. Unusual Ventures is reinventing the VC engagement model by providing entrepreneurs with an unprecedented level of services. Unusual closed its third fund in 2022 and currently has over \$1B under management.

Carlos Delatorre is the Chief Revenue Officer (CRO) at Harness, overseeing the sales and go-to-market (GTM) functions. With a proven track record of enterprise sales leadership, he has consistently built high-performance teams that deliver strong results. Prior to Harness, Carlos was head of Sales at several companies, including Navan and MongoDB.

Currently, Carlos is a board member at Yalo and an investor/advisor to companies including Modern Treasury, Vartana, Exafunction, Starburst, and Outreach. He earned his undergraduate degrees from the University of Miami and Troy State University, and holds an MBA from Troy State University.

John Bonney oversees the Harness company's finance and accounting function. Prior to Harness, he oversaw rapid growth as the CFO at Mapbox as well as FinancialForce, the latter quadrupling revenues to over \$100M during his tenure. Prior to that, he was the Division CFO of the Cloud Business unit at SAP where he oversaw over \$1 billion of Cloud revenues across multiple business units, products, and geographies. Bonney joined SAP from Ariba as a senior finance executive who helped drive rapid expansion and ultimately the sale of Ariba to SAP for \$4.3 billion.

Previously, Luan has served as an advisor for companies like PlanGrid, Headhuntr.io, and 8A8 Inc. He also served as a Guest Lecturer at University of California, Berkeley for the "Organization and Management" course with the International Diploma Programs. Luan holds a degree in International Relations from the University of California, Davis. He has written articles on the "War for Talent"; and has been featured in several publications including Forbes, The Wall Street Journal, The Guardian, and San Francisco Business Times, specifically on the art and science behind recruiting.

â

Sri is responsible for leading the company's global engineering organization, including its R&D teams in United States, Bangalore (India), Latin America, and Europe. Sri previously spent five years scaling Zoom from less than 12 million ARR in the early days to post IPO growth, ultimately leading a team of more than 600 people, launching the company's marketplace, and significantly expanding its ecosystem. Sri has also held key engineering leadership roles at Saba, Plantronics, and Cisco.

Gleb Brichko brings over 20 years of B2B marketing experience across software, security, and infrastructure. Most recently, Gleb built and led the global growth and demand marketing team at Rubrik, where he was responsible for leading, planning and executing the company's go-to-market strategy, brand presence, demand generation campaigns and programs, digital and web, partner marketing, field marketing, customer advocacy, strategic events, content, and operations. His prior roles include marketing leadership positions at Nutanix, Fortinet, ForeScout, and Blue Coat Systems. Gleb holds a bachelor's degree in mass communications from UC Berkeley.

Parmeet Chaddha leads post-sales customer success and services function at Harness. With multiple patents to his name, Parmeet brings 30+ years of general management experience in engineering, business development and customer success. Prior to Harness, he led customer success at Securiti.ai. Previously, he held senior executive roles at Google Cloud, Nutanix, EMC, IBM and Oracle. He has been honored as "Top 50 Technology Executives" by InfoWorld and the "Leading Data Consultants for North America, 2022" by the CDO Magazine. Parmeet holds a B.S. and M.S. from Massachusetts Institute of Technology.

Investors

Harness was spun out of BIG Labs, a startup studio designed to solve hard technology problems and build enduring companies. Harness has raised \$425M of venture capital from top-tier investors.

Our offices around the globe

[Press & news](#)

[Partners](#)

[Contact us](#)

Source URL: <https://www.harness.io/legal/aup>

Harness Acceptable Use Policy

As of Mar 31, 2023

This Acceptable Use Policy ("AUP") describes rules that apply to any party ("you", "your", "yours", or "Customer") using any websites, products and services (collectively, the "Software") provided by Harness Inc. or any of its affiliates (collectively, "Harness") and any user of any software application or service made available by Customer that interfaces with the Software ("User"). The prohibited conduct in this AUP is not exhaustive. Customer is responsible for its User's compliance with this AUP. If Customer or any User violates this AUP, Harness may suspend Customer's use of the Software. This AUP may be updated by Harness from time to time upon reasonable notice, which may be provided via Customer's account, e-mail, or by posting an updated version of this AUP at <https://harness.io/legal/AUP>.

1. No Inappropriate Content or Users.

Do not use the Software to transmit or store any content or communications (commercial or otherwise) that is illegal, harmful, unwanted, inappropriate, or objectionable, including, but not limited to, content or communications which Harness determines:

1.a is false or inaccurate;Â

1.b is hateful or encourages hatred or violence against individuals or groups; orÂ

1.c could endanger public safety.

This prohibition includes use of the Software by a hate group. Customer and its Users are also prohibited from using the Software to promote, or enable the transmission of or access to, any prohibited content or communications described in this paragraph.

2. Prohibited Activities.

Do not use the Software to engage in or encourage any activity that is illegal, deceptive, harmful, a violation of othersâ rights, or harmful to Harnessâs business operations or reputation, including:

2.a Violations of Laws. Violating laws, regulations, governmental orders, or industry standards or guidance in any applicable jurisdiction (collectively, âApplicable Lawsâ). This includes i) violating Applicable Laws requiring (a) consent be obtained prior to transmitting, recording, collecting, or monitoring data or communications or (b) compliance with opt-out requests for any data or communications, or ii) (h) using the Software in violation of any applicable laws or regulations (including any export laws, restrictions, national security controls and regulations) or outside of the license scope set forth in Section 2.1 (License Grant) of the Agreement;Â

2.b Configure the Software to collect or transmit any Prohibited Data, or transmit to Harness any Prohibited Data.Â âProhibited Dataâ means: (a) special categories of data enumerated in European Union Regulation 2016/679, Article 9(1) or any successor legislation; (b) patient, medical, or other protected health information regulated by the Health Insurance Portability and Accountability Act (as amended and supplemented) (âHIPAAâ); (c) credit, debit, or other payment card data or financial account information, including bank account numbers or other personally identifiable financial information; (d) social security numbers, driverâs license numbers, or other government identification numbers; (e) other information subject to regulation or protection under specific laws such as the Childrenâs Online Privacy Protection Act or Gramm-Leach-Bliley Act (âGLBAâ) (or related rules or regulations); or (f) any data similar to the above protected under foreign or domestic laws.

2.c Disclose or, except as reasonably necessary to use the Software, use any non-public information regarding Harness or the Software;

2.d Use the Software in support of any nuclear proliferation, chemical weapon, biological weapon or missile proliferation activity;Â

2.e Interference with the Software. Interfering with or otherwise negatively impacting any aspect of the Software or any third-party networks that are linked to the Software;

2.f Publish the results of any benchmarking tests run on the Software;Â

2.g Use the Software to

- store, download or transmit infringing, libelous, or otherwise unlawful or tortious material, or malicious code or malware;Â
- engage in phishing, spamming, denial-of-service attacks or other fraudulent or criminal activity;Â
- interfere with or disrupt the integrity or performance of third party systems, or the Software or data contained therein;
- perform, or engage any third party to perform, authenticated or unauthenticated penetration testing, vulnerability assessments or other security assessments on the SaaS deployment of the Software;

2.h Falsification of Identity or Origin. Creating a false identity or any attempt to mislead others as to the identity of the sender or the origin of any data or communications;Â

2.i Create a Harness customer account, access or use the Software in order to:

- monitor the Softwareâs availability, performance, or functionality for competitive purposes;Â
- copy ideas, features, functions, or graphics;
- develop competing products or Software; orÂ
- perform any other form of competitive analysis, as determined by Harness in its sole discretion;

2.j Using, or permitting direct or indirect access to, the Software for any form of excessive automated bulk activity such as spamming; inauthentic interactions, such as the creation or use of fake accounts and automated inauthentic activity; or mining or demonstrating proof-of-work or other proof by use of resources for any cryptocurrency or blockchain.

3. No Service Integrity Violations.

Do not violate the integrity of the Software, including:

3.1 Bypassing Service Limitations. Attempting to bypass, exploit, defeat, or disable limitations or restrictions placed on the Software;

3.2 Security Vulnerabilities. Finding security vulnerabilities to exploit the Software or attempting to bypass any security mechanism or filtering capabilities;Â

3.3 Disabling the Software. Any denial of service (DoS) attack on the Software or any other conduct that attempts to disrupt, disable, or overload the Software;

3.4 Harmful Code or Bots. Transmitting code, files, scripts, agents, or programs intended to do harm, including viruses or malware, or using automated means, such as bots, to gain access to or use the Software;

3.5 Unauthorized Access. Attempting to gain unauthorized access to the Software or Harnessâs systems.

4. Data Safeguards.

Customer is responsible for determining whether the Software offers appropriate safeguards for Customerâs use of the Software, including, but not limited to, any safeguards required by Applicable Laws, prior to transmitting or processing, or prior to permitting Users to transmit or process, any data or communications via the Software.

5. Notices and Reporting.

Violations of this AUP, including any prohibited content or communications, may be reported to legalnotices@harness.io. Customer agrees to immediately report any violation of this AUP to Harness and provide cooperation, as requested by Harness, to investigate and/or remedy that violation.

Source URL: <https://www.harness.io/legal/aida-privacy>

AIDA Data Privacy

As of

Unless otherwise agreed by you and Harness, use of the Harness AI Development Assistant (AIDA) is subject to the Harness Privacy Policy, the Acceptable Use Policy and the AIDA Terms.

What data does AIDA collect?

AIDA collects data, information, and other inputs to provide the service, some of which is then saved for further machine learning, analysis, and product improvements. AIDA collects data as described below:

Submission Data

Your submissions are transmitted to AIDA to provide Output to you. Submission data is only transmitted in real-time to return Output, and is discarded after Output is returned. AIDA does not retain submission data.Â

User Engagement Data

When transmitting submissions to AIDA it will collect usage information about events or actions generated from those submissions. These events and actions can include user actions such as output accepted and dismissed, whether the output was helpful or matched what you were looking for, and general usage data to identify metrics like latency, errors and engagement. This information may also be attributed to your Harness account, which will help Harness improve AIDA, provide support, and help troubleshoot issues. Harness will not share your Harness account information with any third party generative AI providers it integrates with.Â

How does AIDA use and share data?

User engagement data is used by Harness and other generative AI providers to provide the service and to enable improvements.

Such uses may include:

- Evaluating AIDA's effectiveness, for example, by measuring the positive impact it has on the user.
- Detecting potential abuse of AIDA or violation of the AIDA Terms or Acceptable Use Policies.
- Conducting experiments and research related to developers and their use of developer tools and services.

How can AIDA users control their data?

When you use AIDA, Harness collects and processes certain user engagement data and general usage data.Â This data may be shared with third parties as described above.Â However, Harness will not share your account information with third parties.

Please see the Harness Privacy Policy, for more information on how Harness processes and uses personal data.Â Â

Effective Date: June 19, 2023

Source URL: <https://www.harness.io/legal/service-packages>

Service Packages

As of Jan 11, 2024

Welcome to our Service Packages page, which contains a list of our various service packages that help bring your software delivery enhancements to life through best practices, thoughtful planning, collaborative sprints, and seamless, efficient execution.Â Below you can find the various onboarding service packages and corresponding terms and conditions.Â

- Residency PodÂ
 - Service Catalog Dollars
 - Accelerator Standardââ
 - âAccelerator Premium
 - Module Appendix
 - Custom Resource Appendix
-

Progressive Delivery: Everything You Need to Know

Progressive delivery is a practice that builds on the capabilities of Continuous Integration (CI) and Continuous Delivery (CD) to help you deliver with control. Software delivery helps organizations and teams get their changes out to customers quickly. Often, what gets forgotten is discussing how we ensure quality and reduce the risks of introducing said changes to customers. Delivering changes rapidly can often be at odds with not breaking things. This blog post will share what you need to know about progressive delivery as we share the ways organizations are using it today.

What is Progressive Delivery?

Progressive delivery is a practice that allows organizations to control how and when new software features or changes are delivered. It builds on the capabilities and practices of feature flag management and deployment strategies like blue-green and canary deployments. Ultimately, progressive delivery combines software development and delivery practices allowing organizations to deliver with control.

When discussing the software development life cycle (SDLC), we often mention continuous integration (CI) and continuous delivery (CD), which captures the how, when, where, and why of our software delivery. CI/CD is integral to quickly, safely, and repeatably we can get our software changes to our users and customers. Progressive delivery leverages your CI/CD pipelines alongside our software development and delivery practices to boost our software delivery capabilities.

The Key Elements of Progressive Delivery

There are several key components to achieving progressive delivery.

Source Code Management and Git Branching Strategy

Source code management is a practice that allows for the reverting, tracking, and correction of software code. A key element of progressive delivery includes application development code that is properly managed and tracked by a version control system like Git. In a basic Git-based workflow, you have the main branch from the main branch, which contains different versions of your application's code, branch from specific versions of the application code when working on a different feature. This allows for development progress to continue without the fear of losing changes as developers work on the same codebase.

Having source management is a key element for progressive delivery as it serves as the foundation for feature development, testing, and rollout.

Continuous Delivery, Continuous Integration, and Feature Flags

CI/CD is a shortened term for Continuous Integration and Continuous Delivery. CI/CD is the combination of principles, practices, and capabilities that allow for software changes of all kinds to get users in a quick, repeatable, and safe manner. This allows for software developers to integrate their feature or service changes continuously and for IT and operations teams to deliver with the standards, security, and confidence businesses need. CI/CD also serves as a foundation for your progressive delivery capabilities. Automated testing ensures that these integrations and deliveries maintain the desired quality.

You can learn more about CI/CD in this blog post.

Feature Flag Management

As a development codebase grows, so often does the development team. With new feature requests, developers and innovation syncing and aligning to a development pace can be difficult, especially for complex feature work. There are many challenges that long-running feature work face when it comes to delivering fast. Merge conflicts arise for long-lived development branches, and often, features require manual testing. Feature flags, also known as feature toggles, control the visibility of features behind a logic (often Boolean) flag. This allows teams to avoid redeploying an application to enable a feature for testing or production.

Feature flag management also has many benefits to development workflows. Contributors to a codebase don't have to wait for a feature to be fully finished to merge into the development or main branch. Developers can also test their features by controlling their feature flag and exposing their feature in a running application in a test or development environment.

A/B Testing

A/B testing, also known as split testing, is a technique that involves exposing two variants of the same application service to different segments of users. By shifting specific over to different versions of an application, teams can compare, experiment, and better understand how these changes impact a specific set of users. To effectively conduct A/B testing, it's essential to know how to create a flag that can toggle between the two variants.

Feature Release Strategies

Also known as feature rollout, these deployment or release strategies are used to change or upgrade a running instance of an application. There are three kinds of deployment strategies, rolling deployments, blue-green deployments, and canary deployments. Of these three

strategies, two are progressive since they involve progressively shifting user traffic to a running service.

The Blue-Green deployment is a deployment strategy that gradually transfers user traffic from a previous version(green) of an application or service to a new release(blue) of the application or service.

The Canary Deployment is a deployment strategy that releases an application or service to a subset of users.

Both blue-green and canary deployments limit the impact of application changes. These capabilities can be beneficial if deployments are risky and have severe consequences to the user base in breaking changes.

Observability

Observability leverages metrics, tracing, and logging capabilities to provide organizations and teams with answers to questions regarding their running services without the need to deliver another instance of the application or service to answer those questions. Understanding your services interact or perform can give developers a better understanding of what is breaking and what is performant. Often we need to understand an application's performance before we can achieve the confidence needed to release a change to an entire user base, and this is where observability matters.

Implementing Progressive Delivery

The best approach to implementing progressive delivery is to build the key elements of progressive delivery incrementally. A common practice is to invest in your development workflows. This can involve introducing feature flag management capabilities through an SDK or by building it a home-grown solution. Some other organizations may need to introduce source code management practices first before even discussing A/B testing capabilities or feature flag management. If you're starting with software development and would like to read more about git-based workflows and how development and feature branching works, take a look at this source-code management guide.

Some organizations use feature flag management within their development processes already but need more capabilities around verifying deployments. It may make sense to invest efforts into monitoring or observability solutions to measure and understand application services. You can't compare or baseline performance without these capabilities. And changes go out with confidence, so this can often be the next step in your progressive delivery journey.

And lastly, there is a deployment aspect to progressive delivery. Your platforms should support deployment strategies like canary or blue-green deployments. Deployment capabilities can come from your continuous delivery platform or other solutions like a service mesh. If you're looking for a complete solution to deploy your services to production safely, I recommend looking at Harness's CI/CD platform.

Improve your Software Delivery with Harness

Today software delivery isn't just about speed. Progressive delivery joins software development and delivery practices to better support and control your software delivery. If you're looking to move fast without breaking things and implementing any progressive delivery element quickly, I recommend trying Harness for free.

For more on progressive delivery, let's look at Feature Flags!

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

[Documentation](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[Case studies](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?
Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/legal/modern-slavery-statement>

Modern Slavery Act Transparency Statement

As of

Introduction

For the Fiscal Year Ending January 31, 2024

This is the modern slavery statement (âStatementâ) for Harness Inc. and its subsidiaries (collectively, âHarness,â âwe,â or the âCompanyâ). This Statement constitutes Harnessâ forced labor and human trafficking statement for Harnessâ fiscal year ending January 31, 2024.

Our Business and Supply Chain

Harness is a leading end-to-end platform for complete software delivery. The Company provides a simple, safe, and secure way for engineering and DevOps teams to release applications into production. Harness uses machine learning to detect the quality of deployments and automatically roll back failed ones, saving time and reducing the need for custom scripting and manual oversight, giving engineers their weekends back.

Harness Inc. is a privately held Delaware corporation headquartered in San Francisco, CA with subsidiaries in the Americas, Europe, and APAC.

Based on our business model and due to the fact that Harness does not manufacture a tangible product, Harness believes that the risk of modern slavery in our business and supply chain is limited.

Commitments

Harness recognizes that modern slavery is a significant worldwide problem and commends the efforts of governments and the private sector to eradicate it. Ensuring human rights are respected is fundamental to how we operate as a business. Our internal policies pertaining to human rights articulate the high standards we hold ourselves to and the expectations we have of the third parties with which we work. We do not tolerate any form of modern slavery or human trafficking in our business and are committed to preventing and detecting any such activity in our own operations and supply chain.

Managing Risk

Even though Harness is not at high risk for modern slavery in our business, we incorporate protections into our relationships with third parties by obligating all vendors, suppliers, distributors, partners, business associates, and third-party representatives to comply with our Vendor Code of Conduct and all applicable governmental laws, rules, and regulations.

Reporting

In accordance with our internal policies, employees may report any actual or potential violations of applicable law to Harness via the Companyâs whistleblower hotline. The whistleblower hotline can be contacted through the following channel:

- EthicsPoint Website (contains country-specific instructions for both online and phone submission of reports)

Harness prohibits retaliation in any form against anyone who, in good faith, reports any actual

or potential violations of law, or any illegal or unethical behavior. All issues raised via the whistleblower hotline are reviewed and assessed for remediation, where circumstances warrant. We promote awareness of this reporting channel through internal policies, communications, and training.

Next Steps

As one of our core values is Continuously Improve, in fiscal year 2025 we plan to focus on the following goals:

We recognize that Harness must continue to develop our policies and procedures with respect to modern slavery risk, and we are committed to monitoring and enhancing our approach to mitigate this risk to our business.

Source URL: <https://www.harness.io/legal/vendor-code-of-conduct>

Vendor Code of Conduct

As of Jun 01, 2023

Introduction

Harness Inc. and its subsidiaries (âHarness,â we,â or the âCompanyâ) operate business in a responsible manner and strive to achieve the highest standard of business and professional integrity. We expect our vendors, suppliers, distributors, partners, business associates, and third party representatives (collectively, âVendorsâ) to do the same. Â

This Vendor Code of Conduct (the âCodeâ) sets forth Harnessâs expectation that our Vendors uphold the highest standards of ethics and comply with all applicable laws and regulations. This expectation is in addition to, and not in replacement of, any Vendorâs existing legal and contractual obligations to Harness, including Harnessâs Partner Code of Conduct (as applicable). Failure to comply with this Code may result in termination of Harnessâs business relationship with a Vendor. We reserve the right to review and update this Code at our sole discretion.

Harness encourages Vendors to bring questions or concerns about this Code to their Harness point of contact or compliance@harness.io.

Compliance with Applicable Governmental Laws, Rules, and Regulations

Harness expects its Vendors to comply with both the letter and the spirit of all laws, rules, and regulations that apply to the Vendorâs business, particularly those related to Vendorâs performance of duties for Harness.

Anti-Bribery and Anti-Corruption

Harness prohibits bribes, kickbacks, or other improper or illegal payments of anything of value from being directly or indirectly offered, given, authorized, promised, solicited, or accepted in any way related to Harness, whether it involves public officials (including officers or employees of governments or state-owned entities) or private parties. Â

Harness prohibits bribery to influence a public official, to obtain or retain business from any party, or to secure an unfair business advantage. Â

Harness also prohibits Vendors from making facilitation payments, or small, unofficial payments to public officials to expedite routine, non-discretionary government decisions (even if permissible under local law). Â

Antitrust, Competition, and Fair Dealing

Harness expects its Vendors to comply with applicable antitrust and competition laws designed to Â promote fair and open competition, particularly as it relates to Harness. Â

Vendors are prohibited from directly or indirectly entering into any formal or informal agreement with competitors that fixes or controls prices, divides or allocates markets, limits the production or sale of products, boycotts certain suppliers or customers, eliminates competition, or otherwise unreasonably restrains trade.

Vendors must deal fairly with Harnessâs customers, service providers, suppliers, competitors, and employees. Â

Vendors may not take unfair advantage of anyone through manipulation, concealment, abuse of Â privileged information, misrepresentation of material facts, or any other unfair dealing practice.

Export, Customs, Trade Control, and Anti-Money Laundering

Harness expects its Vendors to comply with all applicable export, customs, and trade control laws Â and regulations, including economic and trade sanctions laws, antiboycott laws, and any related Â licensing requirements.

Harness also expects its Vendors to comply with all applicable anti-money laundering laws and Â regulations.

Conflicts of Interest and Corporate Opportunities

Vendors must avoid actual or potential business or financial conflicts of interest involving Harnessâi.e., instances where the Vendorâs personal interests (including interests of the Vendor Â itself or the Vendorâs employees, officers, or directors) interfere or appear to interfere with Harnessâs interests.

Vendors are prohibited from directly or indirectly (a) taking personally for themselves opportunities that are discovered through the use of Harness property, information, or positions; (b) using Harness property, information, or positions for personal gain; or (c) competing with Harness for business opportunities.

Any actual or potential conflicts of interest must be promptly reported to Harness.

Record Management and Recording Transactions

Vendors are expected to ensure that all financial books, records, and accounts related to their Â relationship with Harness accurately reflect transactions and events. Vendors must not falsify documents, transactions, or accounting records related to Harness.

Vendors must also establish and maintain a risk management system to verify compliance with applicable laws and to quickly recover and continue operation of their business in the event of a disaster.

Confidential Information

We expect our Vendors to safeguard and protect Harnessâs confidential information, as well as the confidential information of Harnessâs customers, suppliers, stockholders, Harness employees, and/or other third parties. Confidential information should be interpreted broadly to include all non-public information relating to Harness or other companies that would be harmful to the relevant company (or useful to competitors) if disclosed.

Harness prohibits Vendors from misusing proprietary information or trade secret information that Â was obtained without the ownerâs consent, or from using confidential information for personal gain.

Vendors are also prohibited from issuing any press releases or making any other public statements regarding Vendorâs relationship with Harness absent Harnessâs prior written approval.

Data Privacy

Vendors must comply with all applicable laws and regulations regarding the protection of personal information or other sensitive or protected information. Vendors must also assist Harness in complying with its own obligations in this regard.

Human Rights, Fair Labor Practices, and Freedom of Association

We expect our Vendors to comply with all applicable laws prohibiting forced labor, human trafficking, and slavery. Vendors must not use any form of forced labor including prison, indentured, bonded, military, slave, or any other forms of forced labor. Vendors must not participate in the recruitment, transportation, transfer, harboring, or receipt of any persons by means of threat, use of force, or any other forms of coercion, abduction, fraud, deception, abuse of power, or position of vulnerability, or the giving or receiving of payments or benefits to achieve the consent of a person having control over another person for the purpose of exploitation. Vendors must not retain an employeeâs government-issued identification, passport, or work permit as a condition of employment and must allow employees to resign from their positions at any time.

We also expect our Vendors to comply with all applicable laws prohibiting child labor. Vendors must not directly or indirectly employ workers that are younger than the minimum employment age established by the respective country or local jurisdiction. In the event no minimum employment age is established, Vendorsâ employees must not be younger than the age of compulsory education.

Vendors must comply with all applicable wage and hour laws, including those relating to minimum wage, overtime hours, and other elements of compensation, and must provide all legally mandated benefits. Vendors must not require employees to work more than the maximum number of hours permitted under applicable laws.

Vendors must adhere to applicable laws regarding employeesâ rights to affiliate with lawful organizations without interference.

Harassment

Harness expects its Vendors to conduct themselves in a professional manner with courtesy and Â respect for others. We do not tolerate harassment by our Vendors in any form. We also expect our Vendors to ensure their employees can perform work in an environment free from verbal, physical, or sexual harassment, or other abusive conduct.

Non-Discrimination

Harness is committed to providing equal opportunities in employment, development, and Â advancement for all qualified personsâand we expect our Vendors to share that commitment. Â Harness does not tolerate illegal discrimination by its Vendors. Vendors must maintain a workplace free of unlawful discrimination, which includes, but is not limited to, race, gender, sexual orientation, age, pregnancy, caste, disability, union membership, ethnicity, religious belief, or any other factors protected by applicable law.

Environment, Health, and Safety

Harness expects its Vendors to comply with both the letter and the spirit of applicable health, safety and environmental laws and regulations, including requirements for chemical and waste management and disposal, recycling, industrial wastewater treatment and discharge, air emissions controls, environmental permits, and environmental reporting. Vendors must also provide workers with a safe and healthy work environment.

Use and Protection of Harness Corporate Assets

If provided with Harness assets (including technology, software, proprietary information, or other physical assets), Vendors are expected to protect these assets and ensure their efficient use for legitimate business purposes.

Compliance

Vendors must establish and maintain a process of ensuring compliance with this Code. This includes communicating the requirements of this Code to all employees, affiliates, agents, and subcontractors of the Vendor. Vendors must maintain all documentation necessary to demonstrate its compliance with this Code, which may be demonstrated through compliance with Vendor's own code of conduct or applicable company policies. Upon Harness's request, Vendor should be prepared to provide Harness access to such documentation.

Reporting

Vendors may report any actual or potential violations of this Code or applicable law to Harness via the Company's whistleblower hotline. The whistleblower hotline can be contacted through the following channel:

- EthicsPoint Website (contains country-specific instructions for both online and phone submission of reports)

Vendors must prohibit retaliation in any form against anyone who, in good faith, reports any actual or potential violation of this Code, or any illegal or unethical behavior.

Source URL: <https://www.harness.io/legal/previous-privacy-statement-july-23>

Previous Privacy Statement

As of

Harness Inc. (âHarness, âWe, âUsâ) is committed to protecting your data privacy. This Harness Privacy Policy (âPrivacy Policyâ) details our privacy practices for the activities described in this Privacy Policy. Please take the time to read this Privacy Policy carefully in order to understand how we collect, share, and otherwise process information relating to individuals (âPersonal Dataâ), and to learn about your rights and choices regarding our processing of your Personal Data.

In this Privacy Policy, âHarness, âwe, âour, â and âusâ each mean Harness Inc., and the applicable Harness affiliate(s) involved in the processing activity. The addresses of our offices, where Harness Inc. and our affiliates are located, can be found at <https://harness.io/company>.

1. Harness's Responsibilities and Roles

Harness is the controller of your Personal Data, as described in this Privacy Policy, unless otherwise stated. This Privacy Policy does not apply to the extent that we process Personal Data in the role of a processor (or a comparable role such as âservice providerâ in certain jurisdictions) on behalf of our customers, including where we offer to our customers various cloud products and services, through which our customers (and/or their affiliates) connect their own websites and applications to our hosted platform, sell or offer their own products and services, send electronic communications to other individuals, or otherwise collect, use, share or process Personal Data via our cloud products and services.

Each of our customers, not Harness, controls whether they provide you with a subscription to the Harness platform, and if they provide you with a subscription, they control what information about you that they submit to our service. This content may include contact information (such as your first and last name, email address, and phone number), professional information (such as the department you work for at your place of employment), or other types of information a customer chooses to submit. Use of this content by Harness is governed by agreements between Harness and the Customer.

For detailed privacy information applicable to situations where a Harness customer (and/or a customer affiliate) who uses Harness's cloud products and services is the controller, please reach out to the respective customer directly. We are not responsible for the privacy or data security practices of our customers, which may differ from those set forth in this Privacy Policy. If not stated otherwise either in this Privacy Policy or in a separate disclosure, we process such Personal Data in the role of a processor or service provider on behalf of a customer (and/or its affiliates), who is the responsible controller of the applicable Personal Data.

If your Personal Data has been submitted to us by or on behalf of a Harness customer and you wish to exercise any rights you may have under applicable data protection laws, please inquire with the applicable customer directly. Because we may only access a customer's data upon instruction from that customer, if you wish to make your request directly to us, please provide to us the name of the Harness

customer who submitted your Personal Data to us. We will refer your request to that customer and support them as needed in responding to your request within a reasonable timeframe.

2. Personal Data We Collect and Data Sources

a. Covered Data Processing Activities

This Privacy Policy applies to the processing of Personal Data that we collect in the following ways, as detailed in this section.

We collect information about you when you provide it to us, when you interact with our websites and electronic systems, and when you attend events and visit our offices, and when other sources provide it to us, as further described below.

b. Information You Provide To Us Based on our current practices (and including our practices over the last 12 months), we collect the following categories of information about you:

- Contact and Professional Data: We collect contact and/or professional data about you in person, through communications, and through our websites. For example, you provide your contact and professional information to us when you sign up to learn more about Harness's products and services, download content, register for an event and visit our offices. If you attend an event, we may also receive contact and professional details about you when you choose to scan your attendee badge or code, or by providing a business card or other method where you share Personal Data with us. Typically, contact data includes your name and contact methods, such as telephone number, email address, and mailing address, and professional data includes details such as the organization you are affiliated with, your job title, and industry.
- Biographical, Community, and Support-related Data: We may also collect various types of biographical, community, and support-related Personal Data from you via our help center and community support forums. For example, if you register for an online community that we host, we may ask you to provide a username, or biographical information, such as your occupation, organization name and areas of expertise. Additionally, you may provide Personal Data to us when you create user-generated content (for example, by posting in a forum), provide Harness with feedback, or when you participate in interactive features, trainings, online surveys, contests, promotions, sweepstakes, activities, or events. You may also be asked to provide contact information, a summary of the problem you are experiencing, and any other information that would be helpful in resolving a customer support request.
- Job Applicant Data: You may provide your contact and professional information, including your resume with educational and work background, when you apply for a job with Harness. You may also provide us with sensitive information, like your Social Security Number, passport, or other government identifier, racial or ethnic origin, or other such Personal Data in connection with your job application.

c. Personal Data We Collect From Other Sources In the course of doing business (and over the 12 months preceding the effective date of this Privacy Policy), we receive Personal Data and other information from other third parties for our business or commercial purposes. This information varies and typically falls into a few categories:

- Business contact information (such as name, job title, business email, phone number, and address), and/or social profile (such as LinkedIn) which may include other details about your company;
- Third-party platform usernames and identifying information; and
- Details about you as a job candidate (which may include your name, resume, educational and work history, criminal history information, and feedback) as permitted under law.

We receive business contact information that contains Personal Data for commercial purposes, including details about your company from third parties for marketing and business intelligence, such as analyzing business opportunities, identifying potential new customers, and providing our audience with more relevant content and advertising. Typically, we receive this information about you from a few sources, such as (i) third-party marketing initiatives, such as events where we are a sponsor or website forms hosted by third parties that may provide content about us; (ii) when you sign up to attend an event hosted by us or another third party; (iii) companies such as information aggregators and entities from whom we have licensed business contact information; and (iv) referrals. In some situations, we may combine such business contact information with other non-personal and Personal Data we possess or that you have provided to us. For example, we may combine business contact details with details about your organization, such as its address or revenue range, and analyze this information for business opportunities or use this to send you tailored content.

We also receive information from third-party platforms for various business purposes program management, or technical reasons. If you participate in an open-source project or our public bug bounty program, we may receive details about you, such as your username or pull requests, to help us manage your participation in the project or program and provide you with updates.

If you are a candidate applying for a job at Harness, we may receive Personal Data about you from third parties for business purposes, such as thorough background checks (educational, employment, criminal, and financial information), publicly available sources (like social media accounts, including LinkedIn for identifying candidates), feedback about your application and from interviews, and other third parties that may provide feedback about your application.

For our professional services work, as a processor or service provider, Harness may also receive Personal Data about you to perform its obligations under its contract with a third party. Harness partners may also share your business contact information with Harness as part of their recommendation to your company to become a Harness customer. If Harness is interested in partnering with, acquiring, investing in, or partners with, acquires, or invests in your employing or retaining company, Harness may receive Personal Data about you through the (potential or completed) transaction for its business purposes.

3. Device Data, Usage Data, and Metadata We Collect

a. Explanation of Device Data, Usage Data, and Other Metadata and Technology Used Harness collects certain Personal Data from users of its website similar to most websites, applications, and software across the Internet. This type of data collection allows us to better understand how individuals use our websites, products and services and how they perform. For example, we may collect metadata about you, including technical data about your performance or use of our website, products and services. We may also collect device data about you to help us determine that users from one type of device use our websites, products and services in different ways than users of a different type of device, which in turn allows us to improve our websites, products and services, such as through making sure our customers' users have a more efficient user experience. Such data may also include browser type, operating system, Internet Protocol (IP) address (a number that is automatically assigned to your computer when you use the Internet, which may vary from session to

session), domain name, and/or a date/time stamp for your visit.

One common technology we use to collect metadata that may be considered Personal Data is our use of cookies. "Cookies" are small pieces of information that are stored on your hard drive or in device memory. We link the information we store in Cookies to the personally identifiable information you submit while on our site. We may use both session Cookies (which expire once you close your web browser) and persistent Cookies (which stay on your computer until you delete them) to provide you with a more personal and interactive experience on our website. Your Web browser may allow you to delete, block or otherwise disable the use of Cookies or HTML5. If you choose to disable Cookies, some areas of our website may not work properly.

The three main types of cookies are:

Essential cookies. Essential cookies are required for website functionality and security. For example, authentication, security, and session cookies may be required for our website or products to work. Functional cookies. We use functional cookies to help enhance our websites' performance, for market research, or other analytics or advertising that is not tied to a specific individual. For example, we may use Google analytics to help us track how many individuals visited our website. We may also utilize HTML5 local storage cookies for the reasons described in this section. These types of cookies are different from browser cookies in the amount and type of data they store and how they store it. Targeting or advertising cookies. We use targeting and advertising cookies to help us understand our marketing efforts and to reach potential customers across the web. For example, we contract with third-party advertising networks that may track your activity over time and across different channels, including our websites, email activity, and other websites and applications that display advertisements. They may use this tracking information to help us understand and predict your interests, to display an advertisement for Harness on another website, or email you with a marketing communication for a Harness product. A second common technology we use to collect metadata that may be considered Personal Data is beacon technology. We use beacons in our websites and in email communications to you. Beacons provide us with information about your activity and help us to improve our business operations and strategy, such as by understanding our email communications' functionality and improving our websites and content. For example, if you click on a marketing email we send to you about a new product or service, the beacon will provide signals to us that you and your organization may be interested in learning more.

b. Data Collected from Harness Products and Ancillary Products We offer products that collect both customer related data and usage related from Harness products. Our collection of both types of data enables us to provide and innovate upon Harness products, which in turn allows us to act as a service provider to our customers and to continuously improve upon the services we provide to our customers. In conjunction with the products we make available to our customers, we may collect additional data, such as account usernames, user-agent and browser version, the URLs you visit, logs of your usage and click activities, logs about your login history, identity confirmation, and device data (such as whether your device is managed by an administrator, the operating system installed on the device, and similar device or version information). Collectively, we refer to this data as "Ancillary Data". Some of the Ancillary Data we receive is dependent on your organization's policies and settings and what it permits to be shared with Harness. Harness uses Ancillary Data to improve security and to provide and improve its products to customers, including to better understand customer behavior to create new features and provide threat-related insights for our customers. For example, we may use the URL you visit to let you better manage your passwords for the websites you visit.

The following product collects and processes Ancillary Data: the Harness browser plugin. Through the Harness browser plugin, the Ancillary Data we collect includes details about your login session, IP address, user-agent, and the web application name and website address, as well as other information that is not personal in nature. In addition, we may collect interaction data about your use of the Harness browser plugin. To opt-out of this, please see the section on Information Choices below. We use the information collected through the Harness browser plugin for security purposes and to provide features, such as by allowing you to better manage your passwords for websites that you visit.

c. Intentional Disclosures to Third Parties As part of the functionality we make available on our websites and to better reach our customers and prospective customers, there may be categories of third parties that are authorized by us to operate on our websites and access your Personal Data, such as your contact data, IP address or cookies. Depending on your location, (for example, California and the European Union), Harness only shares Personal Data with such third parties if you agree to such sharing through a website banner or form. In other parts of the world, this information may be automatically collected when you visit our websites. These categories of third parties include, but are not limited to, advertising networks and social networks, including Google, Facebook, LinkedIn, Twitter, Reddit and Quora. At any time, you may choose to withdraw your decision to share personal data with these third parties through our websites by visiting the section on Information Choices below. For specific details on these companies' privacy practices, please visit their privacy policies.

4. How We Use Personal Data

How Harness uses the Personal Data it collects depends in part on how you choose to communicate with us, how you use our websites and interact with us, and any preferences you have communicated to us. In general, we use your Personal Data as is necessary to run our business and carry out our day-to-day activities. In addition to the uses identified elsewhere in this Privacy Policy, we may use your Personal Data to accomplish the following tasks (and we have done so during the 12 months preceding the effective date of this Privacy Policy):

Communicate information about our products and services. We may use your Personal Data, such as contact data, Ancillary Data, and metadata, to send you transactional communications, notices, updates, security alerts, and administrative messages regarding our products and services that may be useful to you and your organization. We will respond to your questions, provide tailored communications based on your activity and interactions with us, and help you use our products and services effectively. Support safety and security. We use Personal Data, such as contact data, Ancillary Data and other metadata, about you and your use of our products and services to verify accounts and activity, monitor suspicious or fraudulent activity, assist our customers in their monitoring of suspicious or fraudulent activity, and identify violations of policies regarding the use of our products and services. We also process Personal Data for other security reasons, such as to register visitors to our offices and to manage non-disclosure agreements that visitors may be required to sign. Market and promote our products and services. We use your Personal Data, such as contact data, Ancillary Data, and

other metadata, about how you use the products and services to send promotional communications that may be of specific interest to you and your organization, including by email and by displaying Harness marketing communications on other companies' websites and applications, as well as on third-party platforms like Facebook, Twitter, and Google. These communications are aimed at encouraging engagement and maximizing the benefits that you and your organization can gain from Harness's products and services, including information about new products and features, survey requests, newsletters, and events we think may be of interest to you and your organization. Manage contests or promotions. Harness may occasionally run contests or other special promotions, and if you register for one, we may process your Personal Data, such as contact information, biographical information, and contract-related data to perform our contract with you. Harness may also use the Personal Data, such as contact data, collected in these contests and promotions to send you promotional material about Harness or our partners. Process payments. We process Personal Data, such as contact information, contract-related data, financial information, biographical information, and payment information to process payments to the extent that doing so is necessary to complete a transaction and perform our contract with you or your organization. We process your Personal Data, such as contact, job applicant, and biographical data, to assess your application and to evaluate and improve the recruitment system, our application tracking and recruitment activities. We also use your Personal Data to communicate with you regarding your application or opportunities at Harness and to send you new hire and employee experience information. We may verify your information, including through reference checks and, where allowed, background checks. Other purposes for our legitimate interests: Where required by law or where we believe it is necessary to protect our legal rights, interests, or the interests of others, we may use your Personal Data in connection with legal claims, compliance, regulatory, and audit functions, protecting against misuse or abuse of our products and services, and protecting personal property or safety. Other purposes with your consent: We may use your Personal Data if you have given us consent to do so for a specific purpose not listed above. For example, we may publish testimonials or featured customer stories to promote our products and services, with your permission. If we process your personal data for a purpose other than that set out above, we will provide you with information prior to such processing.

Legal Bases for Processing Personal Data (for European Economic Area Individuals)

If you are an individual in the European Economic Area (EEA), we collect and process information about you only where we have a legal basis or bases for doing so under applicable EU laws. The legal bases depend on the products and services that your organization has purchased from Harness, how such products and services are used, and how you choose to interact and communicate with Harness's website, systems, and whether you attend Harness events. This means we collect and use your Personal Data only where:

We need it to operate and provide you with our products and services, provide customer support and personalized features, and to protect the safety and security of our products and services; It satisfies a legitimate interest of Harness's (which is not overridden by your data protection interests), such as for research and development, to provide information to you about our products and services that we believe you and your organization may find useful, and to protect our legal rights and interests; You give us consent to do so for a specific purpose; or We need to comply with a legal obligation. If you have consented to our use of Personal Data about you for a specific purpose, you have the right to change your mind at any time, but this will not affect any processing that has already taken place. Where we are using your Personal Data because we or a third party (for example, your employer) have a legitimate interest to do so, you have the right to object to that use; however, in some cases, this may mean that you no longer use our products and services.

5. Personal Data Shared by Harness

Service Providers. For all categories of information that we collect, we share Personal Data with our service providers for various business purposes, including, but not limited to, auditing interactions with users, debugging our websites, products and services, security purposes, internal research and gleaning insights through machine learning and artificial intelligence, short-term uses such as credit verification, payment processing, IT services, quality control and safety, as well as to perform other services on our behalf. For example, we may use service providers to host our customer relationship management system. **Event Sponsors.** If you choose to register for or attend an event or webinar that we host (such as our Unscripted conference), enter a contest or raffle with us and a sponsor, or download content (such as a whitepaper) from our website, then we will share your contact information, content interest information or other activity data, and any other information, including Personal Data, collected in the course of these activities for commercial purposes with those sponsors. In many cases, you intentionally disclose your details by providing your information to these sponsors through consent via a registration form or by scanning your badge at the applicable sponsor's booth or entering your access code online. The treatment of this information is subject to each of these third parties' respective privacy statements. **Partners and Resellers.** We share your Personal Data, such as contact information, business details, and content interest and activity details, with our partners and resellers for business purposes, such as to carry out our business or for joint marketing efforts to reach our customers and prospective customers. In many cases, you intentionally disclose your details by providing your information to these sponsors through consent via a registration form. **Protection of Rights, Security and Fraud Detection.** For all categories of data we collect, we share your personal data with third parties for business purposes to protect our customers, users, secure our physical and intellectual property, and to prevent or investigate security or fraudulent attempts against our users through our platform. **Law Enforcement and Legal Requests.** For all categories of data we collect, we may share Personal Data to comply with applicable law or respond to valid legal requests, such as a subpoena, from law enforcement or other authorities. **With our Affiliates, Related to Corporate Transactions, and Provision of Professional Services.** For all categories of data we collect, we share Personal Data among our affiliates and subsidiaries for business purposes, including any service providers and agents that work on our behalf. For example, we may share your Personal Data with support service providers with whom we have in place agreements to protect your Personal Data. We may also share your information as required for us to carry out a corporate transaction, such as a merger or sale of assets of all or part of our company. We will also share your Personal Data with our professional service providers (for example, our auditors, insurance providers, financial service providers, and legal advisors) as needed for us to run our business. **Platform Analytics Data.** We share metadata (for example, unique identifiers and usage data) collected through our platform with analytics service providers for our business purposes, such as to provide a better user experience and generally help make our products and services better. **Advertising and Marketing.** We share your Personal Data, such as metadata and contact data, with third-party advertising and marketing providers, to allow us to better reach our customers and prospective customers, and to sell our products and services. In some circumstances we may ask you to consent to directly disclosing your Personal Data with these third parties prior to sharing your Personal Data, such as via a consent banner on our website. **Anonymous or De-identified Usage Data.** We share anonymized or aggregated usage data or security threat information with third parties or the public. For example, this may include sharing trends regarding organizations' use of Harness's products and services to customers and prospective customers in our

âBusinesses at Workâ report. The data shared in this category is not Personal Data. Harness Community, Help Center, and Other User Generated Content. We make available community forums, as well as blogs and other means for you to post information on our websites. This is publicly-available information that you choose to share and it may be read, collected, and used by others that visit these websites. Except for username (which may be your real name) and the details that you choose to include in your profile, the categories of data shared in these circumstances will depend on what you choose to provide. Recruitment Data. When you apply for a job at Harness, we share your Personal Data, including applicant data, biographical information, and other Personal Data we possess with our affiliate companies for business reasons, such as human resource management and internal reporting; our service providers for business reasons, such as the recruitment platform and to manage background checks; and law enforcement or government authorities, or as otherwise necessary to comply with law.

6. Harnessâs Security Measures

Security is a critical priority for Harness. We maintain a comprehensive, written information security program that contains industry-standard administrative, technical, and physical safeguards designed to prevent unauthorized access to Personal Data.

However, no security system is perfect, and due to the inherent nature of the Internet, we cannot guarantee that data, including Personal Data, is absolutely safe from intrusion or other unauthorized access by others. You are responsible for protecting your password(s) and maintaining the security of your devices.

If you use the Harness online service via a subscription purchased for you by a Harness customer, then that customer is responsible for configuring your instance appropriately. Additional information about security settings and configurations can be found in the documentation related to our online service, which is available at <https://ngdocs.harness.io/>

7. International Data Transfers

Your Personal Data may be collected, transferred to, and stored by us in the United States, and by our affiliates and third parties that are based in other countries. The addresses of our offices where Harness, Inc. and its affiliates are located can be found online at <https://harness.io/company/about-us/>.

Your Personal Data may be processed outside your jurisdiction, and in countries that are not subject to an adequacy decision by the European Commission or your local legislature and/or regulator, and that may not provide for the same level of data protection as your jurisdiction, such as the European Economic Area. We ensure that the recipient of your Personal Data offers an adequate level of data protection, for example, by entering into standard contractual clauses for the transfer of data as approved by the European Commission (as described in Article 46 of the General Data Protection Regulation) (if required), or we will ask you for your prior consent to such international data transfers.

8. Children

Harnessâs websites are not directed at children. We do not knowingly collect Personal Data from children under the age of 13. If you are a parent or guardian and believe that your child has provided us with Personal Data without your consent, please contact us by using the information in the âContact Usâ section, below, and we will take steps to delete such Personal Data from our systems. Harness has not knowingly sold Personal Data (including the Personal Data of minors under 13 without affirmative authorization) to third parties in the 12 months preceding the effective date of this Privacy Policy).

9. How Long Does Harness Keep Your Data?

We will retain your Personal Data for a period of time that is consistent with the original purpose of the data collection, or as necessary to comply with our legal obligations, resolve disputes, and enforce our agreements. We determine the appropriate retention period for Personal Data by considering the amount, nature and sensitivity of your Personal Data processed, the potential risk of harm from unauthorized use or disclosure of your Personal Data and whether we can achieve the purposes of the processing through other means, and on the basis of applicable legal requirements (such as applicable statutes of limitation).

10. Information Choices

a. Your Privacy Choices In the above sections, we describe how we may collect, use and share your Personal Data for providing relevant content and advertising. Below, we describe how you may unsubscribe, opt-out, or otherwise modify settings related to our processing of your Personal Data.

Direct Email Marketing. If you wish to withdraw from direct email marketing communications from Harness, you may click the âunsubscribeâ button included in our emails. Please note, you cannot unsubscribe from critical transactional emails that are related to our provision of our online Service (such as those related to security and your Harness account).

Direct Marketing â Phone or Postal Mailings. If you wish to withdraw from phone call or postal mail marketing communications from Harness, please request to do so by sending us a notice at privacy@harness.io.

Analytics. To opt-out of analytics on our websites, you may adjust your cookie preferences as described below. If you are a user of the Harness online service via a subscription purchased for you by a Harness customer, to opt-out of platform-based analytics on an individual level, please contact us at <https://harness.io/opt-out/>.

Cookie Preferences. To manage the use of targeting and advertising cookies, please see details below:

We use OneTrust as a service provider to help you manage cookies. Cookie Settings for our OneTrust preference center to opt-out of relevant advertising cookies. You may also adjust your web browser settings to opt-out of non-essential cookies. Please understand that blocking or deleting non-essential cookies may affect our websitesâ functionality. Note that any choice with regards to cookie-based advertising only applies to the web browser through which you exercise that choice. You will still continue to see advertising, including potentially from Harness, even if you opt-out of personalized advertising.

b. Your European Privacy Rights Under the General Data Protection Regulation, if you are a European Union data subject, you have rights to understand and request how we collect, use, and disclose Personal Data in our capacity as a data controller, to the extent permitted by applicable law.

Right to Access. You have the right to access your Personal Data held by us.

Right to Rectification. You have the right to rectify inaccurate Personal Data and, taking into account the purpose of processing, to ensure it is complete.

Right to Erasure (or âRight to be Forgottenâ). You have the right to have your Personal Data erased or deleted.

Right to Restrict Processing. You have the right to restrict our processing of your Personal Data.

Right to Data Portability. You have the right to transfer your Personal Data, when possible.

Right to Object You have the right to object to the processing of your Personal Data that is carried out on the basis of legitimate interests, such as direct marketing.

Right Not to be Subject to Automated Decision-Making. You have the right not to be subject to automated decision-making, including profiling, which produces legal effects. Harness does not currently engage in the foregoing on our websites or in our products and services.

If you would like to make a request and exercise your rights described above, please submit your request at <https://preferences.harness.io/privacy> or send an email to privacy@harness.io.

c. Your California Privacy Rights Under the California Consumer Privacy Act of 2018 (âCCPAâ), effective January 1, 2020, if you are a California resident, you have rights to understand and request that we disclose how we collect, use, disclose, and sell your Personal Data to the extent permitted by applicable law.

Right to Know About Personal Data Collected, Disclosed, or Sold. You have the right to request that we disclose what Personal Data we collect, use, disclose, and sell.

Right to Request Deletion of Personal Data. You have the right to request the deletion of your Personal Data collected or maintained by us as a business. Right to Opt-Out of the Sale of Personal Data. You have the right to opt-out of the sale of your Personal Data by us as a business, in the event, we sell Personal Data.

Right to Non-Discrimination for the Exercise of Your Privacy Rights. You have the right not to receive discriminatory treatment by us for the exercise of your privacy rights conferred by the CCPA.

Authorized Agent. You may designate an authorized agent to make a request under the CCPA on your behalf by us with a copy of your power-of-attorney document granting that right.

Financial Incentives. We do not provide any financial incentives tied to the collection, sale, or deletion of your Personal Data.

If you would like to make a request and exercise your rights described above, please submit your request at <https://preferences.harness.io/privacy>, send an email to privacy@harness.io, or contact us at the address listed below. If you would like to opt-out of Personal Data sharing with marketing and advertising third parties through the use of cookies, please see the section above on Your Privacy Choices. Since there is no reasonable way for us to verify the information that we process on behalf of our customers, we will not respond to any requests in relation to data we hold on behalf of our customers.

11. Contacting Harness

Harness welcomes your comments or questions regarding this Privacy Policy. Please contact us at the following postal or email address:
Harness Inc.

Attn: Legal Department 55 Stockton Street, San Francisco, CA 94108
Email: privacy@harness.io

12. Changes to the Policy

This Privacy Policy may be updated from time to time, to reflect changes in our practices, technologies, additional factors, and to be consistent with applicable data protection and privacy laws and principles, and other legal requirements. If we do make updates, we will update the âeffective dateâ at the top of this Privacy Policy webpage. If we make a material update, we may provide you with notice prior to the update taking effect, such as by posting a conspicuous notice on our website or by contacting you using the email address you provided.

6 Use Cases for Kill Switches in Production Using Feature Flags

If you've only come to feature flags recently, you may not know that feature flags as a concept has existed for quite a long time, and can easily be seen as an evolution of app config systems. But first things first - if you need a refresher on feature flags, head on over to our "What Are Feature Flags?" article now.

One of the least used - but most powerful - use cases of kill switches using feature flags can be for Ops and SRE teams. Plenty of folks never think about this potential, but it can be a way in which feature flags can most transform your engineering organization. They're a great way to increase velocity and reduce risk in software development.

Kill switches create tremendous potential both to control granular capabilities or features within an application, and also as long-lived operational switches that affect the entire application. And while feature flags are intuitively thought of as affecting frontend or client-facing applications, they can also be used just as well on the backend. Let's explore how you can leverage kill switches to change the nature of how your team can deliver software.

Use Case 1: Control in Production With Feature Toggles

Consider this the base case of implementing kill switches using feature flags. The concept is simple enough: you can rig any and all features and changes pushed to production (or any environment) with a toggle, or a kill switch. Feature flags, by nature, allow teams to isolate features within their applications so that they can be handled individually instead of as a deployment bundle. Kill switches themselves are the way in which teams can create a control to disable code in production at any time.

This basic kill switch functionality gives teams the ability to turn off a feature when it's not ready. "Not ready" can mean two things in prod:

Unfinished Features

The first condition for not being ready is what slows down many teams during the deployment process. In particular, as many teams work on different features across different applications, integration and deployment take a long time. Without having everything rolled in, release teams and the software development teams that built them both find themselves slowed down.

Using feature flags as kill switches in this context allows teams to push new features to production even as incomplete features, which can result in faster development cycles. It simply needs to be pushed live, but with the kill switch activated (feature turned off) so that it's not actually impacting anything.

Broken Features

Here's a developer favorite: it worked in all the pre-production environments, and then a bug was discovered that's now impacting production. Without the use of a feature flag, that broken feature impacting prod would now necessitate a complete rollback of the deployment - that's right, all of the good features have to be pulled off the shelves too and production is switched back to the last working version.

On the other hand, rigging each new feature or change with a feature flag enables teams to hit the kill switch as soon as an issue is found. Now, all of the features that work get to stay in prod, while the broken features are isolated out and handled as required.

Use Case 2: Incident Management

What happens when a severe issue occurs in production because of feature releases that are broken? Or when some infrastructure failure cascades across all of production? As it turns out, kill switches can be used to mitigate these incidents and improve the MTTR (mean time to resolution) for the teams responsible. Talk about stress management.

Scenario 1

In scenario 1, a broken new feature is causing an incident in production. The good news is, if that new feature was rigged up to a feature flag, the kill switch could be triggered and that feature could be turned off in production right away, resulting in minimal impact. Not only is the feature no longer affecting production, but there is also suddenly no massive response team that needs to be assembled to handle the issue, which would typically mean the whole deployment had to be pulled. Instead, the feature is turned off, the team responsible for the feature is on the hook to fix it, and the one feature fix can be rolled forward when it's ready.

Scenario 2

In scenario 2, an infrastructure failure is one example of what could happen in production. However, production outages can happen due to a variety of issues. Oftentimes, Ops teams will have fallback options or "just turn it all off" runbooks in place, and kill switches can be used here as well. If it's a new infrastructure change entirely, that whole change can be wrapped in a feature flag and a kill switch can be triggered that will failover to the old setup when an issue occurs. Some teams also create long-lived operational flags that can, say, put a whole site or application in "maintenance mode," effectively triggering a kill switch for the whole application. This can be useful in P0

scenarios where things are completely broken and the preference is to not allow any access.

Use Case 3: Application Load Management

Let's say your business is an e-commerce company. During the holidays, you experience high load relative to the rest of the year. Now more than ever, the business doesn't want to have downtime or have customers experience issues! Turns out feature flagging is an incredibly low-cost way to solve this problem.

One approach to solve this problem could be to simply turn off some features that will reduce the load, and this is a great place to use a kill switch, but it might not be the most effective. Another solution might be to kill certain load balancers or servers that are useful in non-peak times, and cut over to infrastructure that's designed specifically for high load.Â

Real World Example

At a previous job, we had log data that would need to go into storage from the short term cache, so we had a system to collect and store them. During peak loads or other periods of instability (particularly upstream with AWS), we would often see this storage become either very expensive or very unreliable. So, we would turn this service off during these periods, but this was manual and involved a series of AWS commands and database changes to both disable and re-enable later on. By wiring this up to a feature flag, we were able to turn this feature off and on instantly in a much more lightweight, visible, and easy to govern way.

Use Case 4: Testing in Production

One of the most common use cases for teams doing feature flagging is testing in production. In short, this is where a specific feature (or set of features) needs to be tested against real production data. After all, pre-prod environments can only test so many things! There are two use cases that we see here:

The use of feature flags as kill switches here is a pretty simple one: have it live until the test has been run, and then turn it off. It's also possible that a feature will fail during testing and need to be turned off to minimize impact. You can dig deeper into testing in production in this dedicated blog.

Use Case 5: Progressive Delivery

Here, let's assume a basic knowledge of progressive delivery. It's becoming an increasingly popular feature release methodology, very similar to the methodology used for canary releases in Continuous Delivery.

With progressive delivery, the need for a kill switch changes slightly. Yes, the main use case is still turning things off when they break or aren't needed, but here, we have to layer in automation. By its nature, progressive delivery is something that teams run in an automated fashion. To do that, they need to pre-define in which scenarios flags are turned on, for whom, and under what conditions more users get access. Conversely, they must define failure criteria in which the kill switch is triggered and new features are turned off, or the user base is shrunk.

In Harness Feature Flags, we automate progressive delivery via the Pipeline. Teams can schedule releases, mandate approvals, integrate with plugins, create trigger events, and template rollouts - and then automate it.

Use Case 6: Personalization

Personalization is a hot topic. Kill switches using feature flags are of huge value in being able to do this well at scale. We can look at three distinct scenarios in which you can use kill switches to improve your ability to personalize experiences for customers.

Compliance & Regulation

When it comes to regulation, it's a non-negotiable item to be able to "personalize" something like a mobile app release to comply with the laws in various countries, or to meet the needs of clients in specific verticals (e.g. government, finance). The most common of these is data privacy - specifically, complying with GDPR.

Considering this, it can be a mess showing a button in one place but not another. Wiring key features up as permanent feature flags allows you to easily turn something on for all your North American users, but not your European users where GDPR compliance is mandatory. It's not just creating a kill switch that given context will turn off data collection in Europe, it's also being able to kill data collection for any user that decides to opt out. It's much easier than having a configuration file that tries to solve for all of the various scenarios.

User-Defined Personalization & Admin Settings

Think of this as giving control to the user on what features they want to kill. This isn't dissimilar to giving them access to admin settings on the app so they can choose to have dark mode or choose what notifications they get.

While on the frontend, users will define what they want and don't for their personal experience on the app, on the backend, it's just exposing parts of the feature flag schema to the end user and letting them make the decision on what to keep and what to kill.

Another consideration here is the use of features like screen time or parental controls, where a kill switch can be triggered to limit the

access of users to specific applications altogether based on settings elsewhere on their devices. Of course, putting the onus on the user to define what they will do with such power is a whole other problem to solve.

Using Harness Feature Flags to Implement Kill Switches

Hereâs the plug: using Harness Feature Flags gives you the ability to implement all of these use cases right out of the box. When you set up feature flags in Harness, youâre able to work either in a visual UI or entirely in code to manage your kill switches. You also give yourself peace of mind when you eventually want to scale your usage of kill switches or other feature flags use cases with built-in governance, compliance, and security considerations, as well as the critical integration into CI/CD (Continuous Integration and Continuous Delivery).Â

After all, you donât want to set up a great feature flagging system that is entirely disconnected from the rest of your software delivery process. It doubles your work on access control, security, governance, and integration maintenance. You end up getting diminishing returns on your feature flagging platform, which is supposed to *accelerate* software delivery.

If you want to see how Harness can fit into your organization, you can sign up for free forever, or contact us for a personalized product demo.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/harness-launches-source-available-continuous-delivery>

Harness Launches Source Available Continuous Delivery

No Red Tape. No Barriers to Entry.

As companies switched to Kubernetes and microservices, Harness created the first Continuous Delivery-as-a-Service solution to tackle modern software delivery. Harness has helped enterprises and small businesses deploy faster with more confidence by abstracting away the complexity of new technologies. Now we are bringing our world-class CD pipelines to users through a source available license. Our goal is to remove the barriers that prevent developers from experiencing our platform.

Developers at large enterprises who have faced hours of security reviews, executive meetings, and internal negotiation to get approval to start using Harness CD can bypass bureaucracy and get straight to deploying.Â

Teams at smaller companies who needed to deploy software without committing to an enterprise contract can build and deploy their first pipeline in roughly 15 minutes.Â

Tinkerers who want to try the latest and greatest software on their own machines have an easy way to evaluate Harness CD.Â Â

The Harness CD Community Edition is provided under a source available license called Polyform Shield License, which is available for use free of charge for most users. We have open source technologies, like Drone, and plan to share more code via open source in the future. If youâd like to find out more about our commitment to open source check out our website.

Harness CD Community Edition

Harness CD Community Edition joins existing enterprise editions of our Continuous Delivery product. The first iteration is only available for Kubernetes (on any platform), and in the near future, AWS (EC2, ECS, ASG, CloudFormation, Lambda, etc.), Azure (VMSS, WebApps, AKS, ACR, ARM, Blueprint), .NET, Google Cloud Build, VMware Tanzu, & Serverless deployments will be supported. Users will have access to Harness CDâs staple of features such as automated canary & blue-green deployments, automated infrastructure provisioning, integrated approval & notification flows, and a developer-friendly pipeline-as-code experience. Harness CD Community Edition also comes with built-in user and secrets management along with built-in approval workflows.Â

The Harness CD Community Edition can be downloaded and run on your laptop or VM with 3 GB of RAM and 2 CPUs. Once downloaded, a streamlined quick start guide will help you create and deploy a pipeline in 15 minutes. Harness CD Community Edition users can upgrade to a paid plan at any time.Â

How to Install, Build, and Deploy a Pipeline

Harness CD Community Edition can be found on GitHub.

Before using Harness CD Community Edition, youâll need to either have Docker Desktop installed or have a Kubernetes cluster where you can install a Helm chart. For this post, we are going to use the Docker Desktop option. In the Resources tab of your Docker Desktop, increase memory by 3 GB RAM and CPUs by 2 CPUs.Â After installing Harness CD, you can deploy microservices to any Kubernetes cluster as long as you have network connectivity to that cluster. The easiest option would be to enable the built-in Kubernetes cluster that ships with Docker Desktop while making sure you allocate additional memory and CPU resources for this cluster as well.

Once you have completed the prerequisites, you can install Harness CD Community Edition. Run the following commands to clone the content of the Harness Git repo into your local directory (docker-compose folder).Â

```
git clone https://github.com/harness/harness-cd-communitycd harness-cd-community/docker-compose/harness
```

Then run the below command to download all required Harness images and bring up the containers.

```
docker-compose up -d
```

Next, youâll need to create an account here: <http://localhost/#/signup>. Now youâre on your way! The Harness wizard will help you set up a project, set up a pipeline, and complete your first deployment.Â

You can also follow the Harness CD Community Edition Quickstart doc that shows you how to deploy a nginx service onto your Kubernetes cluster using the standard rolling deployment strategy. You can run this pipeline and see a deployment similar to the one shown below.

Just the Beginning

This is our first GA release of Harness CD Community Edition, and it is the first time weâve taken a subset of our SaaS offering to create software you can run yourself. SaaS and software are completely different mediums, so we would love to get your feedback! You can give us feedback or ask us any questions on our community page.

Join us live to learn more about Harness CD Community Edition.

You can find the link to the Harness CD Community Edition on GitHub. Download and deploy within 15 minutes to see how you can level up continuous delivery in the new year!

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/devsecops-strategies-secure-applications>

What is DevSecOps? Strategies to Secure Applications

In today's modern business environment, every company is a software company. Therefore, companies should optimize their software development lifecycle to ensure they get the most out of their software engineering investments. This means that businesses should adopt DevOps and modern cloud-native methodologies to improve efficiency and effectiveness in their software engineering processes. Even though companies have DevOps teams, security is somehow getting left behind or, most often, ignored. Sometimes, it's left up to the security teams alone.Â

Where DevOps brings together development organizations and operations teams, the DevSecOps culture (or, development security and operations) adds in the objectives of security teams for shared responsibility. Almost every organization today has some form of technology integral to their operations. Infrastructure security is imperative, and simply adding in security measures for employees is not enough. DevSecOps takes security to the next level with secure code, release management, automated security checks, security controls, and more â all in the production environment before code reaches users. In short, DevSecOps bridges the divide between development, operations, and security practitioners.

This all means many companies are not taking the necessary steps to ensure their software systems, methods, and applications are secure. This has led to a new term in the industry â **DevSecOps**.

The DevSecOps methodology was created by Netflix, Etsy, and Amazon Web Services as they faced an onslaught of cyberattacks that threatened their business continuity. DevSecOps implementation helped these companies improve their response time by moving away from reactive IT practices toward proactive application development. This methodology aims to also reduce the time it takes for developers to push code into production. A DevSecOps environment can be created in any company, regardless of size or industry.Â

DevSecOps in the Software Development Lifecycle

DevSecOps is a new way of approaching application security and taking it seriously rather than as an afterthought. It is an approach to securing the software development process and applications by integrating security into the software development lifecycle (SDLC). With the emergence of DevSecOps, there has been a shift in how we think about security and react to threats. While it focuses on shifting left in the SDLC, it also ensures that security is everyone's job.Â

You could test the code changes for security vulnerabilities through a CI system even before any artifact is created, and then you could test again in between each stage to make sure the true artifacts are being passed to the next stage. DevSecOps aims to prevent security threats before they occur by implementing best security practices throughout the SDLC with more secure code.

Benefits of DevSecOps

Organizations can expect to see significant benefits from implementing a DevSecOps process.Â

Some of the most common benefits include:Â

- Increased software quality and build security as developers become more serious about threats and aware of the code they contribute in software releases
- Finding the security loopholes in the applications and actions to fix them. A continuous integration (CI) tool integrated with an application security testing tool gives more visibility to the vulnerabilities in the code.Â

- Automation of testing processes to ensure they are kept up to date and resolve all issues quickly.
- Improvement of customer experience and developer productivity.
- Organizations can build software that is secure from the start by *shifting left* approach. This means customers will not have to worry about data breaches and can fully trust the software they are using thanks to more secure software.
- Reduced time to market. With a DevSecOps strategy, organizations can eliminate bottlenecks resulting in deployment delays. This means companies can deliver software on time and ready to deploy.
- Increased team collaboration. A DevOps implementation encourages collaboration between the development and operations teams. A DevSecOps strategy takes this step further by including other teams, such as security and business stakeholders, in the process.

Notable DevSecOps Strategies

DevSecOps helps to address these concerns by integrating security into the development process. It also helps to secure the development environment, which is an important step in protecting against cyber attacks. We have some DevSecOps strategies listed below to tackle and mitigate security issues.

Automated Testing for Security Vulnerabilities

One of the biggest challenges when implementing DevSecOps is integrating a security test phase into the SDLC. For years, code testing has been something that was left behind until the end of the project. It used to be ignored, or, even if automated, it was often done poorly.

With DevSecOps, testing needs to be integrated and automated into the SDLC. Code scanners can help with identifying vulnerabilities but lack accuracy, and manual penetration testing is time-consuming and costly. Automated tools can be used to detect vulnerabilities and enforce security standards along with policies. In addition, security tools can be used to identify vulnerabilities in code.

Some DevSecOps tools and practices include:

- Code-level testing is done by inspecting the code and looking for dangerous packages, insecure configurations, and risky parameters.
- Code scanners can help find unsafe functions like strcpy or unsecured calls to system commands.
- Configuration management prevents issues that could allow unauthorized users to access sensitive data.
- Through dynamic application security testing (DAST), potentially risky parameters are being passed to a function, which could be manipulated to cause malicious actions.

Harness Security Testing Orchestration (STO) is one such tool that can help organizations prioritize application security vulnerability data and deliver highly secure applications while maintaining deployment velocity and minimizing rework.

DevSecOps and Continuous Integration and Continuous Delivery (CI/CD)

Another significant concept in DevSecOps is employing CI/CD. CI/CD helps development teams automate code commits, build and test the code, and deploy it to the specified environment. In addition, developers can automate testing to find security issues in their application code by integrating application security as part of their production environment pipeline. Therefore, having a robust CI/CD platform is a must and the prerequisite to do DevSecOps because it integrates continuous monitoring into development cycles. At different stages of the DevOps pipeline, we can have security checkpoints such as vulnerability scanning, JIRA approvals, adherence to governance and security policy, software composition analysis, and more.

Development Teams Test Hard and Test Smart

Your development team needs to act like hackers and security breachers and do not give any chance for them to enter your SDLC premises. By configuring your development cycle with all the possible security analysis and testing tools. With platforms like Harness, integration with any test suite is possible. Make sure you configure from simple tests to load tests to availability tests, so your CI/CD pipeline is attack free.

You can see that the above CI/CD pipeline is configured with various testing suites, which ensures security for the application.

Culture of Automation and Ownership

Another vital aspect of DevSecOps is the culture of automation and ownership. Developers need to be given the freedom to automate processes independently, but they also need to own their code. This means they are responsible for everything in their code, including the security risks. Developers also need to be given the tools to automate processes efficiently. For example, if you want to automate testing, you must have procedures and tools to run these tests. Many tools can help with automation, such as OWASP ZAP, Burp Suite, or Twist. These tools can be integrated into the code delivery process and trigger automated security tests at different stages.

Determine Risk Based on the Criticality of Assets

Another critical aspect of DevSecOps is determining the risk based on the criticality of assets. You cannot treat every change the same; some might pose big risks, while others don't. When it comes to security, you need to know what risks are in your application. This can be done by using a risk-based approach. You can use a risk-impact matrix to identify risks and assign them a severity based on how they affect your application. You can also use a risk-grade model that helps you identify risks and set a priority for them. This can all be done using platforms like Harness's STO (Security Testing Orchestration).

The Harness STO has the capability to centralize security logs and results from over 40+ security scanners into a single dashboard of results. Data centralization also allows you to kick off continuous improvement initiatives. When data is in silos, it is much harder to analyze the results of the development, and hence the quality of improvements will be low.

Have Secrets Management in Place

Your applications will have some type of valuable credentials. It is very critical to ensure the encryption of such credentials and valuable information through secrets management. It can be your GitHub repo auth secrets, database credentials, etc. If such things are leaked, the attackers can easily exploit and pose a security threat. With platforms like Harness, it becomes very easy to keep our application secrets as they get encrypted through Harness secret manager. We also support external security management as well as OOB security management with Harness systems.Â

Â

â

Establish a Security Review Process for Code Changes

Once you know what risks are in your application, itâs important to establish a security review process for code changes. This will help you to track changes and identify which team member is responsible for each change. It can also help you identify patterns in the code and see if any vulnerabilities have been introduced or have not been resolved since the last code review.Â

Monitor for Threats and Anomalies in Real-Time

The threats and anomalies can be monitored in real-time with the help of threat intelligence and anomaly detection. Threat intelligence is the process of gathering, analyzing, and distributing cyber threat information in real-time. It includes malware alerts, network signatures, and malicious IP addresses. Threat intelligence can help identify observed threats and has been proven to work in real-time. Itâs essential to monitor for threats and anomalies in real time because threats are evolving and must be resolved as soon as possible.

The Need for DevSecOps

Businesses must implement DevSecOps, as it will help them to keep their software and applications secure while increasing their speed to market. Adopting DevSecOps has numerous benefits, as we mentioned above in the article. Tackling security issues to patch common vulnerabilities is now possible with DevSecOps practices. Everything from writing code and testing to deploying applications needs to be approached from a new angle when adopting DevSecOps principles. With platforms like Harness, it becomes a reality to handle these complex vulnerabilities, integrate security in the SDLC and streamline the software delivery.Â

If you're interested in learning more about using Harness and DevSecOps software delivery, try it for free and check out our Developer Hub for more step-by-step tutorials, videos, and reference docs.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?
Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/what-are-feature-flags?utm_source=pmm-ff&utm_medium=leveraging-feature-flags-for-zero-downtime-database-migrations

What are Feature Flags and How Do We Use Them?

Building software is a great experiment. Many times we are required to do what has not been done before; this is core to innovation work. However, a lot of the iteration and experimenting tends to happen before software is shipped. That's where the concept of test in production comes into play. Building something from scratch or even jumping into an older project mid-stream requires lots of iteration.Â Â

Once we deploy, for many software engineers, the experimentation on the existing tends to drop off and onwards to the next set of features in the backlog. Product owners/managers typically collect feedback to help prioritize the next set of features. From a software engineering perspective this can feel a little âchicken or egg-ishâ - without a feature actually deployed in a production environment to get feedback, how do we if know we are on the mark?

Letâs illustrate this with an example. Customers are asking for an updated UI view for their dashboards; they want something that surfaces the most relevant information faster. Now that the product team has collected those requirements, itâs up to engineering to figure out how to best build it. There might be a few potential solutions:

With some potential solutions decided, now itâs up to the engineering team to decide which of the three to implement. And before a customer ever sees it, they have to choose 1 of 3, build out the MVP over the next few sprints, and then ship it. This is the first time theyâll get any feedback, and if itâs not the right solution then it has to be built again!

Now, imagine if this new feature cycle could be improved for product development teams. If youâve been paying attention to developer tools over the last few years, you might have seen a new category popping up: feature flags. Feature flags as a concept were born to increase development velocity and to solve this very problem through practices like progressive delivery.

While feature flags as a mature category of hosted tooling is new, feature flags - and what they solve - have been around a little longer. Letâs take a look at what feature flags have grown into, and how to use them.

The Story of Feature Flags

Feature flags, at their core, are the ability to wrap different versions of your code in conditional statements that you can turn on and off at will. You may have previously thought of them as app configs, or even part of your companyâs rules engine.Â

With Harness Feature Flags, weâve taken these concepts you may be familiar with and taken them even further - adding more sophisticated rules, governance, user management, an intuitive UI, and more.Â

Feature Flags lets you separate feature release from code deployment, build more effective strategies for rolling things out and testing with your users, and have more insight than ever into how changes impact your application performance.

Ultimately, feature flags give engineering teams the ability to deliver more features, with less risk. For leaders, itâs a great way to improve engineering velocity and developer experience, and to reduce the risk associated with feature development. And for developers, itâs a great way to work more efficiently, with less stress - think fewer late nights before a deployment, and fewer firefights post-deploy.Â

What Are Feature Flags (or Feature Toggles)?

Feature flags are a way that developers can conditionally turn certain sections of their code on or off. You can think of feature flags as extending Continuous Delivery into Continuous Deployment - a way to put changes into production behind a flag and turn them on in a controlled way later (or hide and remove them in the same way).

But you can also think of feature flags as a new way to think about building and releasing applications - by clearly defining features and components that can be changed and toggled on and off individually, you allow for experimentation, controlled rollouts to different users, as well letting more people (like PMs, sales, and support) turn things on and off for customers.

Together, this results in giving more power and control to developers to create more and better features faster, while at the same time ensuring that engineering is not the bottleneck for the business in having to make changes for individual customers or ensuring a perfect production deployment for every new feature or feature update.

If we want to simplify even further, feature flags essentially create private swim lanes for developers where they can ship a feature directly to customers and then have control over who sees it, get feedback, and turn it on and off as needed. Itâs like a light switch (a pretty complex one)!

Who Uses Feature Flags?

People oftentimes consider feature flags either an engineering tool, or a tool for product managers. The reality is, it's both. Flags can help software development and DevOps teams lower their overhead and increase their velocity, and they can help product managers better control releases, coordinating launch timings and creating a feedback loop more effectively.

Research indicates that 95% of engineering leaders want to implement a feature flag solution for themselves. This comes on the back of 97% of leaders saying they're under pressure to deliver faster, and 65% of whom say they find it difficult to achieve that velocity increase in a safe manner. It's abundantly clear that engineering teams are likely the biggest beneficiaries of implementing and using a feature flag solution, and indeed, they tend to be the primary users. And when they do implement a feature flag solution, they end up delivering 66% more features per application per year!

Feature flags are great for users across the organization beyond product and engineering. For sales and support teams, feature flags provide a way for them to directly manage betas and new features for their customers, and to track down who's using what in case of any issues. And for management, flags can provide new forms of visibility into what's happening in development and how new features are being tested with users.

On a fun fact kind of note and since we're in the "who uses feature flags" section, it's cool to note that big companies like Netflix, Flickr, Google, Reddit, and more use feature flags! They're definitely worth looking into and absolutely are an important part of the software delivery lifecycle.

Why You Should Consider Feature Flags

Whether or not the use of feature flags is important to your software delivery lifecycle comes down to what it is that you need to accomplish. While feature flags are a great tool to have for any team, they are particularly useful if you have these problems:

Problems in the Business

- The risk associated with new feature releases ends up slowing down developers since they have to thoroughly validate design and implementation before releasing.
- There are often multiple possible versions of how a new feature will look and function, and it's up to the developer to choose one (not always the right one) and implement it.
- The decision of when to release new software features is controlled by developers instead of the business.
- Sunsetting old features introduces risk to newer features that might have dependencies on the old code.
- It's all or nothing - a feature is rolled out to everyone at the same time instead of doing a feature rollout incrementally for verification purposes.
- When teams are beholden to engineering release cycles (e.g. once a month), multiple features are released simultaneously, introducing complexity and risk that can result in deployment war rooms, large rollbacks, and dissatisfied customers.

Problems in Engineering

- Releasing a feature takes too much coordination and time across stakeholders.
- Merging new features, or managing separate feature branches for changes, is cumbersome and complex.
- You want to test changes to see how they impact your application and end-users before rolling them out more widely.
- You want to provide access to new features, or to test changes with certain portions of the user base sooner - by location, by customer plans, by customers opting into beta testing, or by any other criteria or logic you can think up.
- You want to reduce the risk of rolling out new code by having instant kill switches that don't require rollbacks or new deployments to fix a disrupted environment.

To simplify, we can consider that feature flags are a great solution if you're trying to do the following things:

- Increase engineering velocity and ship more features.
- Decrease feature development risk by A/B testing multiple solutions with customers.
- Reduce the risk and toil associated with production failures and rollbacks.
- Decouple engineering deployments from feature unveilings to customers.

Different Types of Feature Flags/Toggles

It's natural to want to put all feature flag use cases in a single bucket, but it's more helpful to instead view flags themselves as a means to an end and as having categories of use cases.

Here are some examples of the different kinds of flags that could be used in a system:

- **Release features** - Using flags to let the right people decide who to turn a feature on for, and when, without complex engineering deployment coordination required.
- **Experiment** - Using flags to learn something about how a change impacts things. This can be a technical experiment, such as how a new UI impacts server load, or it can be a user experiment such as how a new button impacts conversion.
- **Ops flags** - Letting you have permanent control around key parts of your app so you can turn them on and off as necessary without a full new deployment cycle. Add in RBAC, and this is a robust way to control who has access to modify your production application.
- **Control access** - Using flags as a way to manage betas, early adopter list, trial access, and more.

Implementing Feature Flags

Implementing feature flags can seem really simple at first, but it turns out that like with any system, the devil is in the details. Building an awesome feature flag solution in-house certainly has its perks, but it can be deceptively difficult. In particular, we've seen three distinct sets of issues arise with feature flag solutions as an organization's implementation matures.

When a feature flag solution is **nonexistent or immature** and an organization is standing it up for the first time, they tend to run into issues with finding value in the tool quickly. Especially with the need for such a solution and the pressure to make it happen quickly, it's critical that value is demonstrated quickly, otherwise it may end up leaving a sour taste. Within that, there's a dichotomy of architecting for eventual scale versus getting an MVP out the door to prove value. This often results in complex design or architecture that isn't scalable and only works for small, targeted use cases.

As feature flag solutions enter a **middle stage of maturity**, or as they **start to scale**, the problem is quite obvious: scaling is hard. Especially if built in-house, ensuring that the tool can be used by multiple users and multiple teams for a variety of use cases, all while maintaining a common set of best practices and building on a single infrastructure can be incredibly daunting. And as this is being figured out and teams have code in production, it can quickly result in stale code or issues keeping production tidy when code changes occur. Add onto that the need to integrate neatly into everyone's workflow and you've got a behemoth of a problem to solve.

Building on that is the scenario we often see with **highly mature** feature flag implementations, where they primarily end up as separate systems from existing software delivery or CI/CD pipelines that teams now have to learn how to use, and to figure out how to port their existing processes or pipelines into the feature flag solution. At this stage, there's the obvious cost of maintenance and future development to keep up with new changes or requirements. Technical debt is bound to build up by this point, and it can become unmanageable without a dedicated team - and perhaps even its own codebase.

Sure, 95% of engineering leaders may want a feature flag implementation, but as it turns out, it's not so easy to build something that works well over time.

Build or Buy: Feature Flag Management

You may be thinking, "Feature flags seem pretty simple, can't we just build this ourselves?"

The real answer? It's complicated.

We've seen in most cases that an internally-built tool will end up causing a lot of overhead and putting a very low ceiling on the benefits you can receive from feature flags. It may be good for your first couple of flags, but it can be hard to make an internal system scalable, performant, and robust for all the things flags can do as your organization gets more and more comfortable using them - and more reliant on them.

For a deeper look at the build vs buy conversation, you can check out an article we wrote earlier this month, [Feature Flags: Should I Build or Buy?](#)

Why Use Harness for Feature Flag Management

Harness provides a feature flag platform made for getting you up and running quickly while being able to scale to any level. We make it easy to get started and quickly see the benefits of feature flags. With Harness Feature Flags, that means you can get your first flag out the door in minutes by following three steps:

- Create a free trial account.
- Create a new flag in just 3 clicks.
- Add one of our SDKs to your code in a few minutes.

And, you've got a flag working!

If that's not quite clear, then it's worth knowing that the fundamental way flags work in an environment like Harness is that SDKs, added to your application as a library, will receive updates on flag statuses to evaluate with your users (you should choose between real-time streaming or pulling updates on intervals). These SDKs are secure, collecting no data you don't want to share, and they are designed for performance and stability.

Harness Feature Flags is fast, secure, and focused on helping teams get the most out of feature flags - from your first initial use all the way to global governance, sophisticated reusable templates, and developer-first experiences.

And if you're wondering how we stack up against LaunchDarkly, Optimizely, Cloudbees Feature Management, Split, and other FF tools - those pages are being worked on right now and we'll update this post with the links when they're live!

If you'd like a little more information on our architecture and on how to get started, you can check out the feature flag documentation or request a demo.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/architecting-for-cost-savings-at-tyler-technologies?utm_medium=blog&utm_source=internal&utm_content=architecting-for-big-query-costs

Architecting for Cost Savings at Tyler Technologies

This is a guest post by Harness customer Chris Camire - Senior Manager, Technical Services, Tyler Technologies

â

When you host cloud infrastructure for your clients across the globe, cloud costs can easily spiral out of control, especially when client non-production environments are fully operational every hour of the day. At Tyler Technologies, that was certainly true for our Enterprise Permitting & Licensing solution. Tyler Technologies is the largest software provider in the U.S. that is solely focused on the public sector; its Enterprise Permitting & Licensing solution is used by government agencies to automate and streamline their community development and business management operations.

We originally tried to tackle this problem with AWS CloudWatch, using alarms to power down non-production infrastructure outside of working hours. We knew this wasn't a long-term solution; our clients often work late hours and need their non-production environments available on-demand. Inversely, we also had the challenge that some environments weren't used daily, or even weekly, but could be needed on a moment's notice.

Ultimately, we decided to adopt Harness Cloud AutoStopping® to help us get control of our idle cloud resources. It's been a massive success, we've gotten a great deal of benefit from it. You can read more about that in our case study that we did with Harness.Â

This blog doesn't focus on how we're using Cloud AutoStopping though, because we realized that before we created a single Cloud AutoStopping rule, we needed to take a good look at how our infrastructure was organized in order to really maximize cloud cost savings.

Organic Growth Leads to Infrastructure Sprawl

Each of our clients is provided with a set of non-production environments, on non-production infrastructure, organized into âpodsâ. These environments are used by our clients to test our software before deploying into their production environments. Given the regulatory nature of the permitting and licensing services we provide to government agencies, ensuring that new features or bug fixes are fully tested and vetted before pushing to production is critical.Â

As our client base grew over the years, we continued to build additional pods to meet demand, and ensure clients had secure, reliable environments to test in. What we didn't do was create a strategy for how we organized our clients across that cloud infrastructure.Â

What we ended up with was clients across very different time zones being hosted in the same pod. Which meant that, from one end of the country to the other, there would always be a client in that pod that needed access to their environments.Â

Being Intentional About Organizing Shared Infrastructure

As we thought about reorganizing our infrastructure for cloud cost savings, we started with one key goal: organize in such a way that Cloud AutoStopping could stop the instances in our pods as much as possible. We're taking a 2-step approach to this, first by client time zone, and then by client activity.Â

Organizing by Time Zone

Categorizing our clients by time zone was the obvious first choice, since it would group clients that start and stop their workdays at roughly the same time. Given that our client base consists solely of public agencies, we already knew which states and time zones those agencies worked in. We got to work and started migrating the underlying applications into new âpodsâ that were designated by timezone.

This new organization has definitely helped us optimize the efficiency of our pods; they are now generally fully idle at the same time, enabling Cloud AutoStopping to power down these instances until they are needed again. When a client needs their environment, they access it just as they normally would, Cloud AutoStopping detects the incoming traffic, auto-starts the instances, and the client is off and running.

The results have been amazing since we started the process 6 months ago. Our cloud cost savings have increased exponentially as we've configured Cloud AutoStopping on more pods; initially saving us \$15K to \$20K a month and now passing the milestone of saving \$100K last month alone.

Organizing by Activity

When we started the process 6 months ago, trying to organize our clients by their activity level was nowhere on our radar. We didn't have any real visibility into that level of our clients' usage. But as we've continued to roll out Cloud AutoStopping across our infrastructure, it has become clear just how much some clients are (and aren't) using their environments based on the idle / active times we see in the Harness Cloud AutoStopping console.Â

The processes and procedures that government agencies have in place to test new software patches or releases can vary greatly; some are testing or training every day - others much less, monthly or even quarterly. That means we have environments that are idle for days or weeks at a time. We realized that if we group these environments together, we can get to the point where pods could be powered down for much longer periods of time.Â

We're working with Harness to get a little more intelligence here to help us be more efficient with this categorization, but we've already started the reorganization with the data we have in hand today.Â

What's Next?

For now, I'm focusing on getting as close to 100% coverage for Cloud AutoStopping rules as I can, since it has such a massive, ongoing positive impact on our bottom line. Then I'll start exploring other features more in-depth like Cloud Asset Governance and implementing recommendations.Â

It's been a great journey for us so far, and you can read more about it in our case study.Â

Editors Note: This is one of a new series of blogs that are focused on Harness Cloud Cost Management users are âArchitecting for Cost Savingsâ, highlighting their experiences where cloud architectural decisions and changes have positively impacted cloud costs.Â

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/harness-policy-as-code?utm_source=pmm-ff&utm_medium=leveraging-feature-flags-for-zero-downtime-database-migrations

Introducing Harness Policy as Code, Powered by OPA

We're excited to announce Harness Policy as Code, powered by Open Policy Agent (OPA), a centralized policy management and rules service that empowers enterprises to centrally define and monitor policies that are enforced across all delivery pipelines and processes. Harness Policy as Code helps organizations create and enforce policies on deployments, infrastructure, and more, providing developer velocity without sacrificing compliance and standards.

Harness Policy as Code is based on OPA, an easy-to-use, extensible solution for creating and enforcing policies across the entire stack. OPA is an open source project accepted by the Cloud Native Computing Foundation (CNCF) with wide adoption across numerous software delivery use cases. Policies are written as declarative code, so they are easy to understand and modify—from simple to complex use cases.

Harness Policy as Code integrates with CI, CD, and Feature Flags enforcing automated approvals, denials, and other advanced pipeline functionality. Check out our technical documentation to learn more.

Why We Need Policy Management in Software Delivery

As DevOps is adopted within an enterprise, typically one team creates and maintains software delivery and processes. That team has full control and visibility, as they are the creators of DevOps processes within the company. As more business units adopt DevOps within the company, that originating team's manual processes can create a bottleneck, which hampers innovation by limiting team autonomy and slowing down software delivery.

Â In an effort to remove the bottleneck and increase velocity, companies can give development teams more autonomy by allowing them to drive their own DevOps processes. That decentralization of process control can lead to more risks for the company.

When governance is decentralized, development teams can miss quality checks or approvals, introduce vulnerabilities, or break compliance. Organizations need to balance autonomy *and* governance, so they can empower teams with the confidence that they are adhering to all compliance standards and security policies â all without slowing down innovation.

Compliance becomes even more critical in regulated industries, such as financial services and healthcareânot only with enterprise standards, but with third-party regulations, like SOC2, PCI, and FedRamp. It is imperative that all software delivery pipelines meet compliance standards with full auditability; otherwise, the organization is at risk of failed audits, heavy fines, and reputational damage.Â

Centralized management and governance of policies across DevOps processes allow enterprises to define standards for the entire organization while enforcing compliance with regulations. Policies enable individual teams to have autonomy over their processes with oversight and guardrails in place to prevent them from straying from standards, ensuring secure and compliant software delivery.Â

Harness Policy as Code Features

Harness Policy as Code is a centralized policy management and rules service that leverages OPA to meet compliance requirements across software delivery. HPE enables organizations to centrally define and monitor policies that are enforced across all delivery pipelines and processes.Â

Policy as Code features for writing and enforcing policies include:

- A Policy Editor that enables developers to start writing policies-as-code quickly. With a library of policies to start from and a testing terminal, developers can try out policies on real inputs during development before enabling them.
- Policies that are configured to be automatically enforced on Harness processes (e.g. on Pipeline Run, on Feature Flag save).
- The ability to set severity, so a policy violation can issue a warning or throw an error to stop processes from continuing.
- An audit trail that can maintain a full history of policy evaluations with detailed outputs for audit and compliance.

With the release of Policy as Code, policies can now be enforced on CI and CD pipelines and Feature Flags.

Pipeline policies govern the requirements of delivery pipelines, and they can be automatically enforced when the pipeline is saved or triggered, or even in the middle of pipeline execution. Policies can enforce specific pipeline configuration, advanced access control use cases, runtime validation, and more. Here are some examples of what the Policy as Code can do:

- Require an approval step before deployment to production.
- Forbid use of Shell scripts in the pipeline.
- Only allow deployment to approved namespace.
- Only allow deployments from approved container registries.
- Validate test step outcome meets minimum threshold before allowing the pipeline to continue.

Policies for Feature Flags are enforced when the flag is updated or toggled on/off, enabling policies for adhering to standards, flag process, and hygiene. This includes:

- Only allowing creation of boolean flags.
- Enforcing flag naming conventions.
- Enforcing when creating a flag the default on and off values must both be false.
- Requiring a Feature Flag be enabled in QA before it can be turned on in Production.

Policy as Code centralizes and standardizes policy management across software delivery, allowing engineering leaders to empower dev teams to own their tools and practices while ensuring that everyone is following company standards for compliance and security. With guardrails in place, security vulnerabilities won't be introduced as development teams are writing their pipelines. Leaders can rest assured that compliance standards are being met, with full auditability of policies and failures, and they can find and report breaches as early as possible with shift-left governance.

Get Started

Check out our platform governance page to learn more about how Harness' modern approach to software delivery governance empowers teams with stable processes that don't slow down delivery, or request your personalized demo today.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Common Feature Flags Security Questions

Two of the scariest things in the world for anyone operational or security-minded are:

It's no surprise that feature flags, both as a concept and a specific solution we sell at Harness, can draw some extra scrutiny. They may *appear* to present risk on both of these vectors. But in this post, we'll take a look at how feature flags *aren't* a security or performance risk - and how they actually *improve* your security and performance posture as an organization.

One note before we proceed. In this article, I will reference aspects of Harness Feature Flags as a product more specifically than we often do on these blogs. This is not to exclusively promote our product, but rather to be able to provide clear and specific explanations. You will find that most feature flag products on the market do a respectable job of most of these same points, as the overall category of feature management tools has evolved responsibly and thoughtfully over time.

Customer Data

When using Feature Flags, you will end up targeting your users, regions, or tenants based on criteria exposed to the service via the SDKs (more on those in a bit). A risk here is that some of the information that could potentially be used may in fact be sensitive, or a violation of PII policies.

Because of this risk, we have drawn a line on not collecting any data proactively. While there may be some simplification value in looking for common user or framework objects and automatically passing them to Harness, we do not want to collect data without your explicit choice to send it to us.

So, when configuring the data you will use for flag targeting, **all data must be manually coded to be sent to us on your side**. If you don't send it, we won't have it.

SDKs

The SDKs are the crux of how any feature flag solution works. You install the SDKs into your code, they communicate with the upstream service, send user or server data for targeting, and receive rule configurations to determine what states of features are served.

Because these SDKs run in your code, we (and everyone else who builds them in our industry) treat them with great care. As we already covered, our SDKs do not proactively collect or communicate any data. Additionally, they are all open source so that they can be fully audited at any time.Â

Every feature flag vendor has slightly different ways their SDKs phone home, but all will follow one or two similar patterns. At Harness, our SDKs support both streaming and polling mode. Streaming mode is where you receive server-sent events proactively, from Harness, to provide real-time updates. Polling mode is favored when you want less connectivity, and instead the SDKs will fetch updates periodically - they don't require ongoing connection. Both are commonly used.

ResiliencyÂ

We often get asked: How do we ensure that we create a flag that isn't an upstream dependency for your application? What if Harness is down? What if Google Cloud or AWS is down? What if the communication is experiencing very high latency for a particular user session? It's important to know that integrating feature flags into your continuous delivery cycle will never result in your users seeing a broken experience.

Feature flag tools are generally architected with all of these concerns in mind. At Harness, we have 3 levels of resiliency that you can rely on:

- The SDKs themselves will cache evaluations (and cache rules for the server sdks), so they will always have a first-level fallback if there is a connectivity issue.
- When implementing your flag in the code, you will always have a hardcoded default that will be used if there is no cache AND no connectivity.
- We also provide a relay proxy that you can run to sit between Harness and your applications. It provides a full cache, limits necessary connectivity to Harness (only the proxy needs to talk to Harness), and can even be side-loaded.

When you combine these things, there is never a situation where your application will fail to resolve because of an impact to the feature flag service, as well as never a time where you cannot change a mission-critical flag, since the proxy can be side-loaded if upstream Harness cannot be reached.

Helping, Not Hurting

As you've seen, feature flag solutions - including ours - are designed with awareness of the risk they could inadvertently carry. Between OSS SDKs, caches, proxies, in-code defaults, and more, these risks are carefully addressed so that you can flag in confidence.

Given that, it's worth considering the ways that feature flags actually become a critical part of your security and reliability posture, rather than a risk *to* it.

- Instant kill switches in production for any changes, without requiring a rollback.
- Audit log of all changes for high observability.
- Robust governance and permission controls around who can make changes and how.
- No more devs or ops people running playbooks directly on the resources! Use flags to make critical processes visible and repeatable.

Conclusion

There's a lot more to know about the specifics of how feature flags are built for security and performance. This is only meant to provide an overview. But, hopefully, you can see the level of care taken to guarantee that feature flags make your application more secure and more performant, with the possibility of introducing risk designed out of the service.

For more information, you can always go to our documentation or send us a question. We are happy to answer any security, architecture or performance related questions you may have.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/feature-flags-policy-governance-tutorial?utm_source=pmm-ff&utm_medium=leveraging-feature-flags-for-zero-downtime-database-migrations

Feature Flags Policy Governance Tutorial

Harness Policy as Code, powered by Open Policy Agent (OPA), allows you to write policies as code to automatically govern flag configuration, usage, and process. Policies are applied on every flag creation, toggle, or modification, ensuring your feature flags always stay in a compliant state.

We are proud to be the first feature management solution to support this capability. Global governance policies for feature flags are a big win for engineering organizations that need to enforce standards at scale. They allow for guardrails to be put in place across all releases to ensure standards are met, and they also automate the process, so the developer experience doesn't change. Developers simply get error messages the way they're used to during the build and test phase.Â

But this post isn't about why you should do it â it's about how you do it. In this post, we go in depth and teach you all about the Feature

Flags policy governance: architecture, use cases, data available to utilize, and more.

Architecture

Here's a quick overview of how we ensure that your flags always remain in a compliant state, regardless of what changes a user makes or how they decide to make them.Â Â

Writing Policies

The Harness Policy Engine is based on OPA, an easy-to-use, extensible solution for creating and enforcing policies across the entire stack. OPA is an open-source project accepted by the Cloud Native Computing Foundation (CNCF) with wide adoption across numerous software delivery use cases within the CI/CD pipeline. Policies are written in Rego as declarative code, so they are easy to understand and modify â from simple to complex use cases.

You can find more information about writing Rego rules in the official OPA docs.Â

Once you're ready to get a policy out there, you can write your own and add it to Harness. We also provide some policies out of the box for common use cases, like enforcing naming conventions and ensuring proper promotion of code.

What Data is Available to Write Policies Against?

We know that policies not only relate to the changed entity in question, but can also depend on contextual data, such as who made the change and when. To support these use cases, we provide an ever growing collection of data to the policy engine that you can use when writing your policies. This includes:

- The full feature flag configuration for the created/updated flag. This includes flag name, description, variations, rules, flag state in every environment you have, and much more.Â This is in json format, which matches the format of our public API docs, so if you interact with your flags as code already, it will be very familiar.Â
- Which user made the change, what RBAC permissions they have, and which user groups they belong to.
- When the change was made.Â

Here's a truncated example of the metadata sent to the policy engine

Debugging Policies

Policies are written as code, and as anyone who writes code knows, there will inevitably be lots of edge cases or error scenarios you want to test while refining your policies. Having to create and configure flags in special ways just to test your policies isÂ frustrating and time consuming. For this reason, we have an integrated policy tester/debugger built right into the UI.Â

Using this debugger, you can go through a standard development process:

- **Develop:** Begin by writing your policies in the main text area. Add as many rules and helper functions as you'd like.
- **Test:** Using the testing terminal, you can get quick feedback on if the new policies you're building are having the desired effect. You can hit the "Select Input" button to populate the input field with real flag data from your own project, giving you reliable and realistic test scenarios for your use cases. You can then hit the "Test" button to run the policies and see for the given data input if it would succeed or fail. You can also manually edit this inputÂ data if you'd like to hand craft particular edge cases.
- **Debug:** Once your policies are being enforced there may be a time that a user isn't sure why a change was blocked. This can be a frustrating experience if you're only provided with a vague error message such as "change forbidden." Luckily, we provide all the tools for a user to view detailed information on exactly which policies failed, along with the ability to click on these policies and enter this debugger mode, viewing the policy and the exact input their change produced. This will help them verify if it's a valid rejection or if the policy itself needs to be modified going forward.

Use Cases

Harness Policy Engine supports a wide array of use cases. These range from simple sanity checks that description fields have been properly completed to complex corporate policies that prevent changes during blackout periods. Feature flag policies broadly fall into these two categories:

Flag Configuration

- Naming conventions, e.g. flag names must match jira ticket format
- Mandatory descriptions
- Only allowing boolean flags to be created (no multivariates)
- Banning certain functionality e.g. no prerequisite rules allowed
- Max number of specific target rules

Change ManagementÂ

- Flag cannot be enabled in production unless it is enabled in QA first
- Flag changes must be made via pipelines with an approval step
- No changes during certain time periods e.g. when tests are running or during certain mandated blackout periods
- Flags can't be enabled by the same user that created it

You can get as creative as you'd like around the rules you need to enforce. We've purposely not taken an opinionated UI wizard-based approach on how to create and combine these rules, so you're free to experiment, start small, and govern what matters to you.Â

Where Can I Try It Out?

To learn more about how Harness manages policies, check out our Policies Overview for Feature Flags documentation, or sign up for a free Feature Flags trial today.Â

â

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/case-studies/choice-hotels-reduces-deployment-effort>

Source URL: https://www.harness.io/blog/flag-pipelines-to-automate-feature-flag-governance-and-daily-management?utm_source=pmm-ff&utm_medium=release-management-with-feature-flag?utm_source=pmm-ff&utm_medium=leveraging-feature-flags-for-zero-downtime-database-migrations

Announcing Flag Pipelines to Automate Feature Flag Governance and Daily Management

When we launched Harness Feature Flags, we were excited about the opportunity to leverage other capabilities across the Harness platform to create better-than-the-sum-of-its-parts value. One of the first things we did was add support for feature release pipelines to Feature Flags, which enabled users to automate software release workflows and policy enforcement.

Today, we're extending how you can use pipelines with the launch of a new concept, Flag Pipelines. Pipelines have always let you build

powerful workflows and automation for doing releases with feature flags, and now development teams can run pipelines automatically on every flag change to enforce governance and control. That's right — flags can have individual pipelines that run every time a flag state is changed. That's a game-changer for automating the day-to-day usage of flags and making sure that every change meets organizational standards.

Essentially, you can now guarantee that any time a flag's state is changed, it goes through a predefined workflow. This can be something as simple as requiring an approval every time a sensitive feature is toggled. Or, it can be something more complex, such as requiring an approval, making an update to Jira and ServiceNow, and sending notifications to customer support and managers.

Governance in the Day-To-Day Flag Lifecycle

Flag pipelines let you build a simple pipeline and assign it to a flag. From that point on, that pipeline will run automatically every time the flag changes, no matter how it changes. This means that whether you change it via the UI, API, or by updating the YAML via Git, you'll see this pipeline triggered.

This is great addition to feature management when your team wants to do things like:

- Require an approval from a manager role, or a PM, on any flag change in the production environment
- Send a Slack message on every flag change
- Update a Jira ticket being used to track feature development steps
- Execute a custom script associated with the flag every time the flag changes

Manual Process Misses the Point Of Feature Flags

Normally, you'd use pipelines with feature flags to build automated release workflows, such as incrementing a flag to 10% of your audience with an approval every 23 hours. And you can still do that for your one-off or specifically-modeled release scenarios. Flag Pipelines let you automate and enforce the things that you want to have happen *every time*, not just when you have a specific one-time scenario.Â

For teams and broader organizations, this feature enhancement provides a broader level of control over how feature flags are handled and managed day-to-day. Previously, users would have had to create an organizational process that required all of the users to run a stated pipeline every time that a flag needed to be changed. This poses a few problems:

It's not that teams don't try to do these things the right way, rather that things slip, people and processes change, and it's inconvenient to follow numerous extra steps for something that should be simple. Instead, what if users could use flags in a way that's more closely aligned with their actual needs, and the release process for flag changes is automatic? By flipping the script, devs aren't slowed down, and the business makes sure that things are done the right way every time.

How to Use Feature Flag Pipelines

â

To set up your flag pipelines, just select any flag and select the new "Flag Pipelines" tab. From there, you can choose or create the pipeline that you want to use.

Because flag pipelines are meant for the subset of workflows that you want to execute every time, not all pipelines can be used as flag pipelines. Flag pipelines must:

- Not have continuous integration/continuous development (CI/CD) steps
- Only have one flag change, not multiple
- Have the flag step set to runtime inputs.Â

Once you have a suitable pipeline and have it selected inside of your Flag Pipelines settings within a flag, you're all set. That pipeline will now automatically run every time that the flag is changed within that environment.

Find more details about how it works in the Harness docs.

Get Started with Harness Feature Flags

Flag Pipelines continue our work in the pursuit of turning feature flags into a true part of your software delivery process. Allowing more sophisticated control of feature flags brings increased governance and feature flag automation to your organization. It's a big step forward in evolving feature flags into a key business process in software delivery.

Ready to get started? Request a demo or sign up for free forever. We'll continue to expand on this in the future and would love your feedback on how you'd like to see this story evolve.Â

Happy developing!

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/what-are-feature-flags?utm_source=pmm-ff&utm_medium=harnessblog-elevating-reliability

What are Feature Flags and How Do We Use Them?

Building software is a great experiment. Many times we are required to do what has not been done before; this is core to innovation work. However, a lot of the iteration and experimenting tends to happen before software is shipped. That's where the concept of test in production comes into play. Building something from scratch or even jumping into an older project mid-stream requires lots of iteration.Â Â

Once we deploy, for many software engineers, the experimentation on the existing tends to drop off and onwards to the next set of features in the backlog. Product owners/managers typically collect feedback to help prioritize the next set of features. From a software engineering perspective this can feel a little âchicken or egg-ishâ - without a feature actually deployed in a production environment to get feedback, how do we if know we are on the mark?

Letâs illustrate this with an example. Customers are asking for an updated UI view for their dashboards; they want something that surfaces the most relevant information faster. Now that the product team has collected those requirements, itâs up to engineering to figure out how to best build it. There might be a few potential solutions:

With some potential solutions decided, now itâs up to the engineering team to decide which of the three to implement. And before a customer ever sees it, they have to choose 1 of 3, build out the MVP over the next few sprints, and then ship it. This is the first time theyâll get any feedback, and if itâs not the right solution then it has to be built again!

Now, imagine if this new feature cycle could be improved for product development teams. If youâve been paying attention to developer tools over the last few years, you might have seen a new category popping up: feature flags. Feature flags as a concept were born to increase development velocity and to solve this very problem through practices like progressive delivery.

While feature flags as a mature category of hosted tooling is new, feature flags - and what they solve - have been around a little longer. Letâs take a look at what feature flags have grown into, and how to use them.

The Story of Feature Flags

Feature flags, at their core, are the ability to wrap different versions of your code in conditional statements that you can turn on and off at will. You may have previously thought of them as app configs, or even part of your companyâs rules engine.Â

With Harness Feature Flags, we've taken these concepts you may be familiar with and taken them even further - adding more sophisticated rules, governance, user management, an intuitive UI, and more. A

Feature Flags lets you separate feature release from code deployment, build more effective strategies for rolling things out and testing with your users, and have more insight than ever into how changes impact your application performance.

Ultimately, feature flags give engineering teams the ability to deliver more features, with less risk. For leaders, it's a great way to improve engineering velocity and developer experience, and to reduce the risk associated with feature development. And for developers, it's a great way to work more efficiently, with less stress - think fewer late nights before a deployment, and fewer firefights post-deploy. A

What Are Feature Flags (or Feature Toggles)?

Feature flags are a way that developers can conditionally turn certain sections of their code on or off. You can think of feature flags as extending Continuous Delivery into Continuous Deployment - a way to put changes into production behind a flag and turn them on in a controlled way later (or hide and remove them in the same way).

But you can also think of feature flags as a new way to think about building and releasing applications - by clearly defining features and components that can be changed and toggled on and off individually, you allow for experimentation, controlled rollouts to different users, as well letting more people (like PMs, sales, and support) turn things on and off for customers.

Together, this results in giving more power and control to developers to create more and better features faster, while at the same time ensuring that engineering is not the bottleneck for the business in having to make changes for individual customers or ensuring a perfect production deployment for every new feature or feature update.

If we want to simplify even further, feature flags essentially create private swim lanes for developers where they can ship a feature directly to customers and then have control over who sees it, get feedback, and turn it on and off as needed. It's like a light switch (a pretty complex one)!

Who Uses Feature Flags?

People oftentimes consider feature flags either an engineering tool, or a tool for product managers. The reality is, it's both. Flags can help software development and DevOps teams lower their overhead and increase their velocity, and they can help product managers better control releases, coordinating launch timings and creating a feedback loop more effectively.

Research indicates that 95% of engineering leaders want to implement a feature flag solution for themselves. This comes on the back of 97% of leaders saying they're under pressure to deliver faster, and 65% of whom say they find it difficult to achieve that velocity increase in a safe manner. It's abundantly clear that engineering teams are likely the biggest beneficiaries of implementing and using a feature flag solution, and indeed, they tend to be the primary users. And when they do implement a feature flag solution, they end up delivering 66% more features per application per year!

Feature flags are great for users across the organization beyond product and engineering. For sales and support teams, feature flags provide a way for them to directly manage betas and new features for their customers, and to track down who's using what in case of any issues. And for management, flags can provide new forms of visibility into what's happening in development and how new features are being tested with users.

On a fun fact kind of note and since we're in the "who uses feature flags" section, it's cool to note that big companies like Netflix, Flickr, Google, Reddit, and more use feature flags! They're definitely worth looking into and absolutely are an important part of the software delivery lifecycle.

Why You Should Consider Feature Flags

Whether or not the use of feature flags is important to your software delivery lifecycle comes down to what it is that you need to accomplish. While feature flags are a great tool to have for any team, they are particularly useful if you have these problems:

Problems in the Business

- The risk associated with new feature releases ends up slowing down developers since they have to thoroughly validate design and implementation before releasing.
- There are often multiple possible versions of how a new feature will look and function, and it's up to the developer to choose one (not always the right one) and implement it.
- The decision of when to release new software features is controlled by developers instead of the business.
- Sunsetting old features introduces risk to newer features that might have dependencies on the old code.
- It's all or nothing - a feature is rolled out to everyone at the same time instead of doing a feature rollout incrementally for verification purposes.
- When teams are beholden to engineering release cycles (e.g. once a month), multiple features are released simultaneously, introducing complexity and risk that can result in deployment war rooms, large rollbacks, and dissatisfied customers.

Problems in Engineering

- Releasing a feature takes too much coordination and time across stakeholders.
- Merging new features, or managing separate feature branches for changes, is cumbersome and complex.

- You want to test changes to see how they impact your application and end-users before rolling them out more widely.
- You want to provide access to new features, or to test changes with certain portions of the user base sooner - by location, by customer plans, by customers opting into beta testing, or by any other criteria or logic you can think up.
- You want to reduce the risk of rolling out new code by having instant kill switches that don't require rollbacks or new deployments to fix a disrupted environment.

To simplify, we can consider that feature flags are a great solution if you're trying to do the following things:

- Increase engineering velocity and ship more features.
- Decrease feature development risk by A/B testing multiple solutions with customers.
- Reduce the risk and toil associated with production failures and rollbacks.
- Decouple engineering deployments from feature unveilings to customers.

Different Types of Feature Flags/Toggles

It's natural to want to put all feature flag use cases in a single bucket, but it's more helpful to instead view flags themselves as a means to an end and as having categories of use cases.

Here are some examples of the different kinds of flags that could be used in a system:

- **Release features** - Using flags to let the right people decide who to turn a feature on for, and when, without complex engineering deployment coordination required.
- **Experiment** - Using flags to learn something about how a change impacts things. This can be a technical experiment, such as how a new UI impacts server load, or it can be a user experiment such as how a new button impacts conversion.
- **Ops flags** - Letting you have permanent control around key parts of your app so you can turn them on and off as necessary without a full new deployment cycle. Add in RBAC, and this is a robust way to control who has access to modify your production application.
- **Control access** - Using flags as a way to manage betas, early adopter list, trial access, and more.

Implementing Feature Flags

Implementing feature flags can seem really simple at first, but it turns out that like with any system, the devil is in the details. Building an awesome feature flag solution in-house certainly has its perks, but it can be deceptively difficult. In particular, we've seen three distinct sets of issues arise with feature flag solutions as an organization's implementation matures.

When a feature flag solution is **nonexistent or immature** and an organization is standing it up for the first time, they tend to run into issues with finding value in the tool quickly. Especially with the need for such a solution and the pressure to make it happen quickly, it's critical that value is demonstrated quickly, otherwise it may end up leaving a sour taste. Within that, there's a dichotomy of architecting for eventual scale versus getting an MVP out the door to prove value. This often results in complex design or architecture that isn't scalable and only works for small, targeted use cases.

As feature flag solutions enter a **middle stage of maturity**, or as they **start to scale**, the problem is quite obvious: scaling is hard. Especially if built in-house, ensuring that the tool can be used by multiple users and multiple teams for a variety of use cases, all while maintaining a common set of best practices and building on a single infrastructure can be incredibly daunting. And as this is being figured out and teams have code in production, it can quickly result in stale code or issues keeping production tidy when code changes occur. Add onto that the need to integrate neatly into everyone's workflow and you've got a behemoth of a problem to solve.

Building on that is the scenario we often see with **highly mature** feature flag implementations, where they primarily end up as separate systems from existing software delivery or CI/CD pipelines that teams now have to learn how to use, and to figure out how to port their existing processes or pipelines into the feature flag solution. At this stage, there's the obvious cost of maintenance and future development to keep up with new changes or requirements. Technical debt is bound to build up by this point, and it can become unmanageable without a dedicated team - and perhaps even its own codebase.

Sure, 95% of engineering leaders may want a feature flag implementation, but as it turns out, it's not so easy to build something that works well over time.

Build or Buy: Feature Flag Management

You may be thinking, "Feature flags seem pretty simple, can't we just build this ourselves?"

The real answer? It's complicated.

We've seen in most cases that an internally-built tool will end up causing a lot of overhead and putting a very low ceiling on the benefits you can receive from feature flags. It may be good for your first couple of flags, but it can be hard to make an internal system scalable, performant, and robust for all the things flags can do as your organization gets more and more comfortable using them - and more reliant on them.

For a deeper look at the build vs buy conversation, you can check out an article we wrote earlier this month, Feature Flags: Should I Build or Buy?

Why Use Harness for Feature Flag Management

Harness provides a feature flag platform made for getting you up and running quickly while being able to scale to any level. We make it easy to get started and quickly see the benefits of feature flags. With Harness Feature Flags, that means you can get your first flag out the door in minutes by following three steps:

- Create a free trial account.
- Create a new flag in just 3 clicks.
- Add one of our SDKs to your code in a few minutes.

And, you've got a flag working!

If that's not quite clear, then it's worth knowing that the fundamental way flags work in an environment like Harness is that SDKs, added to your application as a library, will receive updates on flag statuses to evaluate with your users (you should choose between real-time streaming or pulling updates on intervals). These SDKs are secure, collecting no data you don't want to share, and they are designed for performance and stability.Â

Harness Feature Flags is fast, secure, and focused on helping teams get the most out of feature flags - from your first initial use all the way to global governance, sophisticated reusable templates, and developer-first experiences.

And if you're wondering how we stack up against LaunchDarkly, Optimizely, Cloudbees Feature Management, Split, and other FF tools - those pages are being worked on right now and we'll update this post with the links when they're live!Â

If you'd like a little more information on our architecture and on how to get started, you can check out the feature flag documentation or request a demo.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/introducing-harness-internal-developer-portal-beta-release>

Introducing Harness Internal Developer Portal - Beta release

ââ

Discovery

In October 2022, when we launched the Backstage-Harness integration for our customers - we started talking to customers who were building their Internal Developer Portal on their own. Months into their journey, many of them complained about the struggles a Platform Engineering team goes through when building an IDP ground up which also includes self managing the hosting and operations aspects. While the trend suggests that IDPs will be everywhere, we couldn't ignore the challenges our customers were facing in this journey. A successful IDP requires an investment of 3-5 engineers to get an early Proof of Concept, even when using a popular open source project like Backstage. It takes significant effort to set up basic enterprise features like service onboarding pipeline, user and group management, access control, secrets management, audit trails, and more. With Harness IDP, these features are a core part of our platform and are shipped out of the box.

Today, we are proud to go Beta with a handful of customers who need an IDP but find it challenging, costly and time consuming to manage it themselves or find a good alternative. Our IDP is powered by Backstage, the de-facto platform for building developer portals. We want to thank the community for building the software and we are committed to upstream contributions as we move forward.

But let's back up a second, what is an IDP really? What problems does it solve? And what features does Harness IDP have?

Problems companies are facing today

Let's start with the problems. The pain starts on day 1 with developer onboarding where it takes too much time and too many tickets to have developers commit, build, test code while also learning the process to get their features to production. It doesn't get any easier even for more experienced engineers. In today's cloud-based, decentralized, microservices environments, it's challenging to manually track all software dependencies and things that need to happen every time changes need to be rolled out from dev-to-test-to-prod. Most organizations use a complex collection of infrastructure (cloud VM, Kubernetes, databases and more), frameworks (code, API, serverless and more) and tools (CI/CD, security scanners, monitoring and more) across different layers of the software stack. This internal maze of technologies creates unnecessary overhead, duplicates effort and hurts developer productivity. As a result, developers end up doing nonessential work to manage the cognitive overhead. Developer productivity and happiness requires removing roadblocks, and tool complexity is one of those roadblocks.

Now let's take a look at how we are planning to solve these problems using Harness IDP.

Streamlining new service onboarding

How much time does it take for a developer in your company to create a new service? We have heard various answers from days, weeks to months. Lack of automation and standards result in fragmentation of technology choices. Inter-team dependencies make it really difficult to innovate fast.

In Harness IDP, as a developer, you can create a new backend service, API, or a website by submitting a few details as configured by your platform engineering. On the other hand, as a platform engineer, you can orchestrate the onboarding of services by creating pipelines in the Harness Pipeline Studio.

Developers focus on what they do best, which is writing features, while platform engineers focus on creating software templates, automating processes, and enforcing standards.

A catalog of all software components

Imagine a world where you wake up in the morning, get to a single page which can tell you everything you need to know about your running software - its builds, deployments, alerts, errors, etc. And if you want to use a service owned by another team, you get to see its dependencies, documentation, API reference, owner information and much more!

Technical Documentation and Search

With Harness IDP, all your technical documentation written in markdown is made available right alongside the software homepage in the catalog. The docs-like-code approach ensures that docs are treated just like code, they live alongside the code and are updated in the same Pull Request by engineers. It also helps other developers find the documentation without going anywhere.

And with the Search functionality available not only on documentation, but across all the software components you have registered, you can quickly find what you need. This is so much better than a rumor-driven-development isn't it? Usually that involves asking a question on the #engineering-help channel and waiting for the handful few to notice and answer.

Extensibility through plugins

The number one reason Backstage is the best developer portal platform out there is its plugin architecture. There are hundreds of plugins available in the marketplace which integrate Backstage with third party providers and enhance the portal. We have built our own plugins and integrated with a curated list of Backstage plugins which you can enable and use.

Sign up for Beta

Harness IDP is currently available to a limited set of customers. We want to work closely with our customers and evolve in more use-cases suited for a Developer Portal. To get started, send an email to idp-interest@harness.io for a demo and we'll get you started. To learn more, checkout the documentation at developer hub.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/canaries-and-feature-flags?utm_source=pmm-ff&utm_medium=release-management-with-feature-flags

Progressive Delivery: Canaries and Feature Flags

If you're familiar with using modern CD tools, such as Harness CD, you are probably also familiar with canary releases. Canaries let you serve a new version of your app to a subset of your total traffic in the production environment, learn how it's working, and then cut more traffic over to the new version safely in a progressive way. Feature Flags also let you turn something on for a subset of your users, then ramp it up over time in a safe and progressive way! So, a question we get a lot as a company that sells great software for continuous delivery with canaries and Feature Flags is: what's the difference?

Not Versus. AND.Â

The first thing we always say is, this is a great question and may take a few minutes to unpack. And, the second this is - it's not feature flags vs canaries, it's feature flags *and* canaries.Â

One way to think of CD + Feature Flags in the software delivery process is as a ladder, where CD builds on CI and Feature Flags builds on CD. Using canaries for your deployment strategy and feature flags for your release strategy is additive - and we recommend it.

When to Canary

Canary releases help you change artifacts in your environment. They make sure the new version of your application is running safely, handling traffic, and doing what you need to see metrics-wise before you fully rely on it.Â

Canaries are great for backend and infrastructure-focused changes - as well as to even get the features into prod that you want to start testing - that every deployment contains. The goal of a canary is always to get the new version to 100%, and ideally, to do so as fast as you can.

Canaries, though, are not about testing new features, running beta programs, or having toggles around changes that you will want to target in precise ways or have to persist for longer periods of time. This is where feature flags come in.

When to Flag

Feature Flags are focused *after* your deployment. Feature Flags let you do two primary things:

- Turn on changes, or new features, for a specific subset of your audience (such as location, beta group, or individual users) in a controlled way. This is perfect for beta testing, getting feedback on new features, as well as testing the impact of changes without deploys and rollbacks.
- Maintain operation toggles around things you may want to turn on or off occasionally - such as turning off caching during peak times. Feature Flags are an easier, more visible, and auditable way to have these kinds of operational toggles.

Feature Flags allow for precise, ad-hoc targeting around any dimension you want - target your users by location, plan, individual customer name - anything you can think of that you may want to use for targeting changes.

Conclusion

With Canaries + Feature Flags, you can roll your deployments out and verify that the new code is working, producing good metrics, and not creating havoc in a progressive way - and, on top of that, roll features and changes out to targeted users, or cohorts of users, over time as part of your beta process, experimentation process, or selective feature enablement needs.

This gets you true progressive delivery, both progressive on the deploy and after the deploy, with full control over what you release, when, and to whom.

If you'd like to learn more about Feature Flags, we wrote a great post about what they are and how to use them - it's a great starting point in your FF journey! If you haven't signed up to use Harness yet but want to get started, you can easily sign up for free forever or request a demo. Happy developing!

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/feature-flags-pipelines?utm_source=pmm-ff&utm_medium=leveraging-feature-flags-for-zero-downtime-database-migrations

Announcing: Feature Release Workflows With Feature Flags Pipelines

We're thrilled to announce the introduction of feature release workflows for Harness Feature Flags, called Pipelines. Users of Feature Flags can now standardize the process and steps required to release a feature, and layer in automation to make feature release as simple as hitting "Go".

Feature Release Management Is A Chore - Schedules, Approvals, Monitoring - Oh My!

The most common implementation of a feature flags solution is one that is built in-house. In fact, most of the time people don't even realize they're building a feature flags solution! And that makes sense. There are a number of point problems that can be solved with feature flags, and engineering teams want to solve those. But before they know it, it's turned into another internal tool to maintain and upgrade.

What makes this difficult is two-sided:

Process is important, especially as organizations seek to maximize the value of the feature flag solution that they have built or bought. Even for those who have bought a tool, it's critical that they are able to create a standardized release process that anyone can follow, and that guarantees that any feature release has gone through the appropriate steps. And let's not forget about being able to audit releases for compliance purposes - or because something went wrong and a root cause needs to be identified.

When we look at all of these things together, it becomes obvious that all of the requirements are interrelated. Let's be clear about the problems that need to be addressed. And who knows, maybe there's a way to solve them all at once!

- Integration into the release process: monitoring, approvals, scheduling, etc.
- Standardization of feature releases across teams, products, projects, etc.
- Ability to view audit trails for compliance and root cause analysis.
- Creating a common framework that anyone can work off of.

You can imagine that being able to do this would be a massive burden taken off the shoulders of anyone wanting to use a feature flag solution, whether they're a developer or a manager. If you were able to solve these problems, you'd essentially make feature releases non-eventful.

And maybe you could even automate it?

Enter Pipelines: Feature Release Management Workflows

To solve these problems, we built Pipelines for Harness Feature Flags. Pipelines take all of the requirements and problems to be addressed above and put them into a single construct that is significantly easier to comprehend and use than the traditional hodge-podge of tools, logs, scripts, and emails.

Building Workflows for Common Release Requirements

Check out this simple pipeline below. Even though there are only three steps, it clearly outlines what you want to happen: turn on the feature; get an approval; and then release to the rest of the beta segment of your customers.

If this were your actual release process, you could easily take this pipeline and turn it into a reusable template. Now, every time you have a new feature, you could just click a button and feel good that all steps will be taken care of. Of course, these can get more complex as well. Let's take a look at what that could be:

Part of what makes Pipelines in Harness Feature Flags really useful is the deep library of integrations into tools you and your team probably use. Now we can expand the simple use case above where only an approval is required. The workflow could include updating Jira tickets, sending Slack updates, verifying logs or health metrics. Basically, anything you might need in order to release a feature.

Now you've taken all these disparate activities and rolled them into a single, manageable workflow. If something goes wrong, you can easily access the audit logs to see what changed, who changed it, when, and so on. You also have a visual pipeline, should you choose to use it, that you can use to quickly spotcheck the progress of a feature rollout or verify that the right steps are included.

Automation of Feature Releases

It's not just about eliminating the toil and headache associated with feature releases by creating templated workflows. You might have picked up on the nature of Pipelines, which is that it can be automated.

Once a Pipeline is created to meet the needs of your workflow, you can associate any feature and simply hit the proverbial "Go" button. From there, the Harness Pipeline will take care of going through each of the steps that you've laid out. It can automatically handle rolling out the feature, pinging people for approvals, making changes in Jira, checking health metrics - you name it. Of course, if there are issues that require attention or failures in the process, you will be alerted. This leaves you to do things other than babysit the release pipeline, and you only need to be involved if something goes wrong.

In the near future, we're also looking to integrate automated health metric verification that will take this a step further. Imagine you want to roll out a feature to a subset of your users, verify that things work as expected, or that the right business metrics are moving, and then progressively roll that out to larger portions of your user base. Or, if something goes wrong, you want to take some pre-specified remediation action. All of these things will be possible, meaning that you will truly be able to simply declare what you want, and the

system will take care of making it happen.

Added Value of the Harness Platform

As an end-to-end software delivery platform, it wouldn't make much sense *not* to have Harness Feature Flags have some interplay with the rest of the software delivery process. We commonly see that feature flags are an amplification method for CI/CD and in many ways are the third step of the CI/CD pipeline.

Harness is actually the only feature flag solution on the market today that natively integrates with CI/CD. Why does that matter? And what value does that provide to a product development or ops organization? And why do executives care about it too?

Let's start with a topic we've already covered - automation. The same Pipeline feature in Harness Feature Flags is also available in Harness Continuous Integration and Harness Continuous Delivery. Yes - you get a unified pipeline. Suddenly, you go from disconnected CI/CD and feature flagging to a single pipeline that shares context and can be automated from build through to individual feature delivery to end users. Can you imagine what that could mean for your software delivery process? But that's not the only reason it matters.

Analytics, governance, compliance, risk - all are other really important reasons to have CI/CD and feature flags linked up. By having shared context across the entire SDLC, you're able to do all of the above better:

- Create a canonical analytics dashboard to track software delivery metrics at each stage, and the interplay between them.
- Enforce org-wide governance policies for software delivery, and ensure that governance standards are always met.
- Leverage audit trails across every stage of software delivery to see exactly what's happening and where, and simplify external and internal audits.
- De-risk release at any stage because you're not worried about inter-tool integrations or support and critical context being lost that can impact the business or customers.

Of course, you can go into more detail about any of these topics on our blog, which speaks at length about each of these topics.

How to Get Started

If you're already a Harness Feature Flags user, you can head over to the product and you'll see a new menu item labeled "Pipelines" that you can click and get started building your own!

If you haven't signed up to use Harness yet but want to get started, you can easily sign up for a free trial. Happy developing!

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

[Documentation](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[Case studies](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[We want to hear from you](#)

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/software-development-life-cycle?utm_source=pmm-ff&utm_medium=release-management-with-feature-flags

Understanding the Phases of the Software Development Life Cycle

While the technologies, methods, and perspectives about building high-performance and scalable software services have changed, the responsibilities and actions have not. The Software Development Life Cycle (SDLC) is a series of important phases defined for teams producing and delivering high-quality software. This blog post will discuss the SDLC and its stages in greater detail.

What is the Software Development Life Cycle?

The Software Development Life Cycle refers to the phases of work involved in producing software applications. Each phase corresponds to a role or responsibility that contributors to the software must understand, manage, and optimize to deliver their software services with speed and performance. These stages of work include:

- Requirements Gathering,
- Software Design,
- Software Development,
- Test and Integration,
- Deployment,
- Operationalization and Maintenance.

Let's discuss each stage of work in greater detail.

The Phases of the SDLC

Requirements Gathering

In this stage of work, the team identifies, gathers, and defines current problems, requirements, requests, and customer expectations related to the software application or service.

Some activities related to the requirements gathering phase can involve creating software specifications, creating a detailed plan, documentation, issue tracking, and project or product planning, including allocating the correct resources.

Defining software or product requirements gives teams the foresight and context needed to deliver and produce their software solutions.

Software Design

In this design phase of work, the team makes software design decisions regarding the architecture and make of the software solution. This can involve creating design documents, coding guidelines, and discussing the tools, practices, runtimes, or frameworks that will help the team meet the software requirement specification and goals defined in the requirements gathering phase.

Software Development

In this stage of work, teams build the software solutions based on the design decisions made. Here, teams meet the goals and outcomes set during the software requirements gathering phase by implementing the solution.

The development process may involve teams of people, new technologies, and unexpected challenges; however, the development teams typically work with tech leads and product or project managers to unblock the process, make decisions, or provide support. This stage of work ends once teams have packaged and built their code.

Test and Integration

In this phase of work, a software implementation is packaged and tested to assure quality. Testing or quality assurance ensures the solutions implemented pass the standard for quality and performance. This can involve unit testing, performing integration and end-to-end tests, verification/validation, and reporting or identifying bugs or defects in the software solution.

Deployment

In this stage of work, the software is deployed into a production environment. The work gathered, designed, developed, and tested is shared with the consumers and users of the software service. This process involves provisioning infrastructure within an on-premise or cloud provider and defining a software deployment strategy for delivering the changes to a customer.

If you want to learn more, we break down deployment strategies and discuss when and why to use each type.

Operationalization and Maintenance

In this stage of work, the software is operationalized to ensure there are no issues or incidents related to the deployment. This stage of work can involve reviewing, understanding, and monitoring network settings, infrastructure configurations, and performance of application services in production. This process can involve incident resolution or management in the course of any issues or changes made to impact a customer or user base.

Why Does the SDLC Matter for Software Delivery?

Now that we know more about the work associated with each part of the SDLC, we can discuss why it matters and how it applies to how we deliver software today. For many organizations, a challenge is delivering better software faster. Understanding the SDLC allows teams to understand what it takes to deliver features or code changes to customers.

The reality for many developers is the need to wait months or years to see code changes make it out to users, coupled with a lack of visibility, communication, and collaboration during the process. Organizations and teams that have the capability to deploy on-demand and in a self-service fashion empower their teams to continue doing their best work.

Our Continuous Delivery 2020 Insights report found that engineering teams spend on average \$109,000 annually to deploy and deliver their software applications. Production deployment efforts result, on average, to 25 hours of engineering effort.

The SDLC offers perspective into the distinct work phases needed to produce software. Understanding this work allows teams to avoid the delivery issues by creating and owning checks and balances early on in our development and delivery life cycle.

This is also about incorporating feedback and insights during the software development process to continuously deliver value in a repeatable, quick, and sustained fashion.

Understanding the SDLC allows teams to also improve their DevOps performance, which can be measured through DORA metrics.

Examples of a Software Development Life Cycle Model

Computer scientists, software development practitioners, and leaders have always aimed to deliver better software faster. Over time, several models (such as waterfall, spiral, Agile) emerged to describe and represent the SDLC processes and manage the level of development complexity as demand, tools, processes, and mindsets changed.

Waterfall Model

Perhaps one of the earliest models used to represent the process for delivering software is the waterfall model, developed in 1956. In this model, a chain of linear sequential phases represents the activities for delivering software.

Each phase depends on the delivery and execution of the previous phase, where each phase contained a set of tasks. This model originated from the manufacturing and construction industries and was adopted for knowledge or project-based creative work.

The drawback of this model is its dependencies. Since progress flows in one direction, there was little room to adjust to newly-discovered constraints, requirements, and problems once design decisions were made and implementation began.

Delivering all the software would also lead to increased costs as changes in requirements would lead to major redesigns, redevelopment, and retesting. These drawbacks lead to modified waterfall models, such as the Sashimi (Waterfall with Overlapping Phases), Waterfall with Subprojects, and Waterfall with Risk Reduction.

Iterative and Incremental Model

In response to the perceived problems with the waterfall model, organizations such as the United States Department of Defense released statements, such as the MIL-STD-498, encouraging *Iterative and Incremental Development*. This led to the term that would combine both iterative design and incremental development patterns.

In this model, the software is developed and delivered through repeated cycles of smaller portions of work. This model allows for software teams to take advantage of learnings and insights made earlier on in the process from developing and using the software system. Teams at each iteration of work make the necessary design modifications and additional functional capabilities.

NASA's Project Mercury is an example of the early usage of the Iterative and Incremental Development model. The success of the project later led to further adoption as Project Mercury's engineers took to other teams and projects.

Although the origins of the iterative model stem from the software industry, many hardware and embedded software development efforts are now using iterative and incremental techniques (for example, companies such as SpaceX and Rocket Lab).

The Evolution of Process Models

Following the success of Iterative and Incremental software development methods, other software development methods emerged to leverage more project management principles and development practices.

The spiral model is one risk-driven development model that encourages project teams to deliver based on unique project risks, leveraging

one or many elements of other delivery methodologies. In the 1990s, the Agile manifesto led to the adoption and popularity of the Agile model and subsequent Agile methodologies.Â

Today, we have the DevOps Life Cycle, representing the SDLC and our goals to continuously deliver software value as a cross-functional team.

How you develop or deliver is up to you. Whatâs important is knowing what is involved in the process. This blog post discussed the software development lifecycle and the SDLC models that emerged to allow us to build and share our software services today. Good luck to everyone who manages or is a part of a project life cycle!Â

Expand Your Knowledge of SDLC Tools

Domain experts, architects, systems developers, engineers, and leaders all have a stake in delivering great software. If youâd like to learn more about delivering software value, we recommend getting more in tune with the state of the Kubernetes ecosystem.

We have created an in-depth eBook that you can download *for free*, which you will learn about Kubernetes architecture, options for running Kubernetes across a host of environments, key open source projects in the Kubernetes ecosystem, adoption patterns of cloud-native infrastructure and tools, and more.

Download your free copy of Kubernetes eBook now.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/what-are-feature-flags?utm_source=pmm-ff&utm_medium=release-management-with-feature-flags

What are Feature Flags and How Do We Use Them?

Building software is a great experiment. Many times we are required to do what has not been done before; this is core to innovation work. However, a lot of the iteration and experimenting tends to happen before software is shipped. That's where the concept of test in production comes into play. Building something from scratch or even jumping into an older project mid-stream requires lots of iteration.Â Â

Once we deploy, for many software engineers, the experimentation on the existing tends to drop off and onwards to the next set of features in the backlog. Product owners/managers typically collect feedback to help prioritize the next set of features. From a software engineering perspective this can feel a little âchicken or egg-ishâ - without a feature actually deployed in a production environment to get feedback, how do we if know we are on the mark?

Letâs illustrate this with an example. Customers are asking for an updated UI view for their dashboards; they want something that surfaces the most relevant information faster. Now that the product team has collected those requirements, itâs up to engineering to figure out how to best build it. There might be a few potential solutions:

With some potential solutions decided, now itâs up to the engineering team to decide which of the three to implement. And before a customer ever sees it, they have to choose 1 of 3, build out the MVP over the next few sprints, and then ship it. This is the first time theyâll get any feedback, and if itâs not the right solution then it has to be built again!

Now, imagine if this new feature cycle could be improved for product development teams. If youâve been paying attention to developer tools over the last few years, you might have seen a new category popping up: feature flags. Feature flags as a concept were born to increase development velocity and to solve this very problem through practices like progressive delivery.

While feature flags as a mature category of hosted tooling is new, feature flags - and what they solve - have been around a little longer. Letâs take a look at what feature flags have grown into, and how to use them.

The Story of Feature Flags

Feature flags, at their core, are the ability to wrap different versions of your code in conditional statements that you can turn on and off at will. You may have previously thought of them as app configs, or even part of your companyâs rules engine.Â

With Harness Feature Flags, weâve taken these concepts you may be familiar with and taken them even further - adding more sophisticated rules, governance, user management, an intuitive UI, and more.Â

Feature Flags lets you separate feature release from code deployment, build more effective strategies for rolling things out and testing with your users, and have more insight than ever into how changes impact your application performance.

Ultimately, feature flags give engineering teams the ability to deliver more features, with less risk. For leaders, itâs a great way to improve engineering velocity and developer experience, and to reduce the risk associated with feature development. And for developers, itâs a great way to work more efficiently, with less stress - think fewer late nights before a deployment, and fewer firefights post-deploy.Â

What Are Feature Flags (or Feature Toggles)?

Feature flags are a way that developers can conditionally turn certain sections of their code on or off. You can think of feature flags as extending Continuous Delivery into Continuous Deployment - a way to put changes into production behind a flag and turn them on in a controlled way later (or hide and remove them in the same way).

But you can also think of feature flags as a new way to think about building and releasing applications - by clearly defining features and components that can be changed and toggled on and off individually, you allow for experimentation, controlled rollouts to different users, as well letting more people (like PMs, sales, and support) turn things on and off for customers.

Together, this results in giving more power and control to developers to create more and better features faster, while at the same time ensuring that engineering is not the bottleneck for the business in having to make changes for individual customers or ensuring a perfect production deployment for every new feature or feature update.

If we want to simplify even further, feature flags essentially create private swim lanes for developers where they can ship a feature directly to customers and then have control over who sees it, get feedback, and turn it on and off as needed. Itâs like a light switch (a pretty complex one)!

Who Uses Feature Flags?

People oftentimes consider feature flags either an engineering tool, or a tool for product managers. The reality is, itâs both. Flags can help software development and DevOps teams lower their overhead and increase their velocity, and they can help product managers better control releases, coordinating launch timings and creating a feedback loop more effectively.

Research indicates that 95% of engineering leaders want to implement a feature flag solution for themselves. This comes on the back of 97% of leaders saying theyâre under pressure to deliver faster, and 65% of whom say they find it difficult to achieve that velocity increase in a safe manner. Itâs abundantly clear that engineering teams are likely the biggest beneficiaries of implementing and using a feature flag solution, and indeed, they tend to be the primary users. And when they do implement a feature flag solution, they end up delivering 66% more features per application per year!

Feature flags are great for users across the organization beyond product and engineering. For sales and support teams, feature flags provide a way for them to directly manage betas and new features for their customers, and to track down whoâs using what in case of any issues. And for management, flags can provide new forms of visibility into whatâs happening in development and how new features are being tested with users.

On a âfun factâ kind of note and since weâre in the âwho uses feature flagsâ section, itâs cool to note that big companies like Netflix,

Flickr, Google, Reddit, and more use feature flags! They're definitely worth looking into and absolutely are an important part of the software delivery lifecycle.

Why You Should Consider Feature Flags

Whether or not the use of feature flags is important to your software delivery lifecycle comes down to what it is that you need to accomplish. While feature flags are a great tool to have for any team, they are particularly useful if you have these problems:

Problems in the Business

- The risk associated with new feature releases ends up slowing down developers since they have to thoroughly validate design and implementation before releasing.
- There are often multiple possible versions of how a new feature will look and function, and it's up to the developer to choose one (not always the right one) and implement it.
- The decision of when to release new software features is controlled by developers instead of the business.
- Sunsetting old features introduces risk to newer features that might have dependencies on the old code.
- It's all or nothing - a feature is rolled out to everyone at the same time instead of doing a feature rollout incrementally for verification purposes.
- When teams are beholden to engineering release cycles (e.g. once a month), multiple features are released simultaneously, introducing complexity and risk that can result in deployment war rooms, large rollbacks, and dissatisfied customers.

Problems in Engineering

- Releasing a feature takes too much coordination and time across stakeholders.
- Merging new features, or managing separate feature branches for changes, is cumbersome and complex.
- You want to test changes to see how they impact your application and end-users before rolling them out more widely.
- You want to provide access to new features, or to test changes with certain portions of the user base sooner - by location, by customer plans, by customers opting into beta testing, or by any other criteria or logic you can think up.
- You want to reduce the risk of rolling out new code by having instant kill switches that don't require rollbacks or new deployments to fix a disrupted environment.

To simplify, we can consider that feature flags are a great solution if you're trying to do the following things:

- Increase engineering velocity and ship more features.
- Decrease feature development risk by A/B testing multiple solutions with customers.
- Reduce the risk and toil associated with production failures and rollbacks.
- Decouple engineering deployments from feature unveilings to customers.

Different Types of Feature Flags/Toggles

It's natural to want to put all feature flag use cases in a single bucket, but it's more helpful to instead view flags themselves as a means to an end and as having categories of use cases.

Here are some examples of the different kinds of flags that could be used in a system:

- **Release features** - Using flags to let the right people decide who to turn a feature on for, and when, without complex engineering deployment coordination required.
- **Experiment** - Using flags to learn something about how a change impacts things. This can be a technical experiment, such as how a new UI impacts server load, or it can be a user experiment such as how a new button impacts conversion.
- **Ops flags** - Letting you have permanent control around key parts of your app so you can turn them on and off as necessary without a full new deployment cycle. Add in RBAC, and this is a robust way to control who has access to modify your production application.
- **Control access** - Using flags as a way to manage betas, early adopter list, trial access, and more.

Implementing Feature Flags

Implementing feature flags can seem really simple at first, but it turns out that like with any system, the devil is in the details. Building an awesome feature flag solution in-house certainly has its perks, but it can be deceptively difficult. In particular, we've seen three distinct sets of issues arise with feature flag solutions as an organization's implementation matures.

When a feature flag solution is **nonexistent or immature** and an organization is standing it up for the first time, they tend to run into issues with finding value in the tool quickly. Especially with the need for such a solution and the pressure to make it happen quickly, it's critical that value is demonstrated quickly, otherwise it may end up leaving a sour taste. Within that, there's a dichotomy of architecting for eventual scale versus getting an MVP out the door to prove value. This often results in complex design or architecture that isn't scalable and only works for small, targeted use cases.

As feature flag solutions enter a **middle stage of maturity**, or as they **start to scale**, the problem is quite obvious: scaling is hard. Especially if built in-house, ensuring that the tool can be used by multiple users and multiple teams for a variety of use cases, all while maintaining a common set of best practices and building on a single infrastructure can be incredibly daunting. And as this is being figured out and teams have code in production, it can quickly result in stale code or issues keeping production tidy when code changes occur. Add onto that the need to integrate neatly into everyone's workflow and you've got a behemoth of a problem to solve.

Building on that is the scenario we often see with **highly mature** feature flag implementations, where they primarily end up as separate systems from existing software delivery or CI/CD pipelines that teams now have to learn how to use, and to figure out how to port their existing processes or pipelines into the feature flag solution. At this stage, there's the obvious cost of maintenance and future development to keep up with new changes or requirements. Technical debt is bound to build up by this point, and it can become unmanageable without a dedicated team - and perhaps even its own codebase.

Sure, 95% of engineering leaders may want a feature flag implementation, but as it turns out, it's not so easy to build something that works well over time.

Build or Buy: Feature Flag Management

You may be thinking, "Feature flags seem pretty simple, can't we just build this ourselves?"

The real answer? It's complicated.

We've seen in most cases that an internally-built tool will end up causing a lot of overhead and putting a very low ceiling on the benefits you can receive from feature flags. It may be good for your first couple of flags, but it can be hard to make an internal system scalable, performant, and robust for all the things flags can do as your organization gets more and more comfortable using them - and more reliant on them.

For a deeper look at the build vs buy conversation, you can check out an article we wrote earlier this month, Feature Flags: Should I Build or Buy?

Why Use Harness for Feature Flag Management

Harness provides a feature flag platform made for getting you up and running quickly while being able to scale to any level. We make it easy to get started and quickly see the benefits of feature flags. With Harness Feature Flags, that means you can get your first flag out the door in minutes by following three steps:

- Create a free trial account.
- Create a new flag in just 3 clicks.
- Add one of our SDKs to your code in a few minutes.

And, you've got a flag working!

If that's not quite clear, then it's worth knowing that the fundamental way flags work in an environment like Harness is that SDKs, added to your application as a library, will receive updates on flag statuses to evaluate with your users (you should choose between real-time streaming or pulling updates on intervals). These SDKs are secure, collecting no data you don't want to share, and they are designed for performance and stability.

Harness Feature Flags is fast, secure, and focused on helping teams get the most out of feature flags - from your first initial use all the way to global governance, sophisticated reusable templates, and developer-first experiences.

And if you're wondering how we stack up against LaunchDarkly, Optimizely, Cloudbees Feature Management, Split, and other FF tools - those pages are being worked on right now and we'll update this post with the links when they're live!

If you'd like a little more information on our architecture and on how to get started, you can check out the feature flag documentation or request a demo.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

[Documentation](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer

happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?
Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/introduction-service-level-objective-slo-management?utm_source=pmm-ff&utm_medium=harnessblog-elevating-reliability

Introduction to Service Level Objective (SLO) Management

One of the main responsibilities of the site reliability engineering (SRE) team is to ensure application reliability, or that applications are available and functioning as expected for their end users. One way to measure application reliability is through a service level objective (SLO).¹

In the world of software delivery, there is a never-ending struggle between delivering new software features quickly and providing application services that customers can depend on when they need them. Now, it's up to teams in both IT and development to support software delivery best practices with minimal disruption to customers. SLO management is intended to bring balance to software velocity and reliability needs. In this article, we'll discuss SLO management and how using service level objectives is a key component of site reliability management.

Service Level Terminology

SLO management has its own set of terms and acronyms that are important to know.

Service Level Objective (SLO)

A service level objective (SLO) is a target level of service that an organization aims to provide to its customers or users. Effective SLOs ensure that an organization is delivering a high level of service to its customers and meeting business objectives. Specifically, these are goals within an organization related to the reliability of an application service. An internal SLO is not communicated outside of the organization and is not legally binding.

SLO Management

Service Level Objective (SLO) management involves setting and maintaining the target levels of service, as well as monitoring and measuring the actual system performance to ensure that it meets target levels. Management can involve setting targets for uptime, response time, error rates, and other performance metrics. SRE teams then monitor and analyze these metrics to identify and address any issues that may arise.²

Service Level Indicator (SLI)

A service level indicator (SLIs) provides insights into the health of a service. It is the core metric used to indicate if specific service level indicators are met.³

Service Level Agreement (SLA)

Service level agreements (SLAs) are legal agreements communicated to customers by business and legal teams that explain the implications if an expected service fails to meet the promised targets. For example, for system availability, if uptime is below the promised level, the service provider may be subject to paying fines or penalties to paying customers.

Error Budget

An error budget is a tool that helps the SRE and development teams work in tandem to control release velocity by ensuring that reliability targets are achieved. The error budget is an allowance for SLO violations that can accumulate over a certain timeframe for your service before your customers are impacted. Failures are inevitable when you constantly change your systems. Therefore, normalizing failure as a part of the process helps teams balance innovation with the risk of SLA violation.⁴

The Relationship Between the SLA and SLO Targets

There does not have to be a direct relationship between SLAs and SLOs, but there often is. Many reliability teams set their SLOs based on existing SLAs as a quick and easy starting point. For example, if the SLA for a service is 99.9% availability, then a good SLO for the same service could be 99.95%. The SLO is more restrictive than the SLA because the SLO should be strict enough to preemptively help the team avoid violating the SLA.

How to Define and Manage Service Level Objectives and Service Level Indicators

Setting Up SLIs

Reliability targets must take into account the business needs. SLI metrics will be provided by your monitoring or observability solutions. SLIs should reflect the customer experience of the core application or a particular service, not necessarily every individual service. Too many metrics will only hinder your team's ability to focus.

Common SLIs include the four golden signals: latency, availability, throughput, and error rate. You should consider how to implement each of these. For example, latency (also known as response time) can be measured for all transactions flowing through an application or for a subset of the most important transactions (e.g., login, submit payment, add to cart, etc).Â

You'll need to pick a metric that provides a meaningful representation of your customers' experiences and also define a threshold for that metric.

Setting Up SLOs

SLO definition is a collaborative process driven by the reliability team. SLOs act as the principal driver of decision-making, which enables you to discover the right balance between velocity and reliability. Breaching the SLO can potentially initiate activities that ultimately put pressure on engineering to stabilize the service before releasing new features.

Each of your SLIs will have an associated SLO (possibly having multiple SLIs per SLO). The SLO you define is the percentage of requests that should comply with the threshold you defined when you set up the SLI.Â

Calculating Service Level Objectives

Calculated SLO = # of requests that meet the defined threshold / total number of requests * 100

Example:

2454 login requests took less than 100ms during a 1 minute period

2522 total login requests during that 1 minute period

â

Calculated SLO = 2454/2522*100

Calculated SLO = 97.3% for that 1 minute period

Setting up Error Budgets

An error budget is measured in minutes over a defined time period. For example, if your SLO is set to 99.5% then the associated error budget (room for failure) per week is 50 minutes. Once the error budget is exhausted, teams should cease deploying new features and focus on service quality and reliability.Â

Overall, an error budget is a useful tool for ensuring that a system is reliable and available to users as much as possible, while also allowing for the flexibility to make necessary changes and improvements.

Notifications and Alerts: Common Mistakes

If there are reliability problems brewing in your environment, you want to know about them. Carefully consider what you want to be alerted on. Too many alerts or too few alerts might mean your team is missing some important reliability issues.

What's most important is to know when reliability targets are not being met, indicating poor customer experience. You probably don't want to get an alert on a one-minute violation once a month, because it's not meaningful to the end user if it only happened during that narrow timeframe.

When SLO violations occur too frequently, the error budget burn rate goes up, and the remaining error budget decreases. The decrease in error budget is the time to send a notification via Slack, email, etc.

Perfection in SLO Management is Not Necessary (Nor Desired)

You might be tempted to set 100% reliability as an objective, but perfection is impossible. It'd simply mean that you choose not to make any changes in production, which is definitely not a wise business decision. Setting measurable and concrete reliability targets that allow for an appropriate rate of new feature delivery will result in happy customers. Finding this balance is central to creating the caliber of

software experiences that your business needs to compete.Â

Establishing SLOs and creating error budgets can be a long journey, but the results are well worth the investment. Ready to learn more about SLO management? Request a demo of the Harness SRE solution, Service Reliability Management.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/delivering-reliability-through-sre-practices?utm_source=pmm-ff&utm_medium=harnessblog-elevating-reliability

Delivering Reliability Through SRE Practices

When we talk about continuous delivery, the part that gets left out is how we deliver sustainably and repeatedly. Reliability is everyoneâs responsibility, but sometimes this statement can feel like it is at odds with innovating and delivering features quickly.Â Site Reliability Engineering (SRE) has been a hot topic this year as more guides around the role were shared, and more people stepped into this role than ever before.

When discussing Software Delivery, itâs crucial to discuss SRE and introduce software design implementation and maintenance practices to achieve both innovation and reliability work successfully. This blog shares some popular SRE practices and how to apply them to enable continuous delivery.

Being Available Before, After, and During an Incident.

Site Reliability Engineers are responsible for being available during an incident that means responding, explaining, and retrospecting different aspects of an incident that occurs within an organization. This can involve reviewing production workflows, alert criteria or triggers, and human processes surrounding a deployment.Â

One way SREs can better respond and sustain software delivery is by following an on-call playbook. An on-call playbook is a guide on how to respond to an event. Itâs often a template that automatically generates a ticket with information regarding the severity of the trigger alert, debugging suggestions and actions to mitigate the impact of an incident.Â

Another common practice is to ensure that post-mortems promote continuous improvement, influence product management, and increase visibility into action items captured during a retrospective. Itâs common for SREs to follow up and publish with post-mortem

improvements following an incident.Â

Being on call is not easy. It comes down to thinking about the people, process, and technology involved in our delivery process. When reviewing incidents, common areas to focus on include monitoring and metrics, TOIL and pager load, and the service application itself (whether the incident was caused by a new or an existing bug.)

Defining How Code Gets into Production

The release engineering process defines how code gets into production. For SREs, this can mean defining processes, reviewing artifacts, and owning CI/CD pipelines. Release engineering is about minimizing risk, improving tempo, and automating manual processes that prevent software delivery from being repeatable.Â

One practice to consider in release engineering is introducing Canary deployments or forms of progressive delivery. Progressive delivery shifts a subset of user traffic from an existing service to a newly deployed service. It's worth considering this capability around release engineering. Canary deployments might be useful for critical services where an incident could spell the end of a business. Many solutions exist in the space that help integrate canary deployment capabilities into the release engineering process.

Managing Reliability

The third area that site reliability engineering focuses on involves budgeting errors and owning the reliability of an application. Practices include setting SLAs, measuring latency or performance, and improving the monitoring of an application. It's common for an SRE to block production releases if an application team violates a specific error threshold.Â

There can be many indicators of poor reliability, including low availability and poor health delivery metrics. Common delivery health metrics include mean time to restore, change failure rate, lead time to production, and deployment frequency. When considering metrics look at change over time, the impact of technology solutions, and these differences across teams and services to track and validate specific outcomes.

Successful software is ready, stable, agile, and valuable for users. It's fairly common for an SRE to flag behaviors or outputs that violate any of these four key focus areas. This can involve blocked production releases if an application team violates a specific error threshold. SREs provide an opportunity for app teams to review what is needed to improve an application's performance and reliability.Â

Site Reliability Engineering as the Catalyst for Better Software Development and Operations.

There's an important opportunity that site reliability engineers and site reliability engineering practices have on the continuous delivery lifecycle. Whether it's discovering incidents, preventing them, or resolving them, there are many ways to support your delivery life cycle sustainably. This blog post shares some of the practices that your team or organization can employ to better support software delivery. If you'd like to learn more about how to better support your software delivery, try Harness for free.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/platform-engineering-next-phase-developer-experience-efficiency>

Platform Engineering is the Next Phase of Developer Experience and Efficiency

When Kubernetes became popular around 2015, the developer experience began to change for the worse. Prior to the widespread use of Kubernetes, developers wrote code and then handed it off to site reliability engineers (SREs) for deployment. But with Kubernetes, developers started owning the entire software delivery process, from writing code to deploying it, increasing their responsibilities and making the developer experience significantly less enjoyable.Â

Things just kept going downhill for the developer experience from there. With the increase in the use of microservices, SaaS applications, and multi-cloud environments, developers faced an increasingly chaotic ecosystem as they built and shipped their code. The increase in the number of DevOps tools made things even messier and developers had to contend with a new issue: tool sprawl. With numerous platforms and tools, collaborating became more difficult, and even onboarding new employees became more tedious and unnecessarily complex.

These challenges have led to the rise of platform engineering, a practice that is focused on designing, building, and maintaining the tools to help developers drive efficiency. Let's discuss what platform engineering is and how your team can get started.

Creating A Better Developer Experience with Platform EngineeringÂ

The goal of platform engineering is to give ownership of the infrastructure to a dedicated platform team. Developers can access the tools and services they need more easily, without having to see the complexity of their development environment. In addition, developers can focus more on shipping features and less on navigating a complicated environment.Â

According to Paul Delory, VP Analyst at Gartner: ÂPlatform engineering emerged in response to the increasing complexity of modern software architectures. Today, non-expert end users are often asked to operate an assembly of complicated arcane services. To help end users, and reduce friction for the valuable work they do, forward-thinking companies have begun to build operating platforms that sit between the end user and the backing services on which they rely.Â

Internal Developer Portals: The Main Component of Platform EngineeringÂ

The most essential part of platform engineering is an internal developer portal, which acts as a single pane of glass for the entire development infrastructure. According to Gartner, ÂBy 2025, 75% of organizations with platform teams will provide self-service developer portals to improve developer experience and accelerate product innovation.Â

The developer portal allows developers to find all the tools they need in one place, and it provides teams with a highly personalized view of the services and software components they manage. An internal developer portal allows developers to create new software components (such as services, pipelines, or resources) in seconds following the best practices established by the platform engineering team.Â

A developer portal also allows teams to explore all of the assets in their company, includingÂ APIs, technical docs, and services, supporting greater visibility inside their companyâs tooling and enabling greater collaboration. Enhanced collaboration is especially

important for larger engineering teams, such as those with 500 or more engineers, which can face significant challenges with collaboration.

Creating a Platform Engineering Team

To get started with platform engineering, you'll need a dedicated team. This team may start off as the DevOps team initially. The platform engineering team provides standards, templates, and best practices, along with guard rails such as security practices and permissions. The team can begin to plan the internal developer portal by performing research including talking to developers to try to understand their workflows and challenges.

A platform engineering team does not have to be large. For a team of 1,000 engineers, for example, the platform engineering organization can be as few as five people. This small team can help improve developer velocity metrics such as time to create a tenth pull request for new developers, or the number of open pull requests a developer has.

Platform Engineering: The Next Big Thing for Engineers

The platform engineering approach is the latest way to improve the developer experience, which ultimately leads to greater efficiency. At Harness, our mission is to enable every software engineering team in the world to deliver code reliably, efficiently and quickly to their users. Finding and retaining good developers is a priority for any company that creates software, so naturally, making sure that your developers have the best possible experience working with your technology is a top priority.

To learn more about how your organization can take DevOps to the next level with our eBook, Create an Exceptional Developer Experience. To see how the Harness platform supports the developer experience, request a demo!

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

[Documentation](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[Case studies](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[We want to hear from you](#)

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

[Sign up for our monthly newsletter](#)

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/feature-flags-pipelines?utm_source=pmm-ff&utm_medium=release-management-with-feature-flags

Announcing: Feature Release Workflows With Feature Flags Pipelines

We're thrilled to announce the introduction of feature release workflows for Harness Feature Flags, called Pipelines. Users of Feature

Flags can now standardize the process and steps required to release a feature, and layer in automation to make feature release as simple as hitting a Go button.

Feature Release Management Is A Chore - Schedules, Approvals, Monitoring - Oh My!

The most common implementation of a feature flags solution is one that is built in-house. In fact, most of the time people don't even realize they're building a feature flags solution! And that makes sense. There are a number of point problems that can be solved with feature flags, and engineering teams want to solve those. But before they know it, it's turned into another internal tool to maintain and upgrade.

What makes this difficult is two-sided:

Process is important, especially as organizations seek to maximize the value of the feature flag solution that they have built or bought. Even for those who have bought a tool, it's critical that they are able to create a standardized release process that anyone can follow, and that guarantees that any feature release has gone through the appropriate steps. And let's not forget about being able to audit releases for compliance purposes - or because something went wrong and a root cause needs to be identified.

When we look at all of these things together, it becomes obvious that all of the requirements are interrelated. Let's be clear about the problems that need to be addressed. And who knows, maybe there's a way to solve them all at once!

- Integration into the release process: monitoring, approvals, scheduling, etc.
- Standardization of feature releases across teams, products, projects, etc.
- Ability to view audit trails for compliance and root cause analysis.
- Creating a common framework that anyone can work off of.

You can imagine that being able to do this would be a massive burden taken off the shoulders of anyone wanting to use a feature flag solution, whether they're a developer or a manager. If you were able to solve these problems, you'd essentially make feature releases non-eventful.

And maybe you could even automate it?

Enter Pipelines: Feature Release Management Workflows

To solve these problems, we built Pipelines for Harness Feature Flags. Pipelines take all of the requirements and problems to be addressed above and put them into a single construct that is significantly easier to comprehend and use than the traditional hodge-podge of tools, logs, scripts, and emails.

Building Workflows for Common Release Requirements

Check out this simple pipeline below. Even though there are only three steps, it clearly outlines what you want to happen: turn on the feature; get an approval; and then release to the rest of the beta segment of your customers.

If this were your actual release process, you could easily take this pipeline and turn it into a reusable template. Now, every time you have a new feature, you could just click a button and feel good that all steps will be taken care of. Of course, these can get more complex as well. Let's take a look at what that could be:

Part of what makes Pipelines in Harness Feature Flags really useful is the deep library of integrations into tools you and your team probably use. Now we can expand the simple use case above where only an approval is required. The workflow could include updating Jira tickets, sending Slack updates, verifying logs or health metrics. Basically, anything you might need in order to release a feature.

Now you've taken all these disparate activities and rolled them into a single, manageable workflow. If something goes wrong, you can easily access the audit logs to see what changed, who changed it, when, and so on. You also have a visual pipeline, should you choose to use it, that you can use to quickly spotcheck the progress of a feature rollout or verify that the right steps are included.

Automation of Feature Releases

It's not just about eliminating the toil and headache associated with feature releases by creating templated workflows. You might have picked up on the nature of Pipelines, which is that it can be automated.

Once a Pipeline is created to meet the needs of your workflow, you can associate any feature and simply hit the proverbial Go button. From there, the Harness Pipeline will take care of going through each of the steps that you've laid out. It can automatically handle rolling out the feature, pinging people for approvals, making changes in Jira, checking health metrics - you name it. Of course, if there are issues that require attention or failures in the process, you will be alerted. This leaves you to do things other than babysit the release pipeline, and you only need to be involved if something goes wrong.

In the near future, we're also looking to integrate automated health metric verification that will take this a step further. Imagine you want to roll out a feature to a subset of your users, verify that things work as expected, or that the right business metrics are moving, and then progressively roll that out to larger portions of your user base. Or, if something goes wrong, you want to take some pre-specified remediation action. All of these things will be possible, meaning that you will truly be able to simply declare what you want, and the system will take care of making it happen.

Added Value of the Harness Platform

As an end-to-end software delivery platform, it wouldn't make much sense *not* to have Harness Feature Flags have some interplay with the rest of the software delivery process. We commonly see that feature flags are an amplification method for CI/CD and in many ways are the third step of the CI/CD pipeline.

Harness is actually the only feature flag solution on the market today that natively integrates with CI/CD. Why does that matter? And what value does that provide to a product development or ops organization? And why do executives care about it too?

Let's start with a topic we've already covered - automation. The same Pipeline feature in Harness Feature Flags is also available in Harness Continuous Integration and Harness Continuous Delivery. Yes - you get a unified pipeline. Suddenly, you go from disconnected CI/CD and feature flagging to a single pipeline that shares context and can be automated from build through to individual feature delivery to end users. Can you imagine what that could mean for your software delivery process? But that's not the only reason it matters.

Analytics, governance, compliance, risk - all are other really important reasons to have CI/CD and feature flags linked up. By having shared context across the entire SDLC, you're able to do all of the above better:

- Create a canonical analytics dashboard to track software delivery metrics at each stage, and the interplay between them.
- Enforce org-wide governance policies for software delivery, and ensure that governance standards are always met.
- Leverage audit trails across every stage of software delivery to see exactly what's happening and where, and simplify external and internal audits.
- De-risk release at any stage because you're not worried about inter-tool integrations or support and critical context being lost that can impact the business or customers.

Of course, you can go into more detail about any of these topics on our blog, which speaks at length about each of these topics.

How to Get Started

If you're already a Harness Feature Flags user, you can head over to the product and you'll see a new menu item labeled "Pipelines" that you can click and get started building your own!

If you haven't signed up to use Harness yet but want to get started, you can easily sign up for a free trial. Happy developing!

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/flag-pipelines-to-automate-feature-flag-governance-and-daily-management?utm_source=pmm-ff&utm_medium=release-management-with-feature-flags

Announcing Flag Pipelines to Automate Feature Flag Governance and Daily Management

When we launched Harness Feature Flags, we were excited about the opportunity to leverage other capabilities across the Harness platform to create better-than-the-sum-of-its-parts value. One of the first things we did was add support for feature release pipelines to Feature Flags, which enabled users to automate software release workflows and policy enforcement.

Today, we're extending how you can use pipelines with the launch of a new concept, Flag Pipelines. Pipelines have always let you build powerful workflows and automation for doing releases with feature flags, and now development teams can run pipelines automatically on every flag change to enforce governance and control. That's right — flags can have individual pipelines that run every time a flag state is changed. That's a game-changer for automating the day-to-day usage of flags and making sure that every change meets organizational standards.

Essentially, you can now guarantee that any time a flag's state is changed, it goes through a predefined workflow. This can be something as simple as requiring an approval every time a sensitive feature is toggled. Or, it can be something more complex, such as requiring an approval, making an update to Jira and ServiceNow, and sending notifications to customer support and managers.

Governance in the Day-To-Day Flag Lifecycle

Flag pipelines let you build a simple pipeline and assign it to a flag. From that point on, that pipeline will run automatically every time the flag changes, no matter how it changes. This means that whether you change it via the UI, API, or by updating the YAML via Git, you'll see this pipeline triggered.

This is great addition to feature management when your team wants to do things like:

- Require an approval from a manager role, or a PM, on any flag change in the production environment
- Send a Slack message on every flag change
- Update a Jira ticket being used to track feature development steps
- Execute a custom script associated with the flag every time the flag changes

Manual Process Misses the Point Of Feature Flags

Normally, you'd use pipelines with feature flags to build automated release workflows, such as incrementing a flag to 10% of your audience with an approval every 23 hours. And you can still do that for your one-off or specifically-modeled release scenarios. Flag Pipelines let you automate and enforce the things that you want to have happen *every time*, not just when you have a specific one-time scenario.

For teams and broader organizations, this feature enhancement provides a broader level of control over how feature flags are handled and managed day-to-day. Previously, users would have had to create an organizational process that required all of the users to run a stated pipeline every time that a flag needed to be changed. This poses a few problems:

It's not that teams don't try to do these things the right way, rather that things slip, people and processes change, and it's inconvenient to follow numerous extra steps for something that should be simple. Instead, what if users could use flags in a way that's more closely aligned with their actual needs, and the release process for flag changes is automatic? By flipping the script, devs aren't slowed down, and the business makes sure that things are done the right way every time.

How to Use Feature Flag Pipelines

â

To set up your flag pipelines, just select any flag and select the new "Flag Pipelines" tab. From there, you can choose or create the pipeline that you want to use.

Because flag pipelines are meant for the subset of workflows that you want to execute every time, not all pipelines can be used as flag pipelines. Flag pipelines must:

- Not have continuous integration/continuous development (CI/CD) steps
- Only have one flag change, not multiple
- Have the flag step set to runtime inputs

Once you have a suitable pipeline and have it selected inside of your Flag Pipelines settings within a flag, you're all set. That pipeline will now automatically run every time that the flag is changed within that environment.

Find more details about how it works in the Harness docs.

Get Started with Harness Feature Flags

Flag Pipelines continue our work in the pursuit of turning feature flags into a true part of your software delivery process. Allowing more sophisticated control of feature flags brings increased governance and feature flag automation to your organization. It's a big step forward in evolving feature flags into a key business process in software delivery.

Ready to get started? Request a demo or sign up for free forever. We'll continue to expand on this in the future and would love your feedback on how you'd like to see this story evolve.Â

Happy developing!

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/feature-flags-policy-governance-tutorial?utm_source=pmm-ff&utm_medium=release-management-with-feature-flags

Feature Flags Policy Governance Tutorial

Harness Policy as Code, powered by Open Policy Agent (OPA), allows you to write policies as code to automatically govern flag configuration, usage, and process. Policies are applied on every flag creation, toggle, or modification, ensuring your feature flags always stay in a compliant state.

We are proud to be the first feature management solution to support this capability. Global governance policies for feature flags are a big win for engineering organizations that need to enforce standards at scale. They allow for guardrails to be put in place across all releases to ensure standards are met, and they also automate the process, so the developer experience doesn't change. Developers simply get error messages the way they're used to during the build and test phase.Â

But this post isn't about why you should do it â it's about how you do it. In this post, we go in depth and teach you all about the Feature Flags policy governance: architecture, use cases, data available to utilize, and more.

Architecture

Here's a quick overview of how we ensure that your flags always remain in a compliant state, regardless of what changes a user makes or how they decide to make them.Â

Writing Policies

The Harness Policy Engine is based on OPA, an easy-to-use, extensible solution for creating and enforcing policies across the entire stack. OPA is an open-source project accepted by the Cloud Native Computing Foundation (CNCF) with wide adoption across numerous software delivery use cases within the CI/CD pipeline. Policies are written in Rego as declarative code, so they are easy to understand and modify â from simple to complex use cases.

You can find more information about writing Rego rules in the official OPA docs.Â

Once you're ready to get a policy out there, you can write your own and add it to Harness. We also provide some policies out of the box for common use cases, like enforcing naming conventions and ensuring proper promotion of code.

What Data is Available to Write Policies Against?

We know that policies not only relate to the changed entity in question, but can also depend on contextual data, such as who made the change and when. To support these use cases, we provide an ever growing collection of data to the policy engine that you can use when writing your policies. This includes:

- The full feature flag configuration for the created/updated flag. This includes flag name, description, variations, rules, flag state in every environment you have, and much more.Â This is in json format, which matches the format of our public API docs, so if you interact with your flags as code already, it will be very familiar.Â
- Which user made the change, what RBAC permissions they have, and which user groups they belong to.
- When the change was made.Â

Here's a truncated example of the metadata sent to the policy engine

Debugging Policies

Policies are written as code, and as anyone who writes code knows, there will inevitably be lots of edge cases or error scenarios you want to test while refining your policies. Having to create and configure flags in special ways just to test your policies isÂ frustrating and time consuming. For this reason, we have an integrated policy tester/debugger built right into the UI.Â

Using this debugger, you can go through a standard development process:

- **Develop:** Begin by writing your policies in the main text area. Add as many rules and helper functions as you'd like.
- **Test:** Using the testing terminal, you can get quick feedback on if the new policies you're building are having the desired effect. You can hit the "Select Input" button to populate the input field with real flag data from your own project, giving you reliable and realistic test scenarios for your use cases. You can then hit the "Test" button to run the policies and see for the given data input if it would succeed or fail. You can also manually edit this inputÂ data if you'd like to hand craft particular edge cases.
- **Debug:** Once your policies are being enforced there may be a time that a user isn't sure why a change was blocked. This can be a frustrating experience if you're only provided with a vague error message such as "change forbidden." Luckily, we provide all the tools for a user to view detailed information on exactly which policies failed, along with the ability to click on these policies and enter this debugger mode, viewing the policy and the exact input their change produced. This will help them verify if it's a valid rejection or if the policy itself needs to be modified going forward.

Use Cases

Harness Policy Engine supports a wide array of use cases. These range from simple sanity checks that description fields have been properly completed to complex corporate policies that prevent changes during blackout periods. Feature flag policies broadly fall into these two categories:

Flag Configuration

- Naming conventions, e.g. flag names must match jira ticket format
- Mandatory descriptions
- Only allowing boolean flags to be created (no multivariates)
- Banning certain functionality e.g. no prerequisite rules allowed
- Max number of specific target rules

Change ManagementÂ

- Flag cannot be enabled in production unless it is enabled in QA first
- Flag changes must be made via pipelines with an approval step
- No changes during certain time periods e.g. when tests are running or during certain mandated blackout periods
- Flags can't be enabled by the same user that created it

You can get as creative as you'd like around the rules you need to enforce. We've purposely not taken an opinionated UI wizard-based approach on how to create and combine these rules, so you're free to experiment, start small, and govern what matters to you.Â

Where Can I Try It Out?

To learn more about how Harness manages policies, check out our Policies Overview for Feature Flags documentation, or sign up for a free Feature Flags trial today.Â

â

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/harness-certified-experts-continuous-delivery-developer-certification>

Introducing Harness Certified Experts: Continuous Delivery Developer Certification

Keeping your skills sharp is a continuous journey for engineers. Delivering software in modern environments requires a myriad of skills from source to deployment, especially for the tools developers use. Today, we're pleased to announce a new Harness platform certification series, Harness Certified Experts.

Powered by the Harness Developer Hub, learning material for Harness Certified Experts will be readily available for our users to help credential seekers re-enforce or acquire their Harness knowledge. Becoming a Harness Certified Expert will test your knowledge and validate your skills in the software delivery world.

Our first certification, the Harness Certified Expert - Continuous Delivery Developer, is now live and ready!.Â

Open Learning Paths for Exams

Our goal is to have material readily available for all levels of exam takers to learn with as the capstone for your learning journey. A combination of tutorials and documentation will help you prepare for an exam. Even if you do not take the exam, you can still find a good learning path for your own pace on the Harness Developer Hub. Once you are ready, you can take the certification exams.Â

â

Taking Your Harness Certified Expert Exam

Registering for the exams is simple. Head to Harness Developer Hub, select the exam you would like to take, and click Register for Exam.Â

Our first new certification, Harness Certified Expert - Continuous Delivery Developer, will be offered for free for all to take. After reviewing the study path and exam objectives, sign up for the exam, which you can take immediately. The exam should take about 90 minutes to complete.Â

Once you have successfully completed the exam, a Credly Badge will be issued to you in a few days. With your first Harness Certification under your belt, this is just the beginning!

Harness Certifications: Just Getting StartedÂ

We plan to roll out certifications at multiple levels to all Harness modules. These levels are Developer, Administrator, and Architect levels.

The upper level exams will have practical, interactive portions where you will need to interact with Harness. Make sure to stay tuned on the Harness Developer Hubâs Certification Section seeing as new certifications come onboard.Â

Start learning today!

-Harness Product Education Engineering Team

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/harness-certified-experts-continuous-delivery-admin-certification>

Harness Certified Experts - Continuous Delivery & GitOps Admin Certification is GA

As technology paradigms and platforms continue to evolve, keeping your skills sharp can be an ongoing journey. At Harness, we strive to lower the barrier of entry for learning and validating your experience. Today, we are pleased to announce the first in our administrator level of certifications, the Continuous Delivery & GitOps Administrator Certification.Â

Building upon the certification framework we have started this year, the administrator level certification incorporates a hands-on portion to test a userâs implementation of Harness features and to help you manage your Continuous Delivery capabilities at scale. This will unlock capabilities and abilities to leverage Harness Continuous Delivery & GitOps Enterprise to attain your CD goals.Â

â

A New Hands on Experience

The Continuous Delivery & GitOps Administrator Certification will now incorporate hands-on experience. After the knowledge based exam, you will be presented with several hands on questions to solve in the Harness Continuous Delivery & GitOps Enterprise Module. There is no need to bring your own infrastructure or Harness Account as all of the infrastructure you need is available via a cloud shell in your web browser. You will be provided with an ephemeral Harness Continuous Delivery & GitOps Enterprise account.Â

â

â

There are two portions to the exam, a knowledge and hands on portion. The first step towards your administrator badge is to take a look at the learning path on Harness Developer Hub.Â

Preparing and Taking Your Administrator Exam

The learning material is hosted on Harness Developer Hub. The administrator certification builds off of the developer level certification. It is recommended that you take the developer level certification for free before attempting the administrator level certification.Â

â

Â

Registering for the exam is simple. Head to Harness Developer Hub, then click Register for Exam. You have the ability to take the exam at any time on your schedule.

Admin/Architect Level Exam Video

Once you have successfully completed the exam, a Credly Badge will be issued to you in a few days. At Harness, we are creating certifications for multiple levels.Â

Harness Certifications: Continue Learning

We continue to plan to roll out certifications at multiple levels covering every Harness Module. Even if you do not become certified, check out the learning paths that are open and available to increase your skill regardless of what level you are at.Â

-Harness Product Education Engineering Team

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer

happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/using-feature-flags-for-effective-incident-management?utm_source=pmm-ff&utm_medium=release-management-with-feature-flags

Using Feature Flags for Effective Incident Management

Whether you call it an incident, outage, surprise, or unplanned work, your application isn't working as expected, and you need to deal with the problem. That's where dependable incident management comes in. You never want to be responding to an incident without a plan or prior systems set in place. You may not always be able to stop an incident, but you can always dampen and remediate its effects with good planning.

How Feature Flags Can Help for Incident Management

Feature flags can help you improve your incident management process in a number of ways:

- **Kill switches and circuit breakers:** You can use feature flags to disable features or services quickly and easily without having to redeploy your code. This can be helpful if you need to quickly isolate the source of an incident or prevent a problem from spreading. This can prevent a feature from causing problems for your users.
- **Throttling:** You can use feature flags to throttle traffic to certain features or services. This can be helpful if you're experiencing performance issues or if you need to prevent a feature from being overloaded. This can help to improve the performance of the service for more critical or demanding users.
- **Debugging:** Feature flags can help you debug incidents by providing you with insights into which features or services were recently enabled or changed. This can help you quickly identify the root cause of an incident. If it's not obvious at first, you can also enable and disable different features and services to see how they impact performance and study the affected systems.
- **Automation:** You can use feature flags to automate tasks related to incident management, such as adjusting logging levels, rate-limiting services, or disabling features. This can help you reduce the time it takes to resolve incidents.

Tips for Effective Incident Management

Here are some tips for using feature flags for incident management:

- **Create a flag management plan.** This plan should define how you will create, manage, and use flags for incident management. It should also include a process for reviewing and updating your flagging strategy on a regular basis. We've seen this be very effective for teams using feature flags at Harness. Remember: a little extra time spent on this before can save a lot of time (and money) after.
- **Use flags to implement operational resilience practice.** Operational resilience practices are designed to help you minimize the impact of incidents on your users. For example, you can use flags to implement circuit breakers or to throttle traffic to certain features during an incident.
- **Monitor your flags.** It is important to monitor your flags to ensure that they are being used as intended. You should also monitor the impact of your flags on your application and users.
- **Testing your incident response plan.** Use feature flags to simulate different types of incidents during your incident response testing. This will help you to identify any areas where you need to improve your plan. If, however, an actual incident occurs, treat it as another (harsher) test of your incident response plan and make sure to apply all lessons learned during it. Even if you had an effective incident response, it's important to acknowledge what went right.

By following these tips, you can use feature flags to improve your incident management process and reduce the time it takes to resolve incidents.

Benefits of Using Harness for Incident Management

In addition to the general benefits listed above, Harness Feature Flags also offer a number of differentiators that can be particularly helpful for incident management. Here are some specific examples of how you can use Harness Feature Flags with OPA, Pipelines, and Git Sync for incident management:

- **Use OPA to enforce policies on feature flags:** You can use OPA to enforce policies on feature flags, such as requiring that certain approvals be obtained before a feature flag can be enabled. This can help to prevent accidental or unauthorized changes to feature flags during an incident.
- **Use Pipelines to automate the process of enabling and disabling feature flags:** You can use Harness Pipelines to automate the process of enabling and disabling feature flags. This can help to streamline your incident management process and reduce the risk of human error. For example, you could create a pipeline that enforces that a feature be tested in non-production before the feature is enabled in production.
- **Use Git Sync to keep your feature flags in sync with your Git repository:** You can use Harness Git Sync to keep your feature flags in sync with your Git repository. This can help to ensure that your feature flags are always up-to-date and that you have a complete audit trail of all changes. For example, you could use Git Sync to create a branch for each incident fix and then use that branch to track all changes to feature flags related to that incident.

Beyond incident management, here are some other key benefits of using Harness Feature Flags for your project:

- **Automated lifecycle management :** Harness Feature Flags helps you manage your flag tech-debt by detecting potentially stale or deprecated flags and automatically removing them if neededâ
- **Integrated with CI/CD:** Since Harness covers a range of software delivery products, you can easily integrate your feature flags into CI/CD as a unified pipeline that helps maintain the flags as part of the software development lifecycle.

Conclusion

By using Harness Feature Flags for incident management, you can improve the resilience of your application and reduce the impact of incidents on your users. Incident management starts before you write your first line of code and continues after your code is in production. By following the tips above, you can effectively contain the blast radius of any unexpected incidents and restore the user experience.

If you want to use Harness Feature Flags for incident management, get started for free now!

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Harness CI and Harness CD - Your First True CI/CD Pipeline

Harness now allows you to have both best of breed Continuous Integration and Continuous Delivery capabilities. Let's take a look at what leveraging a Harness CI / Drone Pipeline to build then leveraging Harness CD to deploy to a Kubernetes cluster.

The goal of this example will be to have Harness CI / Drone build and push a GoLang Docker Image to DockerHub then have Harness CD pickup on the new artifact and deploy to an awaiting Kubernetes cluster. This potentially can take you from idea to production easier than ever.

The Moving Pieces

There are not many moving pieces to see an end-to-end example. You need a Harness CI / Drone Instance, a Docker Hub Account, a Harness Account, and of course, a Kubernetes cluster which could even be Minikube.

Like always you can follow along with the blog and/or watch the video.

If leveraging the example Harness CI / Drone GitHub project, you will have to edit the information to be your own Docker Registry for where the Docker Push goes. I will be referencing mine below.

The Harness CD Piece

If this is your first time leveraging the Harness Platform, the quickest way to get started is to leverage a Kubernetes Delegate.

The first step of getting a Harness Delegate installed in your Kubernetes Cluster is pretty straight forward.

Setup -> Harness Delegates -> Download Delegates -> Kubernetes YAML

Give the Delegate a name.

Hit submit to download. Expand the tar.gz which is downloaded.

Inside the delegate folder, run `kubectl apply -f harness-delegate.yaml` to install the delegate.

In a few moments, your Harness Delegate will be available.

Next, add the Kubernetes cluster for Harness to deploy to by adding the Kubernetes cluster as a Cloud Provider.

Setup -> Cloud Providers + Add Cloud Provider

Add a Kubernetes Cluster

Give the Kubernetes cluster a name then for Cluster Details, Inherit from selected Delegate [deployed Delegate]. Hit Test then Submit and you are all wired up.

Next, creating a Harness Application is a simple process.

Setup -> + Add Application.

The next step inside the Application is to create a Harness Service. You can create a Service by going to

Setup -> Captain Canary -> Services + Add Service. The Deployment Type will be Kubernetes.

Inside the Amazing App Service, + Add Artifact Source from a Docker Registry

Can wire to your Docker Image of choice.

Once you hit submit, the scaffolding will be there for the deployment and no need for additional configuration.

With the Service out of the way, we can wire a Harness Environment to deploy to.

Setup -> Captain Canary -> Environments + Add Environment

Once you hit Submit, next can wire an Infrastructure Definition.

Next click + Add Infrastructure and add the Kubernetes cluster. The Cloud Provider Type and Deployment Type will be Kubernetes.

Once you hit Submit, can wire together a Harness Workflow to define the steps to deploy.

Setup -> Captain Canary -> Workflows + Add Workflow. The Workflow Type will be Rolling Deployment. Select the Environment, Service, and Infrastructure Definition that was created in the previous steps.

Next you can create a Harness Trigger on the presence of a new artifact in DockerHub. This will trigger the Workflow once the Docker Push is complete in the Harness CI / Drone Pipeline.

Setup -> Captain Canary -> Triggers + Add Trigger

Hit Next after giving a Name. DockerHub can send out a webhook or we can use a polling interval from Harness to look out for a new artifact. Without the need to manage webhooks, we can just simply configure On New Artifact. Select your artifact source and can filter on specific tags. For the example we just want to deploy what gets picked up and it's fine as a filter.

Hit Next and define the Actions. Execution Type will be Workflow and will execute the Workflow that was created in the previous steps. Can leverage the Last Collected Artifact. If leveraging the example will pick up on the newest explicit tag with the same filter it's fine as a filter.

Click Next and review the Trigger then hit Submit.

Once you hit Submit, all you have to do is kick off a Harness CI / Drone build and push.

Harness CI / Drone Steps

If you do not have a running Harness CI / Drone instance don't worry, not difficult to accomplish. We have a detailed blog and video to get you started. For the example to work will need to be building and pushing a Docker Artifact. The example project has a simple GoLang application. The Harness CI / Drone introduction blog and video go through wiring Harness CI / Drone to a GitHub repository.

To kick off a Harness CI / Drone Build and Push, simply modify the configuration / Drone.yaml which Harness CI / Drone is monitoring for. To explicitly build to a specific version, can add tags to Drone.yaml which works well with the example.

In the example, pushing build version 1.0.2. Make sure to update your repository information also.

Can modify/increment then commit.

Once you hit Commit, Harness CI / Drone will take over.

Once the publish step is finished, a new image will be in DockerHub.

Watch the Magic

Within the polling interval, Harness will pick up on the newest build/tag and start deploying on your behalf.

With the Deployment kicked off, you have an end-to-end CI/CD pipeline. From code to production!

Partner with Harness in your CI/CD Journey

No matter where you are in your CI/CD journey, Harness can simplify and scale your capabilities. With the lightweight and convention-based Harness CI / Drone and the power of the Harness Platform, achieving CI/CD nirvana has never been easier. Get started with Harness CI / Drone and sign up for a Harness Account today!

Cheers,

-Ravi

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/kill-switch-code?utm_source=pmm-ff&utm_medium=release-management-with-feature-flags

6 Use Cases for Kill Switches in Production Using Feature Flags

If you've only come to feature flags recently, you may not know that feature flags as a concept has existed for quite a long time, and can easily be seen as an evolution of app config systems. But first things first - if you need a refresher on feature flags, head on over to our "What Are Feature Flags?" article now.

One of the least used - but most powerful - use cases of kill switches using feature flags can be for Ops and SRE teams. Plenty of folks never think about this potential, but it can be a way in which feature flags can most transform your engineering organization. They're a great way to increase velocity and reduce risk in software development.

Kill switches create tremendous potential both to control granular capabilities or features within an application, and also as long-lived operational switches that affect the entire application. And while feature flags are intuitively thought of as affecting frontend or client-facing applications, they can also be used just as well on the backend. Let's explore how you can leverage kill switches to change the nature of how your team can deliver software.

Use Case 1: Control in Production With Feature Toggles

Consider this the base case of implementing kill switches using feature flags. The concept is simple enough: you can rig any and all features and changes pushed to production (or any environment) with a toggle, or a kill switch. Feature flags, by nature, allow teams to isolate features within their applications so that they can be handled individually instead of as a deployment bundle. Kill switches themselves are the way in which teams can create a control to disable code in production at any time.

This basic kill switch functionality gives teams the ability to turn off a feature when it's not ready. "Not ready" can mean two things in prod:

Unfinished Features

The first condition for not being ready is what slows down many teams during the deployment process. In particular, as many teams work on different features across different applications, integration and deployment take a long time. Without having everything rolled in, release teams and the software development teams that built them both find themselves slowed down.

Using feature flags as kill switches in this context allows teams to push new features to production even as incomplete features, which can result in faster development cycles. It simply needs to be pushed live, but with the kill switch activated (feature turned off) so that it's not actually impacting anything.

Broken Features

Here's a developer favorite: it worked in all the pre-production environments, and then a bug was discovered that's now impacting production. Without the use of a feature flag, that broken feature impacting prod would now necessitate a complete rollback of the deployment - that's right, all of the good features have to be pulled off the shelves too and production is switched back to the last working version.

On the other hand, rigging each new feature or change with a feature flag enables teams to hit the kill switch as soon as an issue is found. Now, all of the features that work get to stay in prod, while the broken features are isolated out and handled as required.

Use Case 2: Incident Management

What happens when a severe issue occurs in production because of feature releases that are broken? Or when some infrastructure failure cascades across all of production? As it turns out, kill switches can be used to mitigate these incidents and improve the MTTR (mean time to resolution) for the teams responsible. Talk about stress management.

Scenario 1

In scenario 1, a broken new feature is causing an incident in production. The good news is, if that new feature was rigged up to a feature flag, the kill switch could be triggered and that feature could be turned off in production right away, resulting in minimal impact. Not only is the feature no longer affecting production, but there is also suddenly no massive response team that needs to be assembled to handle the issue, which would typically mean the whole deployment had to be pulled. Instead, the feature is turned off, the team responsible for the feature is on the hook to fix it, and the one feature fix can be rolled forward when it's ready.

Scenario 2

In scenario 2, an infrastructure failure is one example of what could happen in production. However, production outages can happen due to a variety of issues. Oftentimes, Ops teams will have fallback options or adjust turn it all off runbooks in place, and kill switches can be used here as well. If it's a new infrastructure change entirely, that whole change can be wrapped in a feature flag and a kill switch can be triggered that will failover to the old setup when an issue occurs. Some teams also create long-lived operational flags that can, say, put a whole site or application in maintenance mode, effectively triggering a kill switch for the whole application. This can be useful in P0 scenarios where things are completely broken and the preference is to not allow any access.

Use Case 3: Application Load Management

Let's say your business is an e-commerce company. During the holidays, you experience high load relative to the rest of the year. Now more than ever, the business doesn't want to have downtime or have customers experience issues! Turns out feature flagging is an incredibly low-cost way to solve this problem.

One approach to solve this problem could be to simply turn off some features that will reduce the load, and this is a great place to use a kill switch, but it might not be the most effective. Another solution might be to kill certain load balancers or servers that are useful in non-peak times, and cut over to infrastructure that's designed specifically for high load.Â

Real World Example

At a previous job, we had log data that would need to go into storage from the short term cache, so we had a system to collect and store them. During peak loads or other periods of instability (particularly upstream with AWS), we would often see this storage become either very expensive or very unreliable. So, we would turn this service off during these periods, but this was manual and involved a series of AWS commands and database changes to both disable and re-enable later on. By wiring this up to a feature flag, we were able turn this feature off and on instantly in a much more lightweight, visible, and easy to govern way.

Use Case 4: Testing in Production

One of the most common use cases for teams doing feature flagging is testing in production. In short, this is where a specific feature (or set of features) needs to be tested against real production data. After all, pre-prod environments can only test so many things! There are two use cases that we see here:

The use of feature flags as kill switches here is a pretty simple one: have it live until the test has been run, and then turn it off. It's also possible that a feature will fail during testing and need to be turned off to minimize impact. You can dig deeper into testing in production in this dedicated blog.

Use Case 5: Progressive Delivery

Here, let's assume a basic knowledge of progressive delivery. It's becoming an increasingly popular feature release methodology, very similar to the methodology used for canary releases in Continuous Delivery.

With progressive delivery, the need for a kill switch changes slightly. Yes, the main use case is still turning things off when they break or aren't needed, but here, we have to layer in automation. By its nature, progressive delivery is something that teams run in an automated fashion. To do that, they need to pre-define in which scenarios flags are turned on, for whom, and under what conditions more users get access. Conversely, they must define failure criteria in which the kill switch is triggered and new features are turned off, or the user base is shrunk.

In Harness Feature Flags, we automate progressive delivery via the Pipeline. Teams can schedule releases, mandate approvals, integrate with plugins, create trigger events, and template rollouts - and then automate it.

Use Case 6: Personalization

Personalization is a hot topic. Kill switches using feature flags are of huge value in being able to do this well at scale. We can look at three distinct scenarios in which you can use kill switches to improve your ability to personalize experiences for customers.

Compliance & Regulation

When it comes to regulation, it's a non-negotiable item to be able to personalize something like a mobile app release to comply with the laws in various countries, or to meet the needs of clients in specific verticals (e.g. government, finance). The most common of these is data privacy - specifically, complying with GDPR.

Considering this, it can be a mess showing a button in one place but not another. Wiring key features up as permanent feature flags

allows you to easily turn something on for all your North American users, but not your European users where GDPR compliance is mandatory. It's not just creating a kill switch that given context will turn off data collection in Europe, it's also being able to kill data collection for any user that decides to opt out. It's much easier than having a configuration file that tries to solve for all of the various scenarios.

User-Defined Personalization & Admin Settings

Think of this as giving control to the user on what features they want to kill. This isn't dissimilar to giving them access to admin settings on the app so they can choose to have dark mode or choose what notifications they get.

While on the frontend, users will define what they want and don't for their personal experience on the app, on the backend, it's just exposing parts of the feature flag schema to the end user and letting them make the decision on what to keep and what to kill.

Another consideration here is the use of features like screen time or parental controls, where a kill switch can be triggered to limit the access of users to specific applications altogether based on settings elsewhere on their devices. Of course, putting the onus on the user to define what they will do with such power is a whole other problem to solve.

Using Harness Feature Flags to Implement Kill Switches

Here's the plug: using Harness Feature Flags gives you the ability to implement all of these use cases right out of the box. When you set up feature flags in Harness, you're able to work either in a visual UI or entirely in code to manage your kill switches. You also give yourself peace of mind when you eventually want to scale your usage of kill switches or other feature flags use cases with built-in governance, compliance, and security considerations, as well as the critical integration into CI/CD (Continuous Integration and Continuous Delivery).Â

After all, you don't want to set up a great feature flagging system that is entirely disconnected from the rest of your software delivery process. It doubles your work on access control, security, governance, and integration maintenance. You end up getting diminishing returns on your feature flagging platform, which is supposed to *accelerate* software delivery.

If you want to see how Harness can fit into your organization, you can sign up for free forever, or contact us for a personalized product demo.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

[Documentation](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[Case studies](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[We want to hear from you](#)

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

[Sign up for our monthly newsletter](#)

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Introducing the Harness Developer Hub - Beta Release

Change is the only constant in technology. Between balancing domain knowledge, gaining technical stack expertise, and understanding all of the semantics and terminology, we need information ready at our fingertips so we can deploy safely and on time. With the goal of furthering and simplifying developer and engineering education, we are pleased to announce the launch of the beta release of Harness Developer Hub, or HDH located at <https://developer.harness.io>.

The Harness Developer Hub (HDH) brings all technical material under one easy-to-access and consistent umbrella. The HDH has a variety of tutorials, videos, and reference documentation to help you along your software delivery journey. Users of the HDH can pick common software delivery tasks that are verb-based such as âdeploying an applicationâ or âbuilding codeâ to get started.

Touring the Harness Developer Hub

The HDH is task- and verb-based. The HDH has everything from teaching you how to build and package code to learning common approaches.

If, for example, you wantÂ to learn more about reliability, such as managing service level objectives, those types of tutorials can also be found on the HDH.

Module documentation is also starting to be housed and migrated in the HDH. Reference material for Harness Modules will be accessible in the Documentation section.

These are just a few examples of what you can find on the HDH, and thereâs more to come. In your eyes, if we are missing a tutorial or a piece of material that would help in your journey, you have an opportunity to contribute to the HDH.

How to Contribute to the Harness Developer Hub

The power of many is greater than the power of one. The HDH is open source first, taking in contributions big and small. We are also always open to feedback as this is an evolving beta release. Feel free to submit in the on-site feedback form or leverage GitHub for raising an Issue or a PR of your own.

â

â

Leverage our Contributors Guide if you would like to help put your stamp on the HDH. The HDH will be evolving over the next months as more material and capabilities are built out.

Whatâs Next For The Harness Developer Hub

As we work towards GA on the HDH, we will be onboarding more tutorials and migrating documentation. We are also looking at ways to more closely align what is presented to you with what is in your Harness Account. Head over to the Harness Developer Hub today and let us know your feedback!

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/introducing-harness-feature-flags?utm_source=pmm-ff&utm_medium=release-management-with-feature-flags

Introducing Harness Feature Flags

Harness Feature Flags is the newest addition to the Harness Software Delivery Platform. This module allows organizations to deliver features faster, with less risk.

Software Delivery Always Gets Faster

Companies constantly develop new software features for their customers. Traditionally, these features are made available via a software deployment and become visible to all users at the same time.Â

As a result, every time a new feature is deployed, there is a risk that customers will have a less than stellar experience due to a poorly-implemented feature. The only way to fix a poorly-implemented feature is to immediately roll back to the prior version, or to create a fix as quickly as possible and roll forward by deploying a new version.

Within this traditional release process, many problems become apparent:

- The risk associated with new feature releases ends up slowing down developers since they have to thoroughly validate design and implementation before releasing.
- Often, there are multiple possible versions of how a new feature will look and function, and itâs up to the developer to choose one (not always the right one) and implement it.
- The decision of when to release new software features is controlled by developers instead of the business.
- Sunsetting old features introduces risk to newer features that might have dependencies on the old code.
- Itâs all or nothing - a feature is rolled out to everyone at the same time instead of incrementally rolled out for verification purposes.
- When teams are beholden to engineering release cycles (e.g. once a month), multiple features are released simultaneously, introducing complexity and risk that can result in deployment war rooms, large rollbacks, and dissatisfied customers.

Said simply, it comes down to two things:

So how does an organization achieve higher engineering velocity without breaking existing processes, and indeed, reducing deployment risk even further?

Old Tools Can't Solve New Problems

Companies today primarily try to solve these problems by repurposing their existing release processes. This means that they're using mature and oftentimes complex methodologies that require a complicated git branching strategy to release features piecemeal or faster. Think of trying to use a cannon to kill a fly - it can work, but itâs probably overkill. While repurposing existing tooling can compensate in some ways, it ends up being very expensive and may miss on the specific requirements for an individual feature release.

Most companies are still not using feature flags. It is still not a universally-adopted practice, although itâs on its way to being so. Clearly, repurposing existing tools is not cutting it. And so teams need to find new ways to solve the problem. Quickly.

Issues on the Feature Flag Maturity Journey

The primary problem teams have is simply getting started fast and realizing value. Teams can get bogged down in figuring out how to start, they can design flags in counterproductive ways, and they can create future problems due to this immaturity in the beginning.

For teams who are more mature with feature flagging, there is a âbest practicesâ and management problem. Once a team has dozens or hundreds of flags, that means there is a lot of stale code, not a lot of process for managing it, and teams often struggle operationally to deal with this as part of a healthy workflow.

For some of the most mature teams, feature flags are often closely tied to their software delivery process, but fundamentally disconnected from it due to the existing vendors all being stand-alone tools. This means teams have to invest a lot of time in wiring up feature flags

with metrics systems, build systems, etc. because the tools are not integrated natively. It also introduces significant maintenance overhead to ensure all these systems continue to work together over time.

So clearly, there are working solutions out there, but it seems they all have their own problemsâ|

â

Harness Feature Flags = More Speed, Less Risk

Harness Feature Flags started as an internal project to use feature flags with our own software. The lessons learned from that project were applied to building the customer-facing Harness Feature Flags - everyone gets to benefit from our successes and failures.Â

â

In particular, Harness Feature Flags focuses on the following key concepts:

- **Developer Experience** - Putting heavy emphasis on the developer experience to ensure the setup process is fast and intuitive, and that teams are able to get started and see value from feature flags right away.
- **Management & Governance** - Letting teams build rules and processes and automating cleanup and flag lifecycle management. Benefits of Harness Feature Flags: Developer Experience, Management & Governance, and $2+2=5$. Ensuring teams can keep their systems secure, compliant, and standardized wherever possible is critical to their goals.
- **2+2=5** - Leveraging the Harness platform, making it easy to build workflows that are otherwise very difficult and manual to build. While users donât need to use other Harness products, they create a whole greater than the sum of the parts.

What Makes Harness Feature Flags Unique?

While there are a variety of strong feature flag solutions in the market, including in-house builds, they tend to miss the mark on some key requirements for customers. At its core, feature flags are used as software switches to hide or activate features, but Harness Feature Flags goes beyond just that basic use case:

- **Simple UI-Based Feature Release Workflows** - Customers have the ability to create templates and processes that they can standardize across feature flags that have the same operational needs. Feature flags are now included in a visual pipeline that includes steps related to governance and verification.Â
- **Governance & Verification** - Customers can ensure production pushes always meet defined organizational standards, and that they can minimize the negative impact of any issues in prod. Harness gives customers the ability to create controls in their feature release process, including mandating approvals and creating audit trails. In addition, customers can automate service verification once a feature is live, ensuring that if an issue occurs, the feature is turned off to minimize impact.
- **Integration Into CI/CD** - Many feature flag tools separate feature flags from the rest of the software development lifecycle even though they go hand in hand with continuous delivery. This creates a problem for customers, who inherently view feature flags as just another stage of that lifecycle (many companies even repurpose their existing software release processes to mimic the functionality of feature flags). By representing feature flags as a pipeline, feature flag management becomes a natural step in the everyday workflow of development teams and is integrated into CI/CD as a unified pipeline.

How To Get Started

To get started with Harness Feature Flags, just sign up for a demo and a Harness specialist will get you going.

Better yet, youâll soon be able to get started with Harness Feature Flags in an entirely self-service manner. Check back in a couple of weeks for even more developer goodness!

You can also check out this video that walks through how Harness Feature Flags is designed specially for developers.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/harness-ci-and-harness-cd?utm_source=pmm-ff&utm_medium=release-management-with-feature-flags

Harness CI and Harness CD - Your First True CI/CD Pipeline

Harness now allows you to have both best of breed Continuous Integration and Continuous Delivery capabilities. Let's take a look at what leveraging a Harness CI / Drone Pipeline to build then leveraging Harness CD to deploy to a Kubernetes cluster.Â

The goal of this example will be to have Harness CI / Drone build and push a GoLang Docker Image to DockerHub then have Harness CD pickup on the new artifact and deploy to an awaiting Kubernetes cluster. This potentially can take you from idea to production easier than ever.Â

The Moving Pieces

There are not many moving pieces to see an end-to-end example. You need a Harness CI / Drone Instance, a Docker Hub Account, a Harness Account, and of course, a Kubernetes cluster which could even be Minikube.

Like always you can follow along with the blog and/or watch the video.

If leveraging the example Harness CI / Drone GitHub project, you will have to edit the information to be your own Docker Registry for where the Docker Push goes. I will be referencing mine below.Â

The Harness CD Piece

If this is your first time leveraging the Harness Platform, the quickest way to get started is to leverage a Kubernetes Delegate.Â

The first step of getting a Harness Delegate installed in your Kubernetes Cluster is pretty straight forward.Â

Setup -> Harness Delegates -> Download Delegates -> Kubernetes YAML

Give the Delegate a name.

Hit submit to download. Expand the tar.gz which is downloaded.

Inside the delegate folder, run `kubectl apply -f harness-delegate.yaml` to install the delegate.

In a few moments, your Harness Delegate will be available.

Next, add the Kubernetes cluster for Harness to deploy to by adding the Kubernetes cluster as a Cloud Provider.Â Â

Setup -> Cloud Providers + Add Cloud Provider

Add a Kubernetes Cluster

Give the Kubernetes cluster a name then for Cluster Details, Inherit from selected Delegate [deployed Delegate]. Hit Test then Submit and you are all wired up.

Next, creating a Harness Application is a simple process.Â

Setup -> + Add Application.

The next step inside the Application is to create a Harness Service. You can create a Service by going toÂ

Setup -> Captain Canary -> Services + Add Service. The Deployment Type will be Kubernetes.

Inside the Amazing App Service, + Add Artifact Source from a Docker Registry

Can wire to your Docker Image of choice.

Once you hit submit, the scaffolding will be there for the deployment and no need for additional configuration.

With the Service out of the way, we can wire a Harness Environment to deploy to.Â

Setup -> Captain Canary -> Environments + Add Environment

Once you hit Submit, next can wire an Infrastructure Definition.

Next click + Add Infrastructure and add the Kubernetes cluster.Â The Cloud Provider Type and Deployment Type will be Kubernetes.

Once you hit Submit, can wire together a Harness Workflow to define the steps to deploy.

Setup -> Captain Canary -> Workflows + Add Workflow. The Workflow Type will be Rolling Deployment. Select the Environment, Service, and Infrastructure Definition that was created in the previous steps.

Next you can create a Harness Trigger on the presence of a new artifact in DockerHub. This will trigger the Workflow once the Docker Push is complete in the Harness CI / Drone Pipeline.Â

Setup -> Captain Canary -> Triggers + Add Trigger

Hit Next after giving a Name. DockerHub can send out a webhook or we can use a polling interval from Harness to look out for a new artifact. Without the need to manage webhooks, we can just simply configure On New Artifact. Select your artifact source and can filter on specific tags. For the example we just want to deploy what gets picked up and â.*â is fine as a filter.

Hit Next and define the Actions. Execution Type will be Workflow and will execute the Workflow that was created in the previous steps. Can leverage the Last Collected Artifact. If leveraging the example will pick up on the newest explicit tag with the same filter â.*â.

Click Next and review the Trigger then hit Submit.

Once you hit Submit, all you have to do is kick off a Harness CI / Drone build and push.Â

Harness CI / Drone Steps

If you do not have a donât have a running Harness CI / Drone instance donât worry, not difficult to accomplish. We have a detailed blog and video to get you started. For the example to work will need to be building and pushing a Docker Artifact. The example project has a simple GoLang application. The Harness CI / Drone introduction blog and video go through wiring Harness CI / Drone to a GitHub repository.Â

To kick off a Harness CI / Drone Build and Push, simply modify the configuration / Drone.yaml which Harness CI / Drone is monitoring for. To explicitly build to a specific version, can add tags to Drone.yaml which works well with the example.Â

In the example, pushing build version 1.0.2. Make sure to update your repository information also.

Can modify/increment then commit.

Once you hit Commit, Harness CI / Drone will take over.

Once the publish step is finished, a new image will be in DockerHub.

Watch the Magic

Within the polling interval, Harness will pick up on the newest build/tag and start deploying on your behalf.

With the Deployment kicked off, you have an end-to-end CI/CD pipeline. From code to production!

Partner with Harness in your CI/CD Journey

No matter where you are in your CI/CD journey, Harness can simplify and scale your capabilities. With the lightweight and convention-based Harness CI / Drone and the power of the Harness Platform, achieving CI/CD nirvana has never been easier.Â Get started with Harness CI / Drone and sign up for a Harness Account today!

Cheers,

-RaviÂ

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/what-are-feature-flags?utm_source=pmm-ff&utm_medium=using-feature-flags-for-effective-incident-management

What are Feature Flags and How Do We Use Them?

Building software is a great experiment. Many times we are required to do what has not been done before; this is core to innovation work. However, a lot of the iteration and experimenting tends to happen before software is shipped. That's where the concept of test in production comes into play. Building something from scratch or even jumping into an older project mid-stream requires lots of iteration.Â A

Once we deploy, for many software engineers, the experimentation on the existing tends to drop off and onwards to the next set of features in the backlog. Product owners/managers typically collect feedback to help prioritize the next set of features. From a software engineering perspective this can feel a little âchicken or egg-ishâ - without a feature actually deployed in a production environment to get feedback, how do we if know we are on the mark?

Letâs illustrate this with an example. Customers are asking for an updated UI view for their dashboards; they want something that surfaces the most relevant information faster. Now that the product team has collected those requirements, itâs up to engineering to figure out how to best build it. There might be a few potential solutions:

With some potential solutions decided, now itâs up to the engineering team to decide which of the three to implement. And before a customer ever sees it, they have to choose 1 of 3, build out the MVP over the next few sprints, and then ship it. This is the first time theyâll get any feedback, and if itâs not the right solution then it has to be built again!

Now, imagine if this new feature cycle could be improved for product development teams. If youâve been paying attention to developer tools over the last few years, you might have seen a new category popping up: feature flags. Feature flags as a concept were born to increase development velocity and to solve this very problem through practices like progressive delivery.

While feature flags as a mature category of hosted tooling is new, feature flags - and what they solve - have been around a little longer. Letâs take a look at what feature flags have grown into, and how to use them.

The Story of Feature Flags

Feature flags, at their core, are the ability to wrap different versions of your code in conditional statements that you can turn on and off at will. You may have previously thought of them as app configs, or even part of your company's rules engine.

With Harness Feature Flags, we've taken these concepts you may be familiar with and taken them even further - adding more sophisticated rules, governance, user management, an intuitive UI, and more.

Feature Flags lets you separate feature release from code deployment, build more effective strategies for rolling things out and testing with your users, and have more insight than ever into how changes impact your application performance.

Ultimately, feature flags give engineering teams the ability to deliver more features, with less risk. For leaders, it's a great way to improve engineering velocity and developer experience, and to reduce the risk associated with feature development. And for developers, it's a great way to work more efficiently, with less stress - think fewer late nights before a deployment, and fewer firefights post-deploy.

What Are Feature Flags (or Feature Toggles)?

Feature flags are a way that developers can conditionally turn certain sections of their code on or off. You can think of feature flags as extending Continuous Delivery into Continuous Deployment - a way to put changes into production behind a flag and turn them on in a controlled way later (or hide and remove them in the same way).

But you can also think of feature flags as a new way to think about building and releasing applications - by clearly defining features and components that can be changed and toggled on and off individually, you allow for experimentation, controlled rollouts to different users, as well letting more people (like PMs, sales, and support) turn things on and off for customers.

Together, this results in giving more power and control to developers to create more and better features faster, while at the same time ensuring that engineering is not the bottleneck for the business in having to make changes for individual customers or ensuring a perfect production deployment for every new feature or feature update.

If we want to simplify even further, feature flags essentially create private swim lanes for developers where they can ship a feature directly to customers and then have control over who sees it, get feedback, and turn it on and off as needed. It's like a light switch (a pretty complex one)!

Who Uses Feature Flags?

People oftentimes consider feature flags either an engineering tool, or a tool for product managers. The reality is, it's both. Flags can help software development and DevOps teams lower their overhead and increase their velocity, and they can help product managers better control releases, coordinating launch timings and creating a feedback loop more effectively.

Research indicates that 95% of engineering leaders want to implement a feature flag solution for themselves. This comes on the back of 97% of leaders saying they're under pressure to deliver faster, and 65% of whom say they find it difficult to achieve that velocity increase in a safe manner. It's abundantly clear that engineering teams are likely the biggest beneficiaries of implementing and using a feature flag solution, and indeed, they tend to be the primary users. And when they do implement a feature flag solution, they end up delivering 66% more features per application per year!

Feature flags are great for users across the organization beyond product and engineering. For sales and support teams, feature flags provide a way for them to directly manage betas and new features for their customers, and to track down who's using what in case of any issues. And for management, flags can provide new forms of visibility into what's happening in development and how new features are being tested with users.

On a fun fact kind of note and since we're in the "who uses feature flags" section, it's cool to note that big companies like Netflix, Flickr, Google, Reddit, and more use feature flags! They're definitely worth looking into and absolutely are an important part of the software delivery lifecycle.

Why You Should Consider Feature Flags

Whether or not the use of feature flags is important to your software delivery lifecycle comes down to what it is that you need to accomplish. While feature flags are a great tool to have for any team, they are particularly useful if you have these problems:

Problems in the Business

- The risk associated with new feature releases ends up slowing down developers since they have to thoroughly validate design and implementation before releasing.
- There are often multiple possible versions of how a new feature will look and function, and it's up to the developer to choose one (not always the right one) and implement it.
- The decision of when to release new software features is controlled by developers instead of the business.
- Sunsetting old features introduces risk to newer features that might have dependencies on the old code.
- It's all or nothing - a feature is rolled out to everyone at the same time instead of doing a feature rollout incrementally for verification purposes.
- When teams are beholden to engineering release cycles (e.g. once a month), multiple features are released simultaneously, introducing complexity and risk that can result in deployment war rooms, large rollbacks, and dissatisfied customers.

Problems in Engineering

- Releasing a feature takes too much coordination and time across stakeholders.
- Merging new features, or managing separate feature branches for changes, is cumbersome and complex.
- You want to test changes to see how they impact your application and end-users before rolling them out more widely.
- You want to provide access to new features, or to test changes with certain portions of the user base sooner - by location, by customer plans, by customers opting into beta testing, or by any other criteria or logic you can think up.
- You want to reduce the risk of rolling out new code by having instant kill switches that don't require rollbacks or new deployments to fix a disrupted environment.

To simplify, we can consider that feature flags are a great solution if you're trying to do the following things:

- Increase engineering velocity and ship more features.
- Decrease feature development risk by A/B testing multiple solutions with customers.
- Reduce the risk and toil associated with production failures and rollbacks.
- Decouple engineering deployments from feature unveilings to customers.

Different Types of Feature Flags/Toggles

It's natural to want to put all feature flag use cases in a single bucket, but it's more helpful to instead view flags themselves as a means to an end and as having categories of use cases.

Here are some examples of the different kinds of flags that could be used in a system:

- **Release features** - Using flags to let the right people decide who to turn a feature on for, and when, without complex engineering deployment coordination required.
- **Experiment** - Using flags to learn something about how a change impacts things. This can be a technical experiment, such as how a new UI impacts server load, or it can be a user experiment such as how a new button impacts conversion.
- **Ops flags** - Letting you have permanent control around key parts of your app so you can turn them on and off as necessary without a full new deployment cycle. Add in RBAC, and this is a robust way to control who has access to modify your production application.
- **Control access** - Using flags as a way to manage betas, early adopter list, trial access, and more.

Implementing Feature Flags

Implementing feature flags can seem really simple at first, but it turns out that like with any system, the devil is in the details. Building an awesome feature flag solution in-house certainly has its perks, but it can be deceptively difficult. In particular, we've seen three distinct sets of issues arise with feature flag solutions as an organization's implementation matures.

When a feature flag solution is **nonexistent or immature** and an organization is standing it up for the first time, they tend to run into issues with finding value in the tool quickly. Especially with the need for such a solution and the pressure to make it happen quickly, it's critical that value is demonstrated quickly, otherwise it may end up leaving a sour taste. Within that, there's a dichotomy of architecting for eventual scale versus getting an MVP out the door to prove value. This often results in complex design or architecture that isn't scalable and only works for small, targeted use cases.

As feature flag solutions enter a **middle stage of maturity**, or as they **start to scale**, the problem is quite obvious: scaling is hard. Especially if built in-house, ensuring that the tool can be used by multiple users and multiple teams for a variety of use cases, all while maintaining a common set of best practices and building on a single infrastructure can be incredibly daunting. And as this is being figured out and teams have code in production, it can quickly result in stale code or issues keeping production tidy when code changes occur. Add onto that the need to integrate neatly into everyone's workflow and you've got a behemoth of a problem to solve.

Building on that is the scenario we often see with **highly mature** feature flag implementations, where they primarily end up as separate systems from existing software delivery or CI/CD pipelines that teams now have to learn how to use, and to figure out how to port their existing processes or pipelines into the feature flag solution. At this stage, there's the obvious cost of maintenance and future development to keep up with new changes or requirements. Technical debt is bound to build up by this point, and it can become unmanageable without a dedicated team - and perhaps even its own codebase.

Sure, 95% of engineering leaders may want a feature flag implementation, but as it turns out, it's not so easy to build something that works well over time.

Build or Buy: Feature Flag Management

You may be thinking, "Feature flags seem pretty simple, can't we just build this ourselves?"

The real answer? It's complicated.

We've seen in most cases that an internally-built tool will end up causing a lot of overhead and putting a very low ceiling on the benefits you can receive from feature flags. It may be good for your first couple of flags, but it can be hard to make an internal system scalable, performant, and robust for all the things flags can do as your organization gets more and more comfortable using them - and more reliant on them.

For a deeper look at the build vs buy conversation, you can check out an article we wrote earlier this month, Feature Flags: Should I Build or Buy?

Why Use Harness for Feature Flag Management

Harness provides a feature flag platform made for getting you up and running quickly while being able to scale to any level. We make it easy to get started and quickly see the benefits of feature flags. With Harness Feature Flags, that means you can get your first flag out the door in minutes by following three steps:

- Create a free trial account.
- Create a new flag in just 3 clicks.
- Add one of our SDKs to your code in a few minutes.

And, you've got a flag working!

If that's not quite clear, then it's worth knowing that the fundamental way flags work in an environment like Harness is that SDKs, added to your application as a library, will receive updates on flag statuses to evaluate with your users (you should choose between real-time streaming or pulling updates on intervals). These SDKs are secure, collecting no data you don't want to share, and they are designed for performance and stability.Â

Harness Feature Flags is fast, secure, and focused on helping teams get the most out of feature flags - from your first initial use all the way to global governance, sophisticated reusable templates, and developer-first experiences.

And if you're wondering how we stack up against LaunchDarkly, Optimizely, Cloudbees Feature Management, Split, and other FF tools - those pages are being worked on right now and we'll update this post with the links when they're live!Â

If you'd like a little more information on our architecture and on how to get started, you can check out the feature flag documentation or request a demo.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/faster-incident-resolution?utm_source=pmm-ff&utm_medium=using-feature-flags-for-effective-incident-management

Faster Incident Resolution

Continuing on our theme of things people don't often think about when using feature flags, we want to take a look at another place where feature flags will make a huge impact for your organization, but which you may not have considered yet: incident response. It's true - feature flags can help you gain faster incident resolution. Let's dig into how.Â

Scenario

Let's take a look at how feature flags can help incident resolution. We'll use a hypothetical scenario that maybe isn't so hypothetical.

It's 3pm in California and suddenly, tickets start rolling in that users can't access a key section of the application. It appears to be a frontend bug blocking the clickable area.Â Â

The engineering team on the west coast begins to investigate, but it's not in a service most of them work on. Most of them are backend anyway. For this service, the frontend team is largely based in Madrid where it's midnight, or London where it's 11pm. They're not reachable and while they can be paged, no one likes doing that.

After some digging, it appears that a PR merged by the Europe team near the end of their day is likely a root cause. Rolling back seems like the best way to fix it. However, the ops team is based in India, where it's currently 4:30am, so they won't be online for a few more hours.

Here's where we enter into two different worlds.

Without feature flags, you have three options:

- You can page your killswitch engineers in India to do a rollback, which will also revert good changes in the release (and make the corresponding docs updates incorrect for a few hours).
- You can page the team in Europe to try to push a fix out. But unlike the team in India, they're not close to the start of their day. As such, you may have some local laws that impact working hours once you do this.
- You can simply live with the customer disruption for a few hours until India wakes up to do the revert, and then later, Europe can apply the fix. This means the customer will see an issue for a few hours, and your docs and communications around the release will take about a full day to get back in alignment. That's if the fix is a quick one.

With feature flags, the team in California can disable the flag for that specific UI update, the rest of the release without issues remains live, and the incident is over from the customer's perspective. From reporting to conclusion takes maybe ten minutes, no one is paged, no rollback is needed, and all the good parts of your release stay active.

Expanding a Bit More...

What we see in this scenario is that an incident that would otherwise cause disruption for multiple teams, require pages, rollbacks, or both - and require customer disruption for anywhere between a few hours and a full day - is over in ten minutes.Â One could say that the ability to fail safely is a critical advantage.

This is not limited to front end changes. As we explored previously, feature flags can and should be utilized for backend changes, API changes, and more. We encourage teams to think beyond "feature flags are used to release new features" and instead think about feature flags as a critical part of change management.Â

All changes to your application should potentially be released behind a feature flag. The flag, in this case, plays the role not just of a tool for more effective testing and releasing, but also as a way to make sure that no change to your application ever takes more than a single toggle to disable in prod.

ConclusionÂ

The more you dig into feature flags, the more impact you will find - and not just with faster incident resolution.

Try to think about feature flags not only as part of your release strategy. Instead, see it as part of how you design your apps to be resilient and to guarantee a quick MTTR. That will improve your customer experience, your support satisfaction, and will keep your employees focused on doing the work they most need to do - without constant disruption for the more complicated ways that incidents can unfold.

Hope this article was useful for you! If you'd like to do some further reading on feature flags, how about reading up on 5 Common Challenges When Using Feature Flags?

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/opa-feature-flags?utm_source=pmm-ff&utm_medium=using-feature-flags-for-effective-incident-management

Introducing Global Governance Policies for Feature Flags To Deploy Faster and Safer

We're excited to announce the launch of the first-ever feature flag governance policies for Harness Feature Flags powered by Harness Policy as Code. At Harness, we focus on building solutions that prioritize reducing risk and making software delivery best practices easier and safer to adoptâall without sacrificing the velocity that feature flags bring to Engineering organizations.Â

Feature flags decouple deployment from release, giving developers a sense of control with kill switches, beta testing, experimentation, and progressive rollouts of new features. The one thing you might notice in that list of ways feature flags will help your team, though, is that a lot of it will happen in your production environmentâand things that happen in production can be scary.

To that end, global governance policies are a big win for Engineering orgs that need to enforce standards at scale. They let them put guardrails in place across all releases to ensure standards are met, and they automate the process too, so developers simply get error messages the way they're used to during the build and test phase.

The inclusion of global governance policies in Harness Feature Flags massively differentiates Harness from alternatives in the market. Harness is the only Feature Flag solution with the ability to enforce blanket governance across all feature releases, in addition to specific standards for individual or categories of releases. This global governance also extends to the rest of the SLDC on the Harness platform, so something like naming conventions (to pick an easy example) can be enforced the same across CI, CD, and Feature Flags. And for bonus points, this can all be set up once and automatically enforced without devs having to lift a finger. Instead, they'll just have errors and warnings thrown the same as if it were from a compiler or testing suite.

In addition to providing fast time to initial value and making it simple to create repeatability and scale, this enhancement notably solves the inherent tradeoff between velocity and control - now, you can have both at once.

The Governance Requirement for Scale

If you're a developer, you might recoil a bit when you hear the word *governance*. It sounds so heavy, and you might be thinking of restrictive corporate policies that stop you from being able to do things that would allow you to do your job better and faster. At the same time, you know that at some level, it's unavoidable because you can't scale a wild west of releases. Standards must be in place to ensure proper risk management.

To us, governance is the opposite; it should speed you up by creating guardrails that keep the entire team working in a cohesive and safe way. Governance is about establishing order across the system that empowers teams to move fast, build, and ship without worryingâand without scaring leadership.

Within feature flag management specifically, you want to know:

- Who can control things and where

- How to protect production in different scenarios
- How to make sure flags are ready for use
- How to keep your flags in order
- How to make sure you protect yourself by following policy

That's where governance policies help. They create order in the system that lets engineering teams and leadership answer these questions and establish guardrails around them. The key here is the guardrail. It's not designed to slow you down. In fact, it's meant to speed you up! Imagine as a dev not having to manually check flags or releases against internal policy instead, being free to commit code, knowing it'll automatically get run against policies and approvals. And for managers, not having devs worry about anything other than coding great applications.

Enforcing Global Governance Across Feature Releases

We took the popular open source OPA (Open Policy Agent) project and built it into Harness Policy as Code, complete with an intuitive UI and turnkey set of policies out of the box. OPA itself is open source and highly vetted, and we're excited to bring world class policy and governance management to the feature flag space for the first time.

We built this across the whole platform. So, how does this apply to Feature Flags specifically? The short of it is that developers can write their own Rego scripts in OPA or use our simple policy builder to create rules that will then be applied to feature flag projects. This is all managed through HPE.

How to Use Policy as Code With Feature Flags

What kinds of policies would you want to use with HPE and your feature flags? All the metadata in the feature flag system can be used to define policies, so your team has full control over what can happen, when, and how it can be used. Let's take a look at some of the ones we're providing out of the box.

You can very simply start using these policies and instantly apply these global governance standards across all of your feature flags. Whether you use our out of the box options or write your own Rego scripts, any time a feature release pipeline is run, these policies will automatically be applied.

Global vs. Local Governance for Feature Release Management

If you're familiar with Harness, you might be wondering how this differs from the governance standards you can already create within a Harness pipeline. Think of it simply as local vs global governance application kind of like local vs. global variables in code.

And even if you're not familiar with Harness, well, the same idea applies of local vs. global governance: you can apply specific policies to a type of release, or enforce global standards across all releases, like naming conventions or verifying that a feature has been promoted through all testing environments.

Pipelines allow you to set specific standards within the scope of a specific release or type of release (see below for a simple example). Where HPE policies differ is that they apply global governance across all of your feature releases, in addition to whatever specific release standards you have in place. What that means is you have the option to create two layers of governance, completely custom to your needs.

Platform-Wide Governance With Policy as Code

Let's reiterate here that while governance policies work if you only have Harness Feature Flags, they also work across the entire Harness platform. This means with Harness uniquely, you can govern CI, CD, Feature Flags, and any other Harness module all together.

This opens up possibilities like being able to say that you can't change feature flags in production if there was a failed deployment in the last hour, enabling true control over the software delivery process end-to-end for the first time.

How To Get Started

If you're already using Harness Feature Flags, you'll notice a new sidebar option labeled Policies. To start using these global governance policies powered by HPE, just head over to the Feature Flags module in your Harness console, click on Policies, and start applying your own policies right away. Once you get it set up and run it against your first release, there's no better feeling than seeing the green checkbox giving you the all clear.

If you're not using Harness Feature Flags yet, you can sign up for a free trial or request a demo to learn more.

Conclusion: This Is Just the Beginning

We are on a journey with Harness Feature Flags of building key components for governance across the entire system and tying them together. Governance policies for Feature Flags are a huge step forward, for the first time ever allowing you to truly control your feature flag system and rest easy knowing it's enforced 100% of the time. You can learn more about Harness and our governance capabilities on our platform governance page.

If you haven't signed up to use Harness yet but want to get started, you can easily sign up for free forever. Happy developing!

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/harness-policy-as-code?utm_source=pmm-ff&utm_medium=release-management-with-feature-flags

Introducing Harness Policy as Code, Powered by OPA

We're excited to announce Harness Policy as Code, powered by Open Policy Agent (OPA), a centralized policy management and rules service that empowers enterprises to centrally define and monitor policies that are enforced across all delivery pipelines and processes. Harness Policy as Code helps organizations create and enforce policies on deployments, infrastructure, and more, providing developer velocity without sacrificing compliance and standards.

Harness Policy as Code is based on OPA, an easy-to-use, extensible solution for creating and enforcing policies across the entire stack. OPA is an open source project accepted by the Cloud Native Computing Foundation (CNCF) with wide adoption across numerous software delivery use cases. Policies are written as declarative code, so they are easy to understand and modify—from simple to complex use cases.

Harness Policy as Code integrates with CI, CD, and Feature Flags enforcing automated approvals, denials, and other advanced pipeline functionality. Check out our technical documentation to learn more.

Why We Need Policy Management in Software Delivery

As DevOps is adopted within an enterprise, typically one team creates and maintains software delivery and processes. That team has full control and visibility, as they are the creators of DevOps processes within the company. As more business units adopt DevOps within the company, that originating team's manual processes can create a bottleneck, which hampers innovation by limiting team autonomy and slowing down software delivery.

Â In an effort to remove the bottleneck and increase velocity, companies can give development teams more autonomy by allowing them to drive their own DevOps processes. That decentralization of process control can lead to more risks for the company.

When governance is decentralized, development teams can miss quality checks or approvals, introduce vulnerabilities, or break

compliance. Organizations need to balance autonomy *and* governance, so they can empower teams with the confidence that they are adhering to all compliance standards and security policies â all without slowing down innovation.

Compliance becomes even more critical in regulated industries, such as financial services and healthcareânot only with enterprise standards, but with third-party regulations, like SOC2, PCI, and FedRamp. It is imperative that all software delivery pipelines meet compliance standards with full auditability; otherwise, the organization is at risk of failed audits, heavy fines, and reputational damage.Â

Centralized management and governance of policies across DevOps processes allow enterprises to define standards for the entire organization while enforcing compliance with regulations. Policies enable individual teams to have autonomy over their processes with oversight and guardrails in place to prevent them from straying from standards, ensuring secure and compliant software delivery.Â

Harness Policy as Code Features

Harness Policy as Code is a centralized policy management and rules service that leverages OPA to meet compliance requirements across software delivery. HPE enables organizations to centrally define and monitor policies that are enforced across all delivery pipelines and processes.Â

Policy as Code features for writing and enforcing policies include:

- A Policy Editor that enables developers to start writing policies-as-code quickly. With a library of policies to start from and a testing terminal, developers can try out policies on real inputs during development before enabling them.
- Policies that are configured to be automatically enforced on Harness processes (e.g. on Pipeline Run, on Feature Flag save).
- The ability to set severity, so a policy violation can issue a warning or throw an error to stop processes from continuing.
- An audit trail that can maintain a full history of policy evaluations with detailed outputs for audit and compliance.

With the release of Policy as Code, policies can now be enforced on CI and CD pipelines and Feature Flags.

Pipeline policies govern the requirements of delivery pipelines, and they can be automatically enforced when the pipeline is saved or triggered, or even in the middle of pipeline execution. Policies can enforce specific pipeline configuration, advanced access control use cases, runtime validation, and more. Here are some examples of what the Policy as Code can do:Â

- Require an approval step before deployment to production.
- Forbid use of Shell scripts in the pipeline.
- Only allow deployment to approved namespace.
- Only allow deployments from approved container registries.
- Validate test step outcome meets minimum threshold before allowing the pipeline to continue.

Policies for Feature Flags are enforced when the flag is updated or toggled on/off, enabling policies for adhering to standards, flag process, and hygiene. This includes:

- Only allowing creation of boolean flags.
- Enforcing flag naming conventions.
- Enforcing when creating a flag the default on and off values must both be false.
- Requiring a Feature Flag be enabled in QA before it can be turned on in Production.

Policy as Code centralizes and standardizes policy management across software delivery, allowing engineering leaders to empower dev teams to own their tools and practices while ensuring that everyone is following company standards for compliance and security. With guardrails in place, security vulnerabilities wonât be introduced as development teams are writing their pipelines. Leaders can rest assured that compliance standards are being met, with full auditability of policies and failures, and they can find and report breaches as early as possible with shift-left governance.

Get Started

Check out our platform governance page to learn more about how Harness' modern approach to software delivery governance empowers teams with stable processes that donât slow down delivery, or request your personalized demo today.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/feature-flags-analytics-business-intelligence?utm_source=pmm-ff&utm_medium=using-feature-flags-for-effective-incident-management

Announcing Feature Flags Analytics & Business Intelligence

We're thrilled to announce the introduction of analytics & business intelligence for Harness Feature Flags. Users of Feature Flags - which can be anything from simple software switches to hide or activate features to powerful, automated release strategies - can now gain the most complete visibility into their feature flags usage and operations with custom analytics offered by any feature flags solution.

The Harness Software Delivery Platform supports Continuous Integration, Continuous Delivery, and Feature Flags for development and operations teams. With this new capability, Harness users can bring together analytics across the entirety of the software delivery lifecycle in a way no platform has allowed to date. It's true business intelligence for software delivery.

Leverage analytics to gain visibility into the usage and adoption of feature flags across your teams and drive strategic improvements in these areas, as well as conduct audits for governance and compliance, and identify ways to improve feature release processes.

The Need for Visibility

When we announced Harness Feature Flags earlier this year, we started with a developer-first experience, automated progressive delivery, and visual feature release pipelines. We wanted to ensure that we could build powerful analytics to provide deep visibility into every aspect of users' feature flags solution.

Moreover, understanding metrics like false positive rate helps teams evaluate the efficiency of their flags. Getting deep visibility into individual flags' usage is one thing, but what about visibility across the whole solution?

Without this capability, we found that users struggled to solve these problems:

- No high-level view into how teams are using feature flags.
- Who's doing what with flags?
- Who's creating flags?
- How many flags exist?
- Is the solution actively used?
- Slow audit process, done manually per flag.
- Lack of data to demonstrate utilization or maturity of their feature flag solution.
- Can't relate feature flag metrics to changes across the SDLC.

It's not just about visibility, though. It's about complete access to and control over all of the data contained within the feature flags implementation. Imagine being able to test in production and see how the system is performing or what's being done across any dimension or metric that you can think of - like using a database query, but visual.

Want a high-level view that's scheduled to send to your boss every Monday? How about details into how many flags of a specific type have been evaluated for a customer? These use cases, and many more, are easily visualized using the business intelligence capabilities now available.

Analytics & Business Intelligence for Feature Flags

Harness provides an out-of-the-box dashboard that provides users a quick glance across multiple dimensions that are commonly used by developers, engineering managers, and product teams. Users can see the details of their feature flag implementation in the following ways in this dashboard:

- Feature flags by environment;
- Feature flag type and status;
- Flags by user and target;
- Creation history.

In addition to this base dashboard, users can create new dashboards that are fully customizable across any metric or dimension. Users are also able to customize schedules and alerts within each dashboard, ensuring that **users can automate reporting and be alerted just-in-time about unexpected behavior as it occurs**.

To get your brain jogging, here are some additional examples of things you may want to understand as a user:

- Mix analytics across SDLC if you're using other Harness modules.
- See analytics and usage - creation, evaluations.
- See flag changes in production for compliance needs.
- Schedule reports and create alerts based on custom triggers.
- Allow teams to answer questions that include:
 - Can I see flags by team and other cross-sections?
 - How much activity with flags in production?
 - Who's creating flags?
 - Who are flags targeting?
 - Did my team make any new flags this week?
 - How many flags do we have?
 - Who isn't regularly using flags, and who's using them a lot?
 - Where do we have a lot of activity?
 - All prod flag changes for governance or compliance needs.
- This can also be a weekly report your compliance team can get on Monday mornings.
- Seeing what new flags are created.
- E.g. see all flags added last 7 days.
- Who made them?
- How many evaluations?

Added Value of the Harness Platform

As you might imagine, the combinations and details you might want to see are endless, and with analytics & business intelligence with Harness Feature Flags, your imagination is all that holds you back. For users of additional product modules from Harness, there are a few more use cases that are unlocked!

To share one example, you'd be able to see that you did 90 CI builds, 17 deploys across 3 services, made 7 new flags, and enabled 4 of them across 4 customers.

- Create a canonical SDLC DORA metrics dashboard - seeing how CI+CD+FF work together to improve these key metrics.
- See for a team, or the whole organization, everything from code to feature delivery:
 - Velocity;
 - Blockers;
 - Bottlenecks;
 - Underutilization;
 - And more.

How to Get Started

If you're already a Harness user, you can head over to the Dashboards part of the platform. There, you can filter by Feature Flags to find the out-of-the-box dashboards, or create your own!Â

And if you haven't signed up to use Harness yet but want to get started, you can easily sign up for a free trial. Happy developing!

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/hierarchical-breakdown-of-organizational-cloud-costs?utm_source=internal&utm_medium=blog&utm_term=ccm-chargeback-showback-build-buy

Hierarchical Breakdown of Organizational Cloud Costs

Businesses leveraging public clouds often grapple with understanding and managing their cloud costs. Traditional methods and tools might offer insights, but in today's complex cloud environments, deeper visibility is essential.

Without a clear hierarchical breakdown, costs can easily spiral out of control without any accountability or ownership. Costs can become murky, teams can exceed budgets unknowingly, and companies can waste budgets on underused or redundant cloud resources. Therefore, getting a handle on these costs by understanding them at every tier of the organization is vital.

Harness Cloud Cost Management for Cost Visibility and Attribution

Harness Cloud Cost Management (CCM) offers two powerful cost attribution features â Cost Categories and Perspectives. These features offer a comprehensive view into cloud costs that is as rarely implemented as it is essential. Cost categories offer a way to group cloud costs in ways that are most meaningful for their business and act as the basis for creating and managing perspectives, which are cost views customized to the needs of your business that empower teams to track and manage costs directly.

Cost Categories

Users can group cloud costs in ways that are most meaningful for their business. This not only facilitates clearer budgeting but also provides an in-depth breakdown. Such insights offer clarity on which departments, projects, or applications are consuming the most resources and money.Â

It also ensures that costs across cloud providers and Kubernetes deployments can be included in the same cost categories to provide a complete view of costs, without requiring manual report creation.

Perspectives

Harness CCM offers a flexible approach to viewing and allocating costs. It offers various perspectives that can be tailored to roles, departments, projects, or any other organizational structure. This ensures that everyone gets the most relevant and contextual view of their cost data.Â

Each perspective gives that team direct access to cost forecasting, relevant recommendations to optimize costs they are responsible for, the ability to set cloud budgets, and get direct alerts on anomalous cloud spends. All of this is in the context of the cloud costs that their Perspective is tracking to ensure they get the information they need to track and take action on their cloud costs.

Benefits of Hierarchical Cost Breakdown

- **Transparency** - Organizations gain a transparent view of how cloud spend maps to resources consumed, and can make data-driven decisions.
- **Accountability** - Departments or teams can be held accountable for their specific cloud expenses with more granular and accurate

cost allocation.

- **Optimization** - Identifying which resources are not allocated to any team or department becomes straightforward, leading to cost savings. As they say, you can't improve what you don't measure.
- **Forecasting** - With a clear view of current costs per team or department, predicting future expenses across the organization becomes more accurate.

Real-World Application Example

Let us understand how you can leverage cost categories with a case study. Let's assume we have an Organisation X, with 3 levels of hierarchy at which cloud costs are tracked; VP, Director and Manager. Multiple Managers roll up to a director and multiple directors roll up to a VP. Each manager is responsible for a specific team.

Cost categories are used Cost Buckets as the building block for creating the category. Each Cost Bucket is defined by a set of rules, which can be as simple as pulling from a team tag from a single cloud to as complex as multiple rules pulling data from across all clouds spanning across multiple different rules. This ensures that all costs per team, regardless of source, are included in their cost category.

We'll start at the bottom of the hierarchy, at the Teams category level, and include all costs for each teams cost bucket.

Once the Teams have been defined, we can then create a hierarchical (nested) cost category for Director, which will include the Teams that report into each Director. Following that, we can repeat the process to create the VP Cost Category, defining the Directors that report into each VP.

Once the cost categories are created they can be used to define new perspectives and further breakdown costs in existing perspectives. This can give you high level visibility at multiple levels of the business, at the VP, Director and manager levels, all the way down to the individual teams.

These cost category and cost bucket definitions can also include attribution of costs for shared cloud services. Sharing of costs across cost buckets can be done using various strategies such as proportional cost sharing, sharing costs based on a fixed percentage or sharing costs equally across the different cost buckets.

Without a tool like Harness CCM, understanding which team's activity incurs what costs, including shared costs, can be challenging. However, with Cost Categories and Perspectives, each VP, Director, team, and the finance team, can get a clear and accurate breakdown. They can see costs associated with each department, assess the ROI, and allocate budgets more effectively for the future.

Conclusion

In today's multifaceted cloud environment, having a tool that offers a granular, hierarchical view of costs across your company isn't just a nice-to-have—it's essential. Harness CCM's features of Cost Categories and Perspectives provide this much-needed visibility, ensuring that businesses can use cloud resources effectively without busting their budgets.

Looking to get this kind of granular visibility into your company's cloud costs? Take control of your cloud expenditure and gain unparalleled insights with Harness CCM. Book a demo or sign up for free today.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

[Documentation](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[Case studies](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer

happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/flag-pipelines-to-automate-feature-flag-governance-and-daily-management?utm_source=pmm-ff&utm_medium=using-feature-flags-for-effective-incident-management

Announcing Flag Pipelines to Automate Feature Flag Governance and Daily Management

When we launched Harness Feature Flags, we were excited about the opportunity to leverage other capabilities across the Harness platform to create better-than-the-sum-of-its-parts value. One of the first things we did was add support for feature release pipelines to Feature Flags, which enabled users to automate software release workflows and policy enforcement.

Today, we're extending how you can use pipelines with the launch of a new concept, Flag Pipelines. Pipelines have always let you build powerful workflows and automation for doing releases with feature flags, and now development teams can run pipelines automatically on every flag change to enforce governance and control. That's right — flags can have individual pipelines that run every time a flag state is changed. That's a game-changer for automating the day-to-day usage of flags and making sure that every change meets organizational standards.

Essentially, you can now guarantee that any time a flag's state is changed, it goes through a predefined workflow. This can be something as simple as requiring an approval every time a sensitive feature is toggled. Or, it can be something more complex, such as requiring an approval, making an update to Jira and ServiceNow, and sending notifications to customer support and managers.

Governance in the Day-To-Day Flag Lifecycle

Flag pipelines let you build a simple pipeline and assign it to a flag. From that point on, that pipeline will run automatically every time the flag changes, no matter how it changes. This means that whether you change it via the UI, API, or by updating the YAML via Git, you'll see this pipeline triggered.

This is great addition to feature management when your team wants to do things like:

- Require an approval from a manager role, or a PM, on any flag change in the production environment
- Send a Slack message on every flag change
- Update a Jira ticket being used to track feature development steps
- Execute a custom script associated with the flag every time the flag changes

Manual Process Misses the Point Of Feature Flags

Normally, you'd use pipelines with feature flags to build automated release workflows, such as incrementing a flag to 10% of your audience with an approval every 23 hours. And you can still do that for your one-off or specifically-modeled release scenarios. Flag Pipelines let you automate and enforce the things that you want to have happen *every time*, not just when you have a specific one-time scenario.

For teams and broader organizations, this feature enhancement provides a broader level of control over how feature flags are handled and managed day-to-day. Previously, users would have had to create an organizational process that required all of the users to run a stated pipeline every time that a flag needed to be changed. This poses a few problems:

It's not that teams don't try to do these things the right way, rather that things slip, people and processes change, and it's inconvenient to follow numerous extra steps for something that should be simple. Instead, what if users could use flags in a way that's more closely aligned with their actual needs, and the release process for flag changes is automatic? By flipping the script, devs aren't slowed down, and the business makes sure that things are done the right way every time.

How to Use Feature Flag Pipelines

â

To set up your flag pipelines, just select any flag and select the new "Flag Pipelines" tab. From there, you can choose or create the pipeline that you want to use.

Because flag pipelines are meant for the subset of workflows that you want to execute every time, not all pipelines can be used as flag

pipelines. Flag pipelines must:

- Not have continuous integration/continuous development (CI/CD) steps
- Only have one flag change, not multiple
- Have the flag step set to runtime inputs.

Once you have a suitable pipeline and have it selected inside of your Flag Pipelines settings within a flag, you're all set. That pipeline will now automatically run every time that the flag is changed within that environment.

Find more details about how it works in the Harness docs.

Get Started with Harness Feature Flags

Flag Pipelines continue our work in the pursuit of turning feature flags into a true part of your software delivery process. Allowing more sophisticated control of feature flags brings increased governance and feature flag automation to your organization. It's a big step forward in evolving feature flags into a key business process in software delivery.

Ready to get started? Request a demo or sign up for free forever. We'll continue to expand on this in the future and would love your feedback on how you'd like to see this story evolve.

Happy developing!

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

[Documentation](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[Case studies](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[We want to hear from you](#)

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

[Sign up for our monthly newsletter](#)

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/what-are-feature-flags?utm_source=pmm-ff&utm_medium=how-we-use-our-own-feature-flags-at-harness

What are Feature Flags and How Do We Use Them?

Building software is a great experiment. Many times we are required to do what has not been done before; this is core to innovation work. However, a lot of the iteration and experimenting tends to happen before software is shipped. That's where the concept of test in production comes into play. Building something from scratch or even jumping into an older project mid-stream requires lots of iteration.

Once we deploy, for many software engineers, the experimentation on the existing tends to drop off and onwards to the next set of features in the backlog. Product owners/managers typically collect feedback to help prioritize the next set of features. From a software engineering perspective this can feel a little âchicken or egg-ishâ - without a feature actually deployed in a production environment to get feedback, how do we if know we are on the mark?

Letâs illustrate this with an example. Customers are asking for an updated UI view for their dashboards; they want something that surfaces the most relevant information faster. Now that the product team has collected those requirements, itâs up to engineering to figure out how to best build it. There might be a few potential solutions:

With some potential solutions decided, now itâs up to the engineering team to decide which of the three to implement. And before a customer ever sees it, they have to choose 1 of 3, build out the MVP over the next few sprints, and then ship it. This is the first time theyâll get any feedback, and if itâs not the right solution then it has to be built again!

Now, imagine if this new feature cycle could be improved for product development teams. If youâve been paying attention to developer tools over the last few years, you might have seen a new category popping up: feature flags. Feature flags as a concept were born to increase development velocity and to solve this very problem through practices like progressive delivery.

While feature flags as a mature category of hosted tooling is new, feature flags - and what they solve - have been around a little longer. Letâs take a look at what feature flags have grown into, and how to use them.

The Story of Feature Flags

Feature flags, at their core, are the ability to wrap different versions of your code in conditional statements that you can turn on and off at will. You may have previously thought of them as app configs, or even part of your companyâs rules engine.Â

With Harness Feature Flags, weâve taken these concepts you may be familiar with and taken them even further - adding more sophisticated rules, governance, user management, an intuitive UI, and more.Â

Feature Flags lets you separate feature release from code deployment, build more effective strategies for rolling things out and testing with your users, and have more insight than ever into how changes impact your application performance.

Ultimately, feature flags give engineering teams the ability to deliver more features, with less risk. For leaders, itâs a great way to improve engineering velocity and developer experience, and to reduce the risk associated with feature development. And for developers, itâs a great way to work more efficiently, with less stress - think fewer late nights before a deployment, and fewer firefights post-deploy.Â

What Are Feature Flags (or Feature Toggles)?

Feature flags are a way that developers can conditionally turn certain sections of their code on or off. You can think of feature flags as extending Continuous Delivery into Continuous Deployment - a way to put changes into production behind a flag and turn them on in a controlled way later (or hide and remove them in the same way).

But you can also think of feature flags as a new way to think about building and releasing applications - by clearly defining features and components that can be changed and toggled on and off individually, you allow for experimentation, controlled rollouts to different users, as well letting more people (like PMs, sales, and support) turn things on and off for customers.

Together, this results in giving more power and control to developers to create more and better features faster, while at the same time ensuring that engineering is not the bottleneck for the business in having to make changes for individual customers or ensuring a perfect production deployment for every new feature or feature update.

If we want to simplify even further, feature flags essentially create private swim lanes for developers where they can ship a feature directly to customers and then have control over who sees it, get feedback, and turn it on and off as needed. Itâs like a light switch (a pretty complex one)!

Who Uses Feature Flags?

People oftentimes consider feature flags either an engineering tool, or a tool for product managers. The reality is, itâs both. Flags can help software development and DevOps teams lower their overhead and increase their velocity, and they can help product managers better control releases, coordinating launch timings and creating a feedback loop more effectively.

Research indicates that 95% of engineering leaders want to implement a feature flag solution for themselves. This comes on the back of 97% of leaders saying theyâre under pressure to deliver faster, and 65% of whom say they find it difficult to achieve that velocity increase in a safe manner. Itâs abundantly clear that engineering teams are likely the biggest beneficiaries of implementing and using a feature flag solution, and indeed, they tend to be the primary users. And when they do implement a feature flag solution, they end up delivering 66% more features per application per year!

Feature flags are great for users across the organization beyond product and engineering. For sales and support teams, feature flags provide a way for them to directly manage betas and new features for their customers, and to track down whoâs using what in case of any issues. And for management, flags can provide new forms of visibility into whatâs happening in development and how new features are being tested with users.

On a âfun factâ kind of note and since weâre in the âwho uses feature flagsâ section, itâs cool to note that big companies like Netflix,

Flickr, Google, Reddit, and more use feature flags! They're definitely worth looking into and absolutely are an important part of the software delivery lifecycle.

Why You Should Consider Feature Flags

Whether or not the use of feature flags is important to your software delivery lifecycle comes down to what it is that you need to accomplish. While feature flags are a great tool to have for any team, they are particularly useful if you have these problems:

Problems in the Business

- The risk associated with new feature releases ends up slowing down developers since they have to thoroughly validate design and implementation before releasing.
- There are often multiple possible versions of how a new feature will look and function, and it's up to the developer to choose one (not always the right one) and implement it.
- The decision of when to release new software features is controlled by developers instead of the business.
- Sunsetting old features introduces risk to newer features that might have dependencies on the old code.
- It's all or nothing - a feature is rolled out to everyone at the same time instead of doing a feature rollout incrementally for verification purposes.
- When teams are beholden to engineering release cycles (e.g. once a month), multiple features are released simultaneously, introducing complexity and risk that can result in deployment war rooms, large rollbacks, and dissatisfied customers.

Problems in Engineering

- Releasing a feature takes too much coordination and time across stakeholders.
- Merging new features, or managing separate feature branches for changes, is cumbersome and complex.
- You want to test changes to see how they impact your application and end-users before rolling them out more widely.
- You want to provide access to new features, or to test changes with certain portions of the user base sooner - by location, by customer plans, by customers opting into beta testing, or by any other criteria or logic you can think up.
- You want to reduce the risk of rolling out new code by having instant kill switches that don't require rollbacks or new deployments to fix a disrupted environment.

To simplify, we can consider that feature flags are a great solution if you're trying to do the following things:

- Increase engineering velocity and ship more features.
- Decrease feature development risk by A/B testing multiple solutions with customers.
- Reduce the risk and toil associated with production failures and rollbacks.
- Decouple engineering deployments from feature unveilings to customers.

Different Types of Feature Flags/Toggles

It's natural to want to put all feature flag use cases in a single bucket, but it's more helpful to instead view flags themselves as a means to an end and as having categories of use cases.

Here are some examples of the different kinds of flags that could be used in a system:

- **Release features** - Using flags to let the right people decide who to turn a feature on for, and when, without complex engineering deployment coordination required.
- **Experiment** - Using flags to learn something about how a change impacts things. This can be a technical experiment, such as how a new UI impacts server load, or it can be a user experiment such as how a new button impacts conversion.
- **Ops flags** - Letting you have permanent control around key parts of your app so you can turn them on and off as necessary without a full new deployment cycle. Add in RBAC, and this is a robust way to control who has access to modify your production application.
- **Control access** - Using flags as a way to manage betas, early adopter list, trial access, and more.

Implementing Feature Flags

Implementing feature flags can seem really simple at first, but it turns out that like with any system, the devil is in the details. Building an awesome feature flag solution in-house certainly has its perks, but it can be deceptively difficult. In particular, we've seen three distinct sets of issues arise with feature flag solutions as an organization's implementation matures.

When a feature flag solution is **nonexistent or immature** and an organization is standing it up for the first time, they tend to run into issues with finding value in the tool quickly. Especially with the need for such a solution and the pressure to make it happen quickly, it's critical that value is demonstrated quickly, otherwise it may end up leaving a sour taste. Within that, there's a dichotomy of architecting for eventual scale versus getting an MVP out the door to prove value. This often results in complex design or architecture that isn't scalable and only works for small, targeted use cases.

As feature flag solutions enter a **middle stage of maturity**, or as they **start to scale**, the problem is quite obvious: scaling is hard. Especially if built in-house, ensuring that the tool can be used by multiple users and multiple teams for a variety of use cases, all while maintaining a common set of best practices and building on a single infrastructure can be incredibly daunting. And as this is being figured out and teams have code in production, it can quickly result in stale code or issues keeping production tidy when code changes occur. Add onto that the need to integrate neatly into everyone's workflow and you've got a behemoth of a problem to solve.

Building on that is the scenario we often see with **highly mature** feature flag implementations, where they primarily end up as separate systems from existing software delivery or CI/CD pipelines that teams now have to learn how to use, and to figure out how to port their existing processes or pipelines into the feature flag solution. At this stage, there's the obvious cost of maintenance and future development to keep up with new changes or requirements. Technical debt is bound to build up by this point, and it can become unmanageable without a dedicated team - and perhaps even its own codebase.

Sure, 95% of engineering leaders may want a feature flag implementation, but as it turns out, it's not so easy to build something that works well over time.

Build or Buy: Feature Flag Management

You may be thinking, "Feature flags seem pretty simple, can't we just build this ourselves?"

The real answer? It's complicated.

We've seen in most cases that an internally-built tool will end up causing a lot of overhead and putting a very low ceiling on the benefits you can receive from feature flags. It may be good for your first couple of flags, but it can be hard to make an internal system scalable, performant, and robust for all the things flags can do as your organization gets more and more comfortable using them - and more reliant on them.

For a deeper look at the build vs buy conversation, you can check out an article we wrote earlier this month, Feature Flags: Should I Build or Buy?

Why Use Harness for Feature Flag Management

Harness provides a feature flag platform made for getting you up and running quickly while being able to scale to any level. We make it easy to get started and quickly see the benefits of feature flags. With Harness Feature Flags, that means you can get your first flag out the door in minutes by following three steps:

- Create a free trial account.
- Create a new flag in just 3 clicks.
- Add one of our SDKs to your code in a few minutes.

And, you've got a flag working!

If that's not quite clear, then it's worth knowing that the fundamental way flags work in an environment like Harness is that SDKs, added to your application as a library, will receive updates on flag statuses to evaluate with your users (you should choose between real-time streaming or pulling updates on intervals). These SDKs are secure, collecting no data you don't want to share, and they are designed for performance and stability.

Harness Feature Flags is fast, secure, and focused on helping teams get the most out of feature flags - from your first initial use all the way to global governance, sophisticated reusable templates, and developer-first experiences.

And if you're wondering how we stack up against LaunchDarkly, Optimizely, Cloudbees Feature Management, Split, and other FF tools - those pages are being worked on right now and we'll update this post with the links when they're live!

If you'd like a little more information on our architecture and on how to get started, you can check out the feature flag documentation or request a demo.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

[Documentation](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer

happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/flag-pipelines-to-automate-feature-flag-governance-and-daily-management?utm_source=pmm-ff&utm_medium=release-management-with-feature-flag?utm_source=pmm-ff&utm_medium=the-journey-to-using-feature-flags-at-their-full-potential

Announcing Flag Pipelines to Automate Feature Flag Governance and Daily Management

When we launched Harness Feature Flags, we were excited about the opportunity to leverage other capabilities across the Harness platform to create better-than-the-sum-of-its-parts value. One of the first things we did was add support for feature release pipelines to Feature Flags, which enabled users to automate software release workflows and policy enforcement.

Today, we're extending how you can use pipelines with the launch of a new concept, Flag Pipelines. Pipelines have always let you build powerful workflows and automation for doing releases with feature flags, and now development teams can run pipelines automatically on every flag change to enforce governance and control. That's right — flags can have individual pipelines that run every time a flag state is changed. That's a game-changer for automating the day-to-day usage of flags and making sure that every change meets organizational standards.

Essentially, you can now guarantee that any time a flag's state is changed, it goes through a predefined workflow. This can be something as simple as requiring an approval every time a sensitive feature is toggled. Or, it can be something more complex, such as requiring an approval, making an update to Jira and ServiceNow, and sending notifications to customer support and managers.

Governance in the Day-To-Day Flag Lifecycle

Flag pipelines let you build a simple pipeline and assign it to a flag. From that point on, that pipeline will run automatically every time the flag changes, no matter how it changes. This means that whether you change it via the UI, API, or by updating the YAML via Git, you'll see this pipeline triggered.

This is great addition to feature management when your team wants to do things like:

- Require an approval from a manager role, or a PM, on any flag change in the production environment
- Send a Slack message on every flag change
- Update a Jira ticket being used to track feature development steps
- Execute a custom script associated with the flag every time the flag changes

Manual Process Misses the Point Of Feature Flags

Normally, you'd use pipelines with feature flags to build automated release workflows, such as incrementing a flag to 10% of your audience with an approval every 23 hours. And you can still do that for your one-off or specifically-modeled release scenarios. Flag Pipelines let you automate and enforce the things that you want to have happen *every time*, not just when you have a specific one-time scenario.

For teams and broader organizations, this feature enhancement provides a broader level of control over how feature flags are handled and managed day-to-day. Previously, users would have had to create an organizational process that required all of the users to run a stated pipeline every time that a flag needed to be changed. This poses a few problems:

It's not that teams don't try to do these things the right way, rather that things slip, people and processes change, and it's inconvenient to follow numerous extra steps for something that should be simple. Instead, what if users could use flags in a way that's more closely aligned with their actual needs, and the release process for flag changes is automatic? By flipping the script, devs aren't slowed down, and the business makes sure that things are done the right way every time.

How to Use Feature Flag Pipelines

â

To set up your flag pipelines, just select any flag and select the new "Flag Pipelines" tab. From there, you can choose or create the pipeline that you want to use.

Because flag pipelines are meant for the subset of workflows that you want to execute every time, not all pipelines can be used as flag pipelines. Flag pipelines must:

- Not have continuous integration/continuous development (CI/CD) steps
- Only have one flag change, not multiple
- Have the flag step set to runtime inputs

Once you have a suitable pipeline and have it selected inside of your Flag Pipelines settings within a flag, you're all set. That pipeline will now automatically run every time that the flag is changed within that environment.

Find more details about how it works in the Harness docs.

Get Started with Harness Feature Flags

Flag Pipelines continue our work in the pursuit of turning feature flags into a true part of your software delivery process. Allowing more sophisticated control of feature flags brings increased governance and feature flag automation to your organization. It's a big step forward in evolving feature flags into a key business process in software delivery.

Ready to get started? Request a demo or sign up for free forever. We'll continue to expand on this in the future and would love your feedback on how you'd like to see this story evolve.

Happy developing!

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/targeted-rollouts-release-progression?utm_source=pmm-ff&utm_medium=how-we-use-our-own-feature-flags-at-harness

Targeted Rollouts and Release Progression

When you want to try a new Harness feature out, we enable it for you via a feature flag. Our support/account team turns it on for you.Â Typically, for new features, our release process happens in multiple phases. First, we test internally in lower environments. Second, we test internally in production. Third, we test with a few beta customers in production for a period of time leading to a wider release. Finally, for some features, we release more widely - but still in beta, and often still with some restrictions on who can access them. This is all an example of using targeted rollouts via feature flags, and then progressing releases over time as you learn and respond. We will dig into that further in this blog.

Why Would I Want to Do That?

Feature flags are often thought of as being on the continuum of CI/CD, and thereâs good reason for that. Feature flags de-risk your deployments, let you merge and ship faster, and reduce the need for long-running feature branches. In this way, feature flags level off the classic goals and outcomes of CI/CD.

However, feature flags are not only an extension of CI/CD. They are also a separate process that sits *alongside* CI/CD, enabling new behaviors or *removing* those behaviors from the CI/CD loop entirely. This simplifies CI/CD and also provides you and your customers with new and better release options.Â

When thinking of feature flags this way (as a process that runs alongside CI/CD rather than one that sits on top of it), targeted releases and release progressions are high-value behaviors that immediately emerge.

Targeted releases, specifically, mean turning your change on for a specific subset of your audience initially, and then progressing it to more users if/when you're ready. Or, alternatively, turning it on for this subset and then turning it off right away if there are issues.Â This method is especially useful for reducing the amount of project management needed, where you might want to incrementally introduce changes and measure their effects with very low overhead.

Using feature flags like this allows you to stress test changes, have PMs or other folks drive alpha and beta feature feedback, control releases by customer level or region, and work closely with design partners. All without needing to rely on engineering day-to-day for a deploy/rollback cycle. Or to enable/disable things for the targeted users.

How Is it Different Than a Canary Deployment?

We previously wrote about how features flags and canaries should be thought of as working together, not an either/or. But in the context of targeted releases, itâs worth revisiting the differences between feature flags and canaries.

Canaries are a fantastic way to *test the code or the artifact* for system anomalies, performance issues, scale, and other similar metrics - and roll back as needed. However, feature flags can help you *test the change, not just the code*.

What does *testing the change and not just the code* mean, specifically? It means that the code working as intended doesnât mean the feature is working for users. Does it do what we need for our customers? Can they find it? Can they understand it? What questions do they have? Avoiding a type 2 error---where you mistakenly accept something as fine when it's not---is key.

If we're in beta of a feature and we're trying to prioritize the next batch of work in the most useful order, having this kind of feedback guiding what we do next and what has the highest impact is critical. Itâs not about âDoes the code technically work?âÂ

This is an example of where we want to think about feature flags not as an extension of CI/CD, where the point is simply releasing the change without risk, but rather as a standalone part of our software development process where we are constantly enabling and disabling things to get feedback, learn, and respond while we are building and deploying every day.

Internal Examples

Iâve already talked a little bit about how we use Harness Feature Flags internally, to turn things on for our customers. Thatâs one example of how we use targeted rollouts here at Harness. Though, it is one that often makes Slack messages difficult to parse since sometimes we use Harness Feature Flags to turn on a feature inside of Harness Feature Flags for Harness Feature Flag customers (whew!).

There are some other great ways we use targeted releases, though.

- We may turn a feature off for all on-prem customers, but not have to worry about maintaining more complex conditional code or separate builds.
- We can turn something on for our Enterprise users without yet having it wired to the account framework that enforces plan limits, which lets us learn more about the feature earlier in the process.
- If a feature is early in development, we may give access selectively to our sales engineering team so that they can work with partner customers in a hands-on way to demo the feature, get feedback, or drive conversation around where we are going.

And thatâs just how we use targeted rollouts. Weâve heard of plenty of other teams that turn things on and off per region, to test performance or scale. Weâve also heard of customers turning things on and off for specific infrastructure or clusters, to help with migrations or to get data about possible configurations. Thereâs a lot you can do that isnât immediately obvious.Â

Conclusion

Using feature flags for targeted rollouts is a powerful way to expand how your organization builds software, learns from customers, and

de-risks changes along the way.

Most organizations, when starting with feature flags, have the idea in mind that feature flags require the right use cases to be useful. They often struggle to know where to start. However, thinking of targeted releases helps explain that the more things are behind flags by default, the more options you'll have in the future.

You can't always predict what, when, and where you will want to target. The mindset of targeted releases is about knowing you have the capability to do so, so that as scenarios, questions or needs come up, your organization is able to learn and react.

Want to keep reading about feature flags? Why not check out our pieces on Kill Switches and How To Get Started With Feature Flags?

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/feature-flags-policy-governance-tutorial>

Feature Flags Policy Governance Tutorial

Harness Policy as Code, powered by Open Policy Agent (OPA), allows you to write policies as code to automatically govern flag configuration, usage, and process. Policies are applied on every flag creation, toggle, or modification, ensuring your feature flags always stay in a compliant state.

We are proud to be the first feature management solution to support this capability. Global governance policies for feature flags are a big win for engineering organizations that need to enforce standards at scale. They allow for guardrails to be put in place across all releases to ensure standards are met, and they also automate the process, so the developer experience doesn't change. Developers simply get error messages the way they're used to during the build and test phase.Â

But this post isn't about why you should do it â it's about how you do it. In this post, we go in depth and teach you all about the Feature Flags policy governance: architecture, use cases, data available to utilize, and more.

Architecture

Here's a quick overview of how we ensure that your flags always remain in a compliant state, regardless of what changes a user makes or how they decide to make them.Â

Writing Policies

The Harness Policy Engine is based on OPA, an easy-to-use, extensible solution for creating and enforcing policies across the entire stack. OPA is an open-source project accepted by the Cloud Native Computing Foundation (CNCF) with wide adoption across numerous software delivery use cases within the CI/CD pipeline. Policies are written in Rego as declarative code, so they are easy to understand and modify from simple to complex use cases.

You can find more information about writing Rego rules in the official OPA docs.[Â](#)

Once you're ready to get a policy out there, you can write your own and add it to Harness. We also provide some policies out of the box for common use cases, like enforcing naming conventions and ensuring proper promotion of code.

What Data is Available to Write Policies Against?

We know that policies not only relate to the changed entity in question, but can also depend on contextual data, such as who made the change and when. To support these use cases, we provide an ever growing collection of data to the policy engine that you can use when writing your policies. This includes:

- The full feature flag configuration for the created/updated flag. This includes flag name, description, variations, rules, flag state in every environment you have, and much more.[Â](#) This is in json format, which matches the format of our public API docs, so if you interact with your flags as code already, it will be very familiar.[Â](#)
- Which user made the change, what RBAC permissions they have, and which user groups they belong to.
- When the change was made.[Â](#)

Here's a truncated example of the metadata sent to the policy engine

Debugging Policies

Policies are written as code, and as anyone who writes code knows, there will inevitably be lots of edge cases or error scenarios you want to test while refining your policies. Having to create and configure flags in special ways just to test your policies is^a frustrating and time consuming. For this reason, we have an integrated policy tester/debugger built right into the UI.[Â](#)

Using this debugger, you can go through a standard development process:

- **Develop:** Begin by writing your policies in the main text area. Add as many rules and helper functions as you'd like.
- **Test:** Using the testing terminal, you can get quick feedback on if the new policies you're building are having the desired effect. You can hit the "Select Input" button to populate the input field with real flag data from your own project, giving you reliable and realistic test scenarios for your use cases. You can then hit the "Test" button to run the policies and see for the given data input if it would succeed or fail. You can also manually edit this input^a data if you'd like to hand craft particular edge cases.
- **Debug:** Once your policies are being enforced there may be a time that a user isn't sure why a change was blocked. This can be a frustrating experience if you're only provided with a vague error message such as "change forbidden."[Â](#) Luckily, we provide all the tools for a user to view detailed information on exactly which policies failed, along with the ability to click on these policies and enter this debugger mode, viewing the policy and the exact input their change produced. This will help them verify if it's a valid rejection or if the policy itself needs to be modified going forward.

Use Cases

Harness Policy Engine supports a wide array of use cases. These range from simple sanity checks that description fields have been properly completed to complex corporate policies that prevent changes during blackout periods. Feature flag policies broadly fall into these two categories:

Flag Configuration

- Naming conventions, e.g. flag names must match jira ticket format
- Mandatory descriptions
- Only allowing boolean flags to be created (no multivariates)
- Banning certain functionality e.g. no prerequisite rules allowed
- Max number of specific target rules

Change Management^a

- Flag cannot be enabled in production unless it is enabled in QA first
- Flag changes must be made via pipelines with an approval step
- No changes during certain time periods e.g. when tests are running or during certain mandated blackout periods
- Flags can't be enabled by the same user that created it

You can get as creative as you'd like around the rules you need to enforce. We've purposely not taken an opinionated UI wizard-based approach on how to create and combine these rules, so you're free to experiment, start small, and govern what matters to you.[Â](#)

Where Can I Try It Out?

To learn more about how Harness manages policies, check out our Policies Overview for Feature Flags documentation, or sign up for a free Feature Flags trial today.Â

â

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/kill-switch-code?utm_source=pmm-ff&utm_medium=using-feature-flags-for-effective-incident-management

6 Use Cases for Kill Switches in Production Using Feature Flags

If you've only come to feature flags recently, you may not know that feature flags as a concept has existed for quite a long time, and can easily be seen as an evolution of app config systems. But first things first - if you need a refresher on feature flags, head on over to our "What Are Feature Flags?" article now.

One of the least used - but most powerful - use cases of kill switches using feature flags can be for Ops and SRE teams. Plenty of folks never think about this potential, but it can be a way in which feature flags can most transform your engineering organization. They're a great way to increase velocity and reduce risk in software development.

Kill switches create tremendous potential both to control granular capabilities or features within an application, and also as long-lived operational switches that affect the entire application. And while feature flags are intuitively thought of as affecting frontend or client-facing applications, they can also be used just as well on the backend. Let's explore how you can leverage kill switches to change the nature of how your team can deliver software.

Use Case 1: Control in Production With Feature Toggles

Consider this the base case of implementing kill switches using feature flags. The concept is simple enough: you can rig any and all features and changes pushed to production (or any environment) with a toggle, or a kill switch. Feature flags, by nature, allow teams to isolate features within their applications so that they can be handled individually instead of as a deployment bundle. Kill switches

themselves are the way in which teams can create a control to disable code in production at any time.

This basic kill switch functionality gives teams the ability to turn off a feature when it's not ready. "Not ready" can mean two things in prod:

Unfinished Features

The first condition for not being ready is what slows down many teams during the deployment process. In particular, as many teams work on different features across different applications, integration and deployment take a long time. Without having everything rolled in, release teams and the software development teams that built them both find themselves slowed down.

Using feature flags as kill switches in this context allows teams to push new features to production even as incomplete features, which can result in faster development cycles. It simply needs to be pushed live, but with the kill switch activated (feature turned off) so that it's not actually impacting anything.

Broken Features

Here's a developer favorite: it worked in all the pre-production environments, and then a bug was discovered that's now impacting production. Without the use of a feature flag, that broken feature impacting prod would now necessitate a complete rollback of the deployment - that's right, all of the good features have to be pulled off the shelves too and production is switched back to the last working version.

On the other hand, rigging each new feature or change with a feature flag enables teams to hit the kill switch as soon as an issue is found. Now, all of the features that work get to stay in prod, while the broken features are isolated out and handled as required.

Use Case 2: Incident Management

What happens when a severe issue occurs in production because of feature releases that are broken? Or when some infrastructure failure cascades across all of production? As it turns out, kill switches can be used to mitigate these incidents and improve the MTTR (mean time to resolution) for the teams responsible. Talk about stress management.

Scenario 1

In scenario 1, a broken new feature is causing an incident in production. The good news is, if that new feature was rigged up to a feature flag, the kill switch could be triggered and that feature could be turned off in production right away, resulting in minimal impact. Not only is the feature no longer affecting production, but there is also suddenly no massive response team that needs to be assembled to handle the issue, which would typically mean the whole deployment had to be pulled. Instead, the feature is turned off, the team responsible for the feature is on the hook to fix it, and the one feature fix can be rolled forward when it's ready.

Scenario 2

In scenario 2, an infrastructure failure is one example of what could happen in production. However, production outages can happen due to a variety of issues. Oftentimes, Ops teams will have fallback options or "just turn it all off" runbooks in place, and kill switches can be used here as well. If it's a new infrastructure change entirely, that whole change can be wrapped in a feature flag and a kill switch can be triggered that will failover to the old setup when an issue occurs. Some teams also create long-lived operational flags that can, say, put a whole site or application in "maintenance mode," effectively triggering a kill switch for the whole application. This can be useful in P0 scenarios where things are completely broken and the preference is to not allow any access.

Use Case 3: Application Load Management

Let's say your business is an e-commerce company. During the holidays, you experience high load relative to the rest of the year. Now more than ever, the business doesn't want to have downtime or have customers experience issues! Turns out feature flagging is an incredibly low-cost way to solve this problem.

One approach to solve this problem could be to simply turn off some features that will reduce the load, and this is a great place to use a kill switch, but it might not be the most effective. Another solution might be to kill certain load balancers or servers that are useful in non-peak times, and cut over to infrastructure that's designed specifically for high load.Â

Real World Example

At a previous job, we had log data that would need to go into storage from the short term cache, so we had a system to collect and store them. During peak loads or other periods of instability (particularly upstream with AWS), we would often see this storage become either very expensive or very unreliable. So, we would turn this service off during these periods, but this was manual and involved a series of AWS commands and database changes to both disable and re-enable later on. By wiring this up to a feature flag, we were able to turn this feature off and on instantly in a much more lightweight, visible, and easy to govern way.

Use Case 4: Testing in Production

One of the most common use cases for teams doing feature flagging is testing in production. In short, this is where a specific feature (or set of features) needs to be tested against real production data. After all, pre-prod environments can only test so many things! There are

two use cases that we see here:

The use of feature flags as kill switches here is a pretty simple one: have it live until the test has been run, and then turn it off. It's also possible that a feature will fail during testing and need to be turned off to minimize impact. You can dig deeper into testing in production in this dedicated blog.

Use Case 5: Progressive Delivery

Here, let's assume a basic knowledge of progressive delivery. It's becoming an increasingly popular feature release methodology, very similar to the methodology used for canary releases in Continuous Delivery.

With progressive delivery, the need for a kill switch changes slightly. Yes, the main use case is still turning things off when they break or aren't needed, but here, we have to layer in automation. By its nature, progressive delivery is something that teams run in an automated fashion. To do that, they need to pre-define in which scenarios flags are turned on, for whom, and under what conditions more users get access. Conversely, they must define failure criteria in which the kill switch is triggered and new features are turned off, or the user base is shrunk.

In Harness Feature Flags, we automate progressive delivery via the Pipeline. Teams can schedule releases, mandate approvals, integrate with plugins, create trigger events, and template rollouts - and then automate it.

Use Case 6: Personalization

Personalization is a hot topic. Kill switches using feature flags are of huge value in being able to do this well at scale. We can look at three distinct scenarios in which you can use kill switches to improve your ability to personalize experiences for customers.

Compliance & Regulation

When it comes to regulation, it's a non-negotiable item to be able to "personalize" something like a mobile app release to comply with the laws in various countries, or to meet the needs of clients in specific verticals (e.g. government, finance). The most common of these is data privacy - specifically, complying with GDPR.

Considering this, it can be a mess showing a button in one place but not another. Wiring key features up as permanent feature flags allows you to easily turn something on for all your North American users, but not your European users where GDPR compliance is mandatory. It's not just creating a kill switch that given context will turn off data collection in Europe, it's also being able to kill data collection for any user that decides to opt out. It's much easier than having a configuration file that tries to solve for all of the various scenarios.

User-Defined Personalization & Admin Settings

Think of this as giving control to the user on what features they want to kill. This isn't dissimilar to giving them access to admin settings on the app so they can choose to have dark mode or choose what notifications they get.

While on the frontend, users will define what they want and don't for their personal experience on the app, on the backend, it's just exposing parts of the feature flag schema to the end user and letting them make the decision on what to keep and what to kill.

Another consideration here is the use of features like screen time or parental controls, where a kill switch can be triggered to limit the access of users to specific applications altogether based on settings elsewhere on their devices. Of course, putting the onus on the user to define what they will do with such power is a whole other problem to solve.

Using Harness Feature Flags to Implement Kill Switches

Here's the plug: using Harness Feature Flags gives you the ability to implement all of these use cases right out of the box. When you set up feature flags in Harness, you're able to work either in a visual UI or entirely in code to manage your kill switches. You also give yourself peace of mind when you eventually want to scale your usage of kill switches or other feature flags use cases with built-in governance, compliance, and security considerations, as well as the critical integration into CI/CD (Continuous Integration and Continuous Delivery).Â

After all, you don't want to set up a great feature flagging system that is entirely disconnected from the rest of your software delivery process. It doubles your work on access control, security, governance, and integration maintenance. You end up getting diminishing returns on your feature flagging platform, which is supposed to *accelerate* software delivery.

If you want to see how Harness can fit into your organization, you can sign up for free forever, or contact us for a personalized product demo.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/harness-ci-and-harness-cd?utm_source=pmm-ff&utm_medium=using-feature-flags-for-effective-incident-management

Harness CI and Harness CD - Your First True CI/CD Pipeline

Harness now allows you to have both best of breed Continuous Integration and Continuous Delivery capabilities. Let's take a look at what leveraging a Harness CI / Drone Pipeline to build then leveraging Harness CD to deploy to a Kubernetes cluster.Â

The goal of this example will be to have Harness CI / Drone build and push a GoLang Docker Image to DockerHub then have Harness CD pickup on the new artifact and deploy to an awaiting Kubernetes cluster. This potentially can take you from idea to production easier than ever.Â

The Moving Pieces

There are not many moving pieces to see an end-to-end example. You need a Harness CI / Drone Instance, a Docker Hub Account, a Harness Account, and of course, a Kubernetes cluster which could even be Minikube.

Like always you can follow along with the blog and/or watch the video.

If leveraging the example Harness CI / Drone GitHub project, you will have to edit the information to be your own Docker Registry for where the Docker Push goes. I will be referencing mine below.Â

The Harness CD Piece

If this is your first time leveraging the Harness Platform, the quickest way to get started is to leverage a Kubernetes Delegate.Â

The first step of getting a Harness Delegate installed in your Kubernetes Cluster is pretty straight forward.Â

Setup -> Harness Delegates -> Download Delegates -> Kubernetes YAML

Give the Delegate a name.

Hit submit to download. Expand the tar.gz which is downloaded.

Inside the delegate folder, run `kubectl apply -f harness-delegate.yaml` to install the delegate.

In a few moments, your Harness Delegate will be available.

Next, add the Kubernetes cluster for Harness to deploy to by adding the Kubernetes cluster as a Cloud Provider.Â

Setup -> Cloud Providers + Add Cloud Provider

Add a Kubernetes Cluster

Give the Kubernetes cluster a name then for Cluster Details, Inherit from selected Delegate [deployed Delegate]. Hit Test then Submit and you are all wired up.

Next, creating a Harness Application is a simple process.Â

Setup -> + Add Application.

The next step inside the Application is to create a Harness Service. You can create a Service by going toÂ

Setup -> Captain Canary -> Services + Add Service. The Deployment Type will be Kubernetes.

Inside the Amazing App Service, + Add Artifact Source from a Docker Registry

Can wire to your Docker Image of choice.

Once you hit submit, the scaffolding will be there for the deployment and no need for additional configuration.

With the Service out of the way, we can wire a Harness Environment to deploy to.Â

Setup -> Captain Canary -> Environments + Add Environment

Once you hit Submit, next can wire an Infrastructure Definition.

Next click + Add Infrastructure and add the Kubernetes cluster.Â The Cloud Provider Type and Deployment Type will be Kubernetes.

Once you hit Submit, can wire together a Harness Workflow to define the steps to deploy.

Setup -> Captain Canary -> Workflows + Add Workflow. The Workflow Type will be Rolling Deployment. Select the Environment, Service, and Infrastructure Definition that was created in the previous steps.

Next you can create a Harness Trigger on the presence of a new artifact in DockerHub. This will trigger the Workflow once the Docker Push is complete in the Harness CI / Drone Pipeline.Â

Setup -> Captain Canary -> Triggers + Add Trigger

Hit Next after giving a Name. DockerHub can send out a webhook or we can use a polling interval from Harness to look out for a new artifact. Without the need to manage webhooks, we can just simply configure On New Artifact. Select your artifact source and can filter on specific tags. For the example we just want to deploy what gets picked up and â.*â is fine as a filter.

Hit Next and define the Actions. Execution Type will be Workflow and will execute the Workflow that was created in the previous steps. Can leverage the Last Collected Artifact. If leveraging the example will pick up on the newest explicit tag with the same filter â.*â.

Click Next and review the Trigger then hit Submit.

Once you hit Submit, all you have to do is kick off a Harness CI / Drone build and push.Â

Harness CI / Drone Steps

If you do not have a donât have a running Harness CI / Drone instance donât worry, not difficult to accomplish. We have a detailed blog and video to get you started. For the example to work will need to be building and pushing a Docker Artifact. The example project has a simple GoLang application. The Harness CI / Drone introduction blog and video go through wiring Harness CI / Drone to a GitHub repository.Â

To kick off a Harness CI / Drone Build and Push, simply modify the configuration / Drone.yaml which Harness CI / Drone is monitoring for. To explicitly build to a specific version, can add tags to Drone.yaml which works well with the example.Â

In the example, pushing build version 1.0.2. Make sure to update your repository information also.

Can modify/increment then commit.

Once you hit Commit, Harness CI / Drone will take over.

Once the publish step is finished, a new image will be in DockerHub.

Watch the Magic

Within the polling interval, Harness will pick up on the newest build/tag and start deploying on your behalf.

With the Deployment kicked off, you have an end-to-end CI/CD pipeline. From code to production!

Partner with Harness in your CI/CD Journey

No matter where you are in your CI/CD journey, Harness can simplify and scale your capabilities. With the lightweight and convention-based Harness CI / Drone and the power of the Harness Platform, achieving CI/CD nirvana has never been easier. Get started with Harness CI / Drone and sign up for a Harness Account today!

Cheers,

-Ravi

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/chaos-engineering-benefits>

Benefits of Chaos Engineering & More

Chaos Engineering first emerged in 2010 by Netflix to test system stability, and it has become the next big thing for testing in DevOps. It began with the launch of various chaos testing tools, such as Chaos Monkey from Netflix, LitmusChaos, Chaos Gorilla, Gremlin, Chaos Mesh, and so on. It's being adopted by numerous organizations and across engineering teams, including developers, quality assurance, and site reliability.

As defined by Google, Chaos Engineering is simply the process of testing a system's capability to ensure that it can withstand turbulent conditions. It relies on principles of Chaos Engineering, focusing on random and unpredictable behavior to build more resilient systems.

Chaos Engineering emerged as a preventive measure to identify the failure scenarios in systems before they develop and cause downtime. You can identify and fix issues immediately by doing pragmatic chaos testing on how a system responds under injected breakage. This helps safeguard end users from any negative impacts.

Read on to learn how Chaos Engineering improves and benefits the resiliency of large-scale distributed systems and microservices.

Why Do We Practice Chaos Engineering?Â

Over the past few years, leading organizations in the industry have faced costly network outages. For example, the 2018 Amazon blackout incurred a loss of up to \$99 Million, and the 2019 Meta Platforms Inc. (Facebook, Messenger, WhatsApp, and Instagram) downtime cost over \$89.6 Million.Â

These mishaps are ongoing problems that every enterprise has faced since the internet began. Recently, the Information Technology Intelligence Testing (ITIC) 2021 Global Server Hardware, Server OS Reliability Survey indicated that the COVID-19 pandemic, security hacks, and remote working are the driving factors of these blackouts. All of these factors culminate in massive revenue loss and maintenance costs.

But money is not the only thing that an entity loses while facing outages. Outages also create a domino effect, and along with the money, there is the loss of customer and employee confidence, brand integrity, stock prices, and more. In some cases, companies are even subject to legal action by their stakeholders.Â

Adoption of Chaos Engineering

With such vast risks stemming from a single problem, a long-term solution for resilient systems was long overdue. Chaos Engineering now allows companies to:

- Identify the weak points in their systems
- See how the systems respond to coercion as real-world events in real-timeÂ
- Prepare and educate the engineering team for actual failures
- Create a cost-efficient failover plan that saves money in the event of a failure
- Validate the resiliency of their business data in the event of a catastrophic failure.

For enterprises, these Chaos Engineering examples are essential in addressing the identified weaknesses before they cause data loss or service impacts.

Let's dive into some of the significant benefits of Chaos Engineering.

Chaos Engineering Benefits

Testing the limitations of your applications and distributed systems can provide a vast range of information for the development teams and organizations. Here are a handful of the benefits of Chaos Engineering in practice with chaos testing tools.

1. Increases Reliability and Resiliency

Using a Chaos Engineering tool to conduct planned chaos experiments will help test the system's capability and thus increase its resilience.Â

While conducting these chaos experiments, you must choose your metrics wisely and hypothesize the steady state. The initial Chaos Engineering experiments can take place in staging or any pre-production stage where the blast radius is minimal. This makes sure that if any negative impact occurs, users aren't affected.Â

2. Increases End User and Stakeholder Satisfaction

Once you and your team gain confidence to perform Chaos Engineering experiments, you can run experiments close to production. All experiments are conducted directly using the actual input received in the production environment in the ideal implementation.Â

As the production environment is treated as the real system, advancing your experiments here gives you a precise idea of what your end users would experience. The system will experience fewer network failures and service disruption, which will in turn improve the user experience.

3. Advances Team Collaboration and Confidence

The insights that chaos engineers gain from these chaotic experiments improve the engineering team's knowledge. This results in faster response times and improved collaboration and confidence. Furthermore, these insights can be used to educate newer colleagues.

4. Improves Incident Response Time

With the technical team having been made aware and brought up to speed from the previous chaos experimentations, troubleshooting, repairs, and incident responses can increase in velocity. Therefore, insights received after running chaos testing can lead to a reduction in future production incidents.Â

Gamedays are one way to improve incident response time. The idea is to provide room in your workflow for the team to practice what they'll do if something goes wrong with the production environment.

5. Improves Application Performance Monitoring

Chaos testing is considered one of the most holistic approaches to performance engineering and testing methodologies. Regularly conducting chaos experiments develops confidence in the distributed systems, and it helps to make sure that applications perform well

despite major unexpected failures.

Conclusion

In the current DevOps industry and Software Development Life Cycle, Chaos Engineering has evolved into a fantastic tool that may assist organizations not only in improving entire system resources, resiliency, flexibility, and velocity, but also in operating distributed systems.

Along with these advantages, it allows us to address problems before they can negatively impact the overall physical cloud infrastructure. Chaos Engineering implementation is critical and should be adopted for better results.

Leading companies like Microsoft, Amazon, LinkedIn, and many more have implemented Chaos Engineering in their tech stack. Chaos Engineering should be added as a performance-achieving metric for strengthening the resiliency of any organization that constructs and manages a distributed system and aspires to achieve a high rate of development velocity.

The marketplace for Chaos Engineers has opened many arenas and opportunities. Chaos Engineering is still a young field, with new techniques and Chaos Engineering tools emerging all the time. As a true supporter of this resilient technology, I have shared this article about integrating Chaos Engineering and its benefits as experienced by its customers.

Please feel free to look at the Harness Chaos Engineering tool, or the full Harness Software Delivery Platform to see how we can help you take software delivery to the next level.

Join the LitmusChaos Community

LitmusChaos is an amazing Chaos Engineering tool for conducting chaos testing on your systems. With 2.7k+ GitHub Stars and around 1300 Slack Community Members, we're a lively community for young learners. Check out our Chaos Hub, which is an open-source marketplace containing all of the different chaos experiments offered by LitmusChaos, pod-delete being our most popular chaos experiment.

With LitmusChaos, you can start your journey toward becoming a Chaos Engineer. Want to get help with queries, learnings, and contributions? Join the LitmusChaos Chaos Engineering Slack community. To join, just follow these steps!

Step 1: Join the Kubernetes Slack using the following link: <https://slack.k8s.io/>

Step 2: Join the #litmus channel on the Kubernetes Slack, or use this link after joining the Kubernetes Slack: <https://slack.litmuschaos.io/>

Looking forward to seeing all of the amazing folks from the open source world! Here are some important links for you to reference:

LitmusChaos Twitter

LitmusChaos Website

LitmusChaos GitHub Repo

LitmusChaos Docs

LitmusChaos YouTube Channel

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/verifying-your-kubernetes-deployments-made-easy>

Verifying Your Kubernetes Deployments Made Easy

Software organizations today are heavily focusing on automation and developer efficiency. No doubt, the DevOps approach gives wings to the software development and delivery teams, but security, logging, and monitoring have become way more important these days. After deploying your application and services to production, it is highly recommended that you monitor them. If any anomalies are detected, correct them then and there before they affect your services and customers. But how can you continuously monitor your services and applications? Need more experienced professionals who understand logging and monitoring; third-party tools are expensive and time-consuming to set up.

Keeping these pain points in mind, Harness (the modern continuous delivery platform) has introduced a feature called **continuous verification** to help DevOps professionals to verify their deployments through any monitoring tool of their choice.Â

Today we will talk about continuous verification and show you how to verify your Kubernetes deployments using the Harness continuous verification feature.

What is Continuous Verification?

Continuous Verification (CV) is a practice that involves continuously monitoring and validating the quality of software deployments, making sure that the deployed applications and services are serving as expected. It is a process that ensures that changes made to a software system are deployed successfully and do not negatively impact the system's performance or functionality. In the context of deployments in software, Continuous Verification involves using a monitoring tool such as Prometheus, AppDynamics, NewRelic, Splunk, DataDog, Dynatrace, Cloud Watch, ElasticSearch etc, to validate and confirm that the newly deployed application or service is working correctly. This service/app is always monitored to make sure it always works as expected.

This includes monitoring application performance, and logs, checking for errors and bugs, and ensuring that the application meets all functional requirements. Continuous Verification is a crucial part of the Continuous Delivery (CD) process, which aims to enable software development teams to deliver new software versions rapidly, frequently, and with confidence. By continuously verifying the quality of deployments, development teams can catch issues early, fix them quickly, and ensure that the application is always running smoothly. In case any anomalies are detected, they should be reported and fixed quickly to avoid service downtimes.

Harness Continuous Verification

Harness Continuous Verification is a powerful tool that can help you ensure the quality and performance of your deployments. With Harness, you can easily set up a pipeline to verify your deployments, connecting a variety of monitoring tools of your choice. Once you've set up your verification step in the pipeline, Harness uses unsupervised machine learning to detect anomalies in the deployed applications or services. You can set a threshold for these anomalies, and when they cross the set threshold, the organizations will be able to auto roll back and de-risk their deployments.

The teams get alerts if any issues are detected, and they can view detailed reports on the status of the deployments and take action to fix any issues that arise. Overall, Harness Continuous Verification is an essential feature for any organization that wants to ensure the reliability and quality of their deployments. Harness can do production and post-production deployment verifications to help you make sure that the deployments are taken care of and monitored properly.

To verify deployments in Harness CD with Prometheus, you can follow these simple steps:

Configure Prometheus in Harness Through Connector: Provide the necessary connection details, such as the Prometheus server URL and authentication credentials if required. This step allows Harness to access Prometheus for data retrieval.

Define Verification Criteria with queries: Specify the metrics you want to verify and define the criteria for success or failure. For example, you might verify that a specific metric value remains within an acceptable range or meets a certain threshold. You can configure assertions based on query expressions, comparing values, or checking for specific patterns in the returned metrics.

By following these steps, you can leverage Harness CD's integration with Prometheus to verify your deployments based on the metrics collected by Prometheus.

Let us see how to set up continuous verification functionality practically to verify deployments using Harness CD.

Prerequisites

- Harness account with CD free plan
- Kubernetes cluster access to deploy our sample application
- A Prometheus Endpoint, we will show how to get it in our tutorial

In this tutorial, we will deploy an application that writes to a Prometheus endpoint and is validated by Harness Continuous Verification.

If you do not have access to Prometheus, you can install the Prometheus on your Kubernetes cluster.

â

Install Prometheus with Helm.

The next step would be to expose Prometheus via LoadBalancer.

You can easily get the Prometheus endpoints when you expose the service with the LoaddBalancer type, and you can see the endpoints in your dashboard. [I am using GCP to create a cluster]

Tutorial

In this tutorial, you will see how the verification step works with Rolling as well as the Canary deployment strategy.

Sign up for the Harness CD module and get started with creating a pipeline.

Before continuous verification, you should know how to deploy a Kubernetes manifest. Follow this guide and create a simple CD pipeline to deploy a Guestbook application.

Create a project and a CD pipeline, as stated in this tutorial.

After configuring the CD pipeline, you can save and run the pipeline to see a successful deployment.

We successfully deployed our application; Let's add a verification step to verify the deployment.

Adding Continuous Verification Step

After the successful deployment, edit the pipeline and add a continuous verification step from the step library.

But wait, to be more successful, let's change the deployment type from ***Rolling to Canary***, as it is considered more effective while releasing new features or a piece of software.

Why Canary Deployment?

Canary deployment is a deployment strategy used in continuous delivery (CD) to mitigate risks and ensure the stability of new releases. It involves gradually rolling out a new version of an application to a subset of users or servers while still routing the majority of traffic to the stable, existing version.

Based on the analysis and verification results, Harness CD can decide whether to proceed with a gradual rollout, increasing the canary deployment's scope, or roll back the deployment entirely if issues are detected. Prometheus metrics play a crucial role in this decision-making process.

By using the canary deployment with Prometheus monitoring in Harness CD, you gain the following benefits:

- **Risk Mitigation:** Canary deployments allow you to gradually test new versions in a controlled manner, reducing the impact of issues or bugs on your entire user base.
- **Performance Validation:** Prometheus metrics enable you to evaluate the performance of the new version compared to the stable version, ensuring it meets the desired criteria.
- **Automated Decision-making:** Integrating Prometheus with Harness CD allows you to automate the analysis and verification process, making deployment decisions based on predefined rules and thresholds.

Even in the âVerifyâ step, change the âContinuous Verification Typeâ to âCanaryâ.

You can add the following details to the verification step.

Continuous Verification Type: Canary

Sensitivity: HIGH

Duration: 5 Min

Artifact Tag:

You can click on **Add** to add a Health Source. You will be presented with a variety of health and monitoring tools to connect with. Choose the one that is feasible for you. In this tutorial, we will be selecting Prometheus from the list.

Select Prometheus and create a new Prometheus connector. Harness uses Connectors to authenticate and perform operations with a 3rd party tool. Harness uses a Delegate to test the Connector by establishing network connectivity and authentication.

As you can see above, **promo-connector** is my Prometheus connector (you can name whatever you wish). Let's see how to create this connector and connect with Harness Delegate.

Share your Prometheus endpoint URL.

Connect with the available Delegate.

Make sure the connection to Delegate is successful.

We have successfully added the Prometheus connector.Â

Click next, and you will land on this query configuration page.

Add all the required details and build your query. You need to create the **Group Name** and you can edit the query tab and add the below query string.

Submit, save the pipeline and run the pipeline.

If there are any anomalies detected, you can find them in this console view.

It takes some time to validate the logs, and finally, you can see the successful pipeline with the continuous verification step.Â

Expand the metric to see the details.

Also, you can easily check the external API calls and execution logs to see the real-time events.Â

Similarly, you can add other monitoring tools, such as Splunk, NewRelic, DataDog, AppDynamics, etc.

NewRelic Monitoring: Verification Step Failure Scenario

We even took an example of setting up a NewRelic monitoring source with a sample Node.js application.Â

We followed everything as usual, as we did in the previous tutorial and added **NewRelic** as the health source in the verification step. The application is getting deployed with the Canary deployment strategy. Would love to show you what happens when the verification step fails. Let's get started.Â

â

â

This is what our pipeline looks like.

Purposely we are making the verification step fail here to help you understand how the pipeline works in case of verification failure.

What next? You need to perform an action with the available actions presented. You can rollback, ignore, retry, mark it as a success, abort, proceed with default values or mark it as a failure.Â

It depends on your team what action you like to perform here in case of verification failure.Â

In this case, as we said above, you can manually intervene and pick any action that is suitable. Here I picked **StageRollback**, and you can see the pipeline execution below.Â

The Harness continuous verification works really well when you have complex deployments such as multi-service deployments. There are many possibilities with Harness continuous verification to verify your deployments by connecting different logging and monitoring tools. While many organizations don't take post-deployments and monitoring seriously, this gives you an edge over others to deploy your applications and services confidently.

Try Harness CD Today!

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/feature-flags-pipelines?utm_source=pmm-ff&utm_medium=how-we-use-our-own-feature-flags-at-harness

Announcing: Feature Release Workflows With Feature Flags Pipelines

We're thrilled to announce the introduction of feature release workflows for Harness Feature Flags, called Pipelines. Users of Feature Flags can now standardize the process and steps required to release a feature, and layer in automation to make feature release as simple as hitting 'Go.'

Feature Release Management Is A Chore - Schedules, Approvals, Monitoring - Oh My!

The most common implementation of a feature flags solution is one that is built in-house. In fact, most of the time people don't even realize they're building a feature flags solution! And that makes sense. There are a number of point problems that can be solved with feature flags, and engineering teams want to solve those. But before they know it, it's turned into another internal tool to maintain and upgrade.

What makes this difficult is two-sided:

Process is important, especially as organizations seek to maximize the value of the feature flag solution that they have built or bought. Even for those who have bought a tool, it's critical that they are able to create a standardized release process that anyone can follow, and that guarantees that any feature release has gone through the appropriate steps. And let's not forget about being able to audit releases for compliance purposes - or because something went wrong and a root cause needs to be identified.

When we look at all of these things together, it becomes obvious that all of the requirements are interrelated. Let's be clear about the problems that need to be addressed. And who knows, maybe there's a way to solve them all at once!

- Integration into the release process: monitoring, approvals, scheduling, etc.
- Standardization of feature releases across teams, products, projects, etc.
- Ability to view audit trails for compliance and root cause analysis.
- Creating a common framework that anyone can work off of.

You can imagine that being able to do this would be a massive burden taken off the shoulders of anyone wanting to use a feature flag solution, whether they're a developer or a manager. If you were able to solve these problems, you'd essentially make feature releases non-eventful.

And maybe you could even automate it?

Enter Pipelines: Feature Release Management Workflows

To solve these problems, we built Pipelines for Harness Feature Flags. Pipelines take all of the requirements and problems to be addressed above and put them into a single construct that is significantly easier to comprehend and use than the traditional hodge-podge of tools, logs, scripts, and emails.

Building Workflows for Common Release Requirements

Check out this simple pipeline below. Even though there are only three steps, it clearly outlines what you want to happen: turn on the feature; get an approval; and then release to the rest of the beta segment of your customers.Â

If this were your actual release process, you could easily take this pipeline and turn it into a reusable template. Now, every time you have a new feature, you could just click a button and feel good that all steps will be taken care of. Of course, these can get more complex as well. Letâs take a look at what that could be:

Part of what makes Pipelines in Harness Feature Flags really useful is the deep library of integrations into tools you and your team probably use. Now we can expand the simple use case above where only an approval is required. The workflow could include updating Jira tickets, sending Slack updates, verifying logs or health metricsâ| Basically, anything you might need in order to release a feature.

Now youâve taken all these disparate activities and rolled them into a single, manageable workflow. If something goes wrong, you can easily access the audit logs to see what changed, who changed it, when, and so on. You also have a visual pipeline, should you choose to use it, that you can use to quickly spotcheck the progress of a feature rollout or verify that the right steps are included.

Automation of Feature Releases

Itâs not just about eliminating the toil and headache associated with feature releases by creating templated workflows. You might have picked up on the nature of Pipelines, which is that it can be automated.

Once a Pipeline is created to meet the needs of your workflow, you can associate any feature and simply hit the proverbial âGoâ button. From there, the Harness Pipeline will take care of going through each of the steps that youâve laid out. It can automatically handle rolling out the feature, pinging people for approvals, making changes in Jira, checking health metrics - you name it. Of course, if there are issues that require attention or failures in the process, you will be alerted. This leaves you to do things other than babysit the release pipeline, and you only need to be involved if something goes wrong.

In the near future, weâre also looking to integrate automated health metric verification that will take this a step further. Imagine you want to roll out a feature to a subset of your users, verify that things work as expected, or that the right business metrics are moving, and then progressively roll that out to larger portions of your user base. Or, if something goes wrong, you want to take some pre-specified remediation action. All of these things will be possible, meaning that you will truly be able to simply declare what you want, and the system will take care of making it happen.

Added Value of the Harness Platform

As an end-to-end software delivery platform, it wouldnât make much sense *not* to have Harness Feature Flags have some interplay with the rest of the software delivery process. We commonly see that feature flags are an amplification method for CI/CD and in many ways are the third step of the CI/CD pipeline.

Harness is actually the only feature flag solution on the market today that natively integrates with CI/CD. Why does that matter? And what value does that provide to a product development or ops organization? And why do executives care about it too?

Letâs start with a topic weâve already covered - automation. The same Pipeline feature in Harness Feature Flags is also available in Harness Continuous Integration and Harness Continuous Delivery. Yes - you get a unified pipeline. Suddenly, you go from disconnected CI/CD and feature flagging to a single pipeline that shares context and can be automated from build through to individual feature delivery to end users. Can you imagine what that could mean for your software delivery process? But thatâs not the only reason it matters.

Analytics, governance, compliance, risk - all are other really important reasons to have CI/CD and feature flags linked up. By having shared context across the entire SDLC, youâre able to do all of the above better:

- Create a canonical analytics dashboard to track software delivery metrics at each stage, and the interplay between them.
- Enforce org-wide governance policies for software delivery, and ensure that governance standards are always met.
- Leverage audit trails across every stage of software delivery to see exactly whatâs happening and where, and simplify external and internal audits.
- De-risk release at any stage because youâre not worried about inter-tool integrations or support and critical context being lost that can impact the business or customers.

Of course, you can go into more detail about any of these topics on our blog, which speaks at length about each of these topics.

How to Get Started

If youâre already a Harness Feature Flags user, you can head over to the product and youâll see a new menu item labeled âPipelinesâ that you can click and get started building your own!

If you havenât signed up to use Harness yet but want to get started, you can easily sign up for a free trial. Happy developing!

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/general-availability-harness-developer-hub-hdh>

Announcing the General Availability of Harness Developer Hub

We are excited to announce the general availability (GA) of Harness Developer Hub (HDH), a modern developer learning platform that brings together Tutorials, Documentation (including Release Notes), and Certifications for all Harness modules and the underlying Harness Platform. This announcement builds on top of the Beta release a few months back.Â

HDH has three sections geared towards helping developers become experts in software delivery: Tutorials, Documentation, and Certifications.

Get Started with Tutorials

The Beta release was primarily focused on Tutorials and since then we have expanded on both the breadth and depth of the tutorials. Every tutorial is designed with a specific outcome in the end and is powered by a hands-on code/config example that is either available inline or stored in a public GitHub repository. These tutorials are organized under the following sections:

- Build & Test Code
- Deploy Services
- Manage Feature Flags
- Manage Cloud Costs
- Manage Service Reliability
- Orchestrate Security Tests
- Run Chaos Experiments
- Administer Harness PlatformÂ

The current list of tutorials is just the beginning. We encourage the broader Harness Community to send in requests for new tutorials through either Pull Request-based contributions or GitHub issues on the public HDH GitHub repository. Increasing our ability to accept such contributions is one of the key reasons behind choosing a documentation-as-code approach for HDH.

Learn More with Documentation

Harness Docs have resided on docs.harness.io since the beginning. As part of the HDH GA, we have migrated all content from docs.harness.io to HDH. Additionally, we have ensured that the previous docs.harness.io based URLs are automatically redirected to their corresponding location on HDH. The Documentation section in HDH is divided into three sections: Module Feature & Reference Docs, Platform Reference & Feature Docs and Release Notes.

Module Feature & Reference Docs

As listed below, each of the Harness modules have their own dedicated section for feature and reference docs:

- Continuous Integration
- Continuous Delivery & GitOps
- Feature Flags
- Cloud Cost Management
- Service Reliability Management
- Security Testing Orchestration
- Chaos Engineering
- Harness FirstGen

Platform Feature & Reference Docs

Administrative tasks that you perform on the common Harness Platform are documented in their dedicated section. The following are some of the key topics included in the Harness Platform docs:

- Pipelines
- Templates
- Delegates
- Secrets Management
- Organizations & Projects
- User Management
- Roles-Based Access Control
- Dashboards & Reports
- Policy as Code
- Terraform Provider
- REST API and API Reference
- Self-Managed Enterprise Edition

Release Notes

Overall release notes in the form of What's New and Early Access are available in the Documentation section. You can also find module-specific and Harness Platform release notes in this section. And, we are pleased to note that you can now subscribe via RSS to Harness Release Notes.Â

Become a Harness-Certified Expert

After you have leveraged the Tutorials to get started and the Documentation to gain deeper expertise, it's time for you to formally become a Harness Certified Expert. You can get started with the Software Delivery Foundations certification today.Â We will be adding multiple module-specific certifications in the near future. Not only will you be able to learn and strengthen your software delivery skills, but you'll also be able to certify your skill sets in the Hub and build on your Harness knowledge.Â

Search & Learn

If you simply want to learn about a specific topic, then we have a new search experience for you. The search results can be filtered by not only Content Type (such as NextGen Docs, FirstGen Docs, Tutorials and Release Notes) but also by Module name. This allows you to reach the content you are looking for with ease.

The Road Ahead

We are excited for this initial launch of the HDH aimed to structure and accelerate the developer learning journey in the context of software delivery. The road ahead involves adding more hands-on tutorials. We will also increase the depth of the feature and reference docs while ensuring that the tutorials and docs content serve as the foundation for you to prepare for and ace your certifications. If you have any feedback for us, please do not hesitate to fill the Feedback form on the bottom right of the HDH home page. Looking forward to hearing from you!

Similar Blogs

[ArgoCD, Terraform and Harness](#)

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/flag-pipelines-to-automate-feature-flag-governance-and-daily-management?utm_source=pmm-ff&utm_medium=release-management-with-feature-flag?utm_source=pmm-ff&utm_medium=how-we-use-our-own-feature-flags-at-harness

Announcing Flag Pipelines to Automate Feature Flag Governance and Daily Management

When we launched Harness Feature Flags, we were excited about the opportunity to leverage other capabilities across the Harness platform to create better-than-the-sum-of-its-parts value. One of the first things we did was add support for feature release pipelines to Feature Flags, which enabled users to automate software release workflows and policy enforcement.

Today, we're extending how you can use pipelines with the launch of a new concept, Flag Pipelines. Pipelines have always let you build powerful workflows and automation for doing releases with feature flags, and now development teams can run pipelines automatically on every flag change to enforce governance and control. That's right — flags can have individual pipelines that run every time a flag state is changed. That's a game-changer for automating the day-to-day usage of flags and making sure that every change meets organizational standards.

Essentially, you can now guarantee that any time a flag's state is changed, it goes through a predefined workflow. This can be something as simple as requiring an approval every time a sensitive feature is toggled. Or, it can be something more complex, such as requiring an approval, making an update to Jira and ServiceNow, and sending notifications to customer support and managers.

Governance in the Day-To-Day Flag Lifecycle

Flag pipelines let you build a simple pipeline and assign it to a flag. From that point on, that pipeline will run automatically every time the flag changes, no matter how it changes. This means that whether you change it via the UI, API, or by updating the YAML via Git, you'll see this pipeline triggered.

This is great addition to feature management when your team wants to do things like:

- Require an approval from a manager role, or a PM, on any flag change in the production environment
- Send a Slack message on every flag change
- Update a Jira ticket being used to track feature development steps
- Execute a custom script associated with the flag every time the flag changes

Manual Process Misses the Point Of Feature Flags

Normally, you'd use pipelines with feature flags to build automated release workflows, such as incrementing a flag to 10% of your audience with an approval every 23 hours. And you can still do that for your one-off or specifically-modeled release scenarios. Flag Pipelines let you automate and enforce the things that you want to have happen *every time*, not just when you have a specific one-time scenario.

For teams and broader organizations, this feature enhancement provides a broader level of control over how feature flags are handled and managed day-to-day. Previously, users would have had to create an organizational process that required all of the users to run a stated pipeline every time that a flag needed to be changed. This poses a few problems:

It's not that teams don't try to do these things the right way, rather that things slip, people and processes change, and it's inconvenient to follow numerous extra steps for something that should be simple. Instead, what if users could use flags in a way that's more closely aligned with their actual needs, and the release process for flag changes is automatic? By flipping the script, devs aren't slowed down, and the business makes sure that things are done the right way every time.

How to Use Feature Flag Pipelines

â

To set up your flag pipelines, just select any flag and select the new "Flag Pipelines" tab. From there, you can choose or create the pipeline that you want to use.

Because flag pipelines are meant for the subset of workflows that you want to execute every time, not all pipelines can be used as flag pipelines. Flag pipelines must:

- Not have continuous integration/continuous development (CI/CD) steps
- Only have one flag change, not multiple
- Have the flag step set to runtime inputs

Once you have a suitable pipeline and have it selected inside of your Flag Pipelines settings within a flag, you're all set. That pipeline will now automatically run every time that the flag is changed within that environment.

Find more details about how it works in the Harness docs.

Get Started with Harness Feature Flags

Flag Pipelines continue our work in the pursuit of turning feature flags into a true part of your software delivery process. Allowing more sophisticated control of feature flags brings increased governance and feature flag automation to your organization. It's a big step forward in evolving feature flags into a key business process in software delivery.

Ready to get started? Request a demo or sign up for free forever. We'll continue to expand on this in the future and would love your feedback on how you'd like to see this story evolve.

Happy developing!

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?
Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/combining-cicd-and-feature-flags?utm_source=pmm-ff&utm_medium=the-journey-to-using-feature-flags-at-their-full-potential

Control and Velocity in Production: Combining CI/CD and Feature Flags

Needing to deliver software faster and more safely is a foregone conclusion. Engineering teams are constantly trying to deliver software faster than before, but without introducing more risk to the system.

As part of this evolution, software delivery pipelines were born, most notably CI/CD. However, as teams push the boundaries of CI/CD, they find that they're running into new problems at the tail end of their software development life cycle.

In this blog post, we'll explore this tail end of the problem: deployment. And you can expect to learn about where feature flagging ties into this.

Why Software Delivery Pipelines?

Whether you call it a pipeline, process, lifecycle, or any other term, the idea is the same - there's a steady stream of changes and new feature releases being developed, and they need to make it into the hands of customers.

Especially as you scale, you *need* a standard way of delivering software: the pipeline. This ensures that what you ship consistently passes muster, and also minimizes your risk (and of course your potential for rework and fire drills).

Delivering software more quickly and safely is why teams adopt CI/CD. And now, teams are also adopting feature flags to extend that software delivery pipeline even further. The same way in which the end result of CI, the artifact, is consumed by the CD pipeline, the end result of CD, the deployment, is consumed by a feature flag system to take that next step.

But why do teams need to adopt feature flags? Aren't we good to go after code has been deployed to production? Aha! Therein lies the problem: what happens once we're *in* production.

The Limitations of CD

Let's start with the obvious statement: CD ends when code is deployed to production. What control you have once the deployment is live is restricted to rolling back the entire deployment. You have no control over the contents of the deployment, and you can't solve for single points of failure in production without pulling the entire deployment out of production. Let that sink in for a minute: you lose control the moment you push to production. You might feel a pit in your stomach just at the thought of doing something that ridiculous.

The reality is, however, that many teams are living this every day. They need the speed and control before things are pushed to production, but they have limited control of what happens once it's sent over the proverbial wall. This can create some interesting problems before and after deployment.

Before Deployment

- Deployments get larger, compounding risk and causing the deployment process to take longer and add more stress.
- Deployments are bottlenecked by individual changes that are not ready.
- Have to build app configs, DB flags, or environment variables to enable release control to specific user sets.
- Multiple versions or long-lived source control branches for distinctly different feature set requirements by customers.

The kicker here is that even after all of this is done, deployments can still be incredibly stressful, and hitting a Go is as dramatic as hitting the big red button in the movies. The biggest thing here is that if something goes wrong, it's all hands on deck. Not only is this stressful for DevOps, it also slows down the velocity of new features because development teams have to do rework.

Beyond that, creating multiple experiences based on the customer profile (e.g. free vs. paid user) can be burdensome and doesn't always work as expected. But the thing is, that control is something that's necessary after deployment - not just for engineering teams, but also for product, sales, and customer success teams.

After Deployment

- If 50 changes are deployed and only one has an issue, the whole deployment needs to be rolled back.
- Clunky to control which customers get access to specified features, and engineering usually has to do it manually.
- Hard to use data to understand how a specific change or new feature is performing compared to expectations.
- Real-world data often causes some sort of unexpected behavior and requires rework, or in the worst case scenario, a full deployment rollback.

Despite our best intentions, we often find that we end up with a real mess on our hands in production. The clunky customer experience controls often break, long-lived feature branches cause weird merge issues, and if something is just brokenâ€¦ wellâ€¦ say hello to a long list of stressed and irate engineers. Despite knowing what needs to happen in production, itâ€s incredibly difficult to get that control and achieve the right objectives without a proper system in production that establishes control and removes all of these barriers.

The Role of Feature Flags

There will be no surprises when I say that a feature flag management system is what needs to be in place to solve the limitations of CD. And doesn't it make sense? Teams are already trying to implement feature flags with the tooling that they have in place.

Not having control after a deployment is scary - and risky. It's not just developer stress levels that go through the roof, there are also tangible business impacts to not having control in production - especially when something goes wrong! And that's the first role of feature flags.

Control in Production

If you have a feature flagging system in place, you realize a lot of value immediately. Knowing that you don't lose control of your new changes and features once they're deployed is huge for development and DevOps teams. Allow me to list out of the benefits of using feature flags to get control in production:

- **You have a first line of defense.** If something goes wrong, it can be handled at the feature level, in production. No more rollbacks of monolithic deployments because of one bad feature. Think of it like creating an easy-access kill switch per feature. We go more in-depth on the topic in this blog.
- **Simplify feature access control for customers.** Instead of a runtime config, you're able to granularly define which set of users get access to individual features. An immediate use case here is the delineation of beta vs. GA or paid vs. free users.
- **Decouple deployment from release.** Deploy doesn't have to mean that all users get access immediately to everything in the deployment. But with only CD, that tends to be the case. Feature flags allow you to deploy everything in an OFF state, and then you can choose when the feature is live to customers - and who gets to access it.
- **Empower non-technical teams and take the load off of engineering.** In a world without feature flags, engineers unfortunately become the bottleneck to customer needs, having to manually turn on and off features and permissions. Once in production, feature flags let you hand over control to customer-facing teams so they can manage the customer experience, and you can let engineers get back to their main job.
- **Test code in production.** You can only test so much in pre-prod environments. Once real data starts flowing in, that's when you know how good or bad the solution really is. Feature flags essentially allow for safe testing in production by rolling out multiple solution implementations (or even just one) to specified user cohorts and see what best solves the customer problem, or if the feature is ready for primetime. We go more in-depth on the topic in this blog.

Velocity in Software Delivery

The other side of the coin is how feature flags let you realize higher velocity in your software delivery process. Recall, first, that CD does increase velocity, but there tend to be diminishing returns as deployments get larger because of 1) more risk, and 2) more features/changes that need to be completed. So how do feature flags benefit this?

- **Deploy without release.** Push code into production that's not ready by putting it behind a flag, and test features with small user cohorts before you roll out to all users. Now, deployments can go live as planned and on time as long as things are put behind a flag.
- **Deploy in smaller chunks.** This at first seems counterintuitive, but if you think about it, by deploying more often and in smaller chunks, don't you deliver faster and with less risk concentration? Feature flags allow you to ship things as they're ready, which ultimately lets engineers move at their own pace - not the pace of the slowest or biggest release.
- **Relieve the engineering bottleneck.** Feature flags free up the engineering bandwidth that was previously spent making manual changes for customers. It's a double whammy for customers, who get requests resolved much faster, and get new features sooner.
- **Feedback, feedback, feedback.** How do you know if you've built the best solution for your users? The answer is always feedback - but how do you get more feedback and faster? Can you turn a 3-month process into a 1-month process? Using feature flags, this is what teams are able to start doing better. It may seem like adding more feedback points would slow down the process, but in practice it turns out it's faster!

It Doesn't Make Sense to Separate CD and Feature Flags

Feature flags are to CD what CD is to CI. It's the natural follow-on that makes it not CI/CD but CI/CD/FF. And the same logic applies too - you need CD to do CI properly, and you need FF to do CD properly. In fact, let me ask it another way: why *wouldn't* you do CD and FF together?

Like Batman and Robin, it's the combination of the two that truly does justice to the goal of velocity and control in software delivery. CD is about shipping as fast as possible, but that velocity is rate-limited by the reliability and quality of what you ship. Feature flags

remove this ceiling by consistently making shipping safe, freeing up engineers to focus on what matters most - not on the stress of software delivery.

In fact, teams that use feature flags regularly outperform teams using only CD on the DORA metrics:

- 80% increase in deployment frequency
- 66% reduction in time to go from commit to deploy
- 90% of incidents resolved within one day

Teams want real velocity and real control, not to get stuck part way there. They want to minimize risk and stress, and maximize the quality and quantity of new features. While feature flags or CD on their own bring teams closer to that vision, it's not until Batman and Robin work together that this future is truly realized.

True Continuous Delivery With Harness

The thing is, most feature flag implementations result in a disconnect with CD, which introduces unnecessary risk and leaves management and visibility fragmented - not what you want for mission-critical systems. You miss out on metrics, audit logs, governance, and security.

That's where Harness comes in. Harness CD and Harness Feature Flags together provide the common enterprise thread required to perfect Continuous Delivery in all its forms. Of course, you don't need to have both - but then again, having Batman and Robin together only makes perfect sense.

Final Thoughts

At this point, you should feel like you have a fairly good understanding of the high-level implications of keeping CD and feature flags separate, and really why it doesn't make sense to do that.

To be clear: you're missing out on the full value of both CD and feature flags if you're not thinking about using them together. And in fact, you're probably creating more headache for yourself using just one and not the other.

Harness meets you at each stage of your software delivery life cycle, whether that's CD, feature flags, or something else. If you're ready to check out how we do what we do, you can sign up for free forever or reach out to us for a formal demo.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/harness-ci-and-harness-cd?utm_source=pmm-ff&utm_medium=how-we-use-our-own-feature-flags-at-harness

Harness CI and Harness CD - Your First True CI/CD Pipeline

Harness now allows you to have both best of breed Continuous Integration and Continuous Delivery capabilities. Let's take a look at what leveraging a Harness CI / Drone Pipeline to build then leveraging Harness CD to deploy to a Kubernetes cluster.Â

The goal of this example will be to have Harness CI / Drone build and push a GoLang Docker Image to DockerHub then have Harness CD pickup on the new artifact and deploy to an awaiting Kubernetes cluster. This potentially can take you from idea to production easier than ever.Â

The Moving Pieces

There are not many moving pieces to see an end-to-end example. You need a Harness CI / Drone Instance, a Docker Hub Account, a Harness Account, and of course, a Kubernetes cluster which could even be Minikube.

Like always you can follow along with the blog and/or watch the video.

If leveraging the example Harness CI / Drone GitHub project, you will have to edit the information to be your own Docker Registry for where the Docker Push goes. I will be referencing mine below.Â

The Harness CD Piece

If this is your first time leveraging the Harness Platform, the quickest way to get started is to leverage a Kubernetes Delegate.Â

The first step of getting a Harness Delegate installed in your Kubernetes Cluster is pretty straight forward.Â

Setup -> Harness Delegates -> Download Delegates -> Kubernetes YAML

Give the Delegate a name.

Hit submit to download. Expand the tar.gz which is downloaded.

Inside the delegate folder, run `kubectl apply -f harness-delegate.yaml` to install the delegate.

In a few moments, your Harness Delegate will be available.

Next, add the Kubernetes cluster for Harness to deploy to by adding the Kubernetes cluster as a Cloud Provider.Â Â

Setup -> Cloud Providers + Add Cloud Provider

Add a Kubernetes Cluster

Give the Kubernetes cluster a name then for Cluster Details, Inherit from selected Delegate [deployed Delegate]. Hit Test then Submit and you are all wired up.

Next, creating a Harness Application is a simple process.Â

Setup -> + Add Application.

The next step inside the Application is to create a Harness Service. You can create a Service by going toÂ

Setup -> Captain Canary -> Services + Add Service. The Deployment Type will be Kubernetes.

Inside the Amazing App Service, + Add Artifact Source from a Docker Registry

Can wire to your Docker Image of choice.

Once you hit submit, the scaffolding will be there for the deployment and no need for additional configuration.

With the Service out of the way, we can wire a Harness Environment to deploy to.Â

Setup -> Captain Canary -> Environments + Add Environment

Once you hit Submit, next can wire an Infrastructure Definition.

Next click + Add Infrastructure and add the Kubernetes cluster.Â The Cloud Provider Type and Deployment Type will be Kubernetes.

Once you hit Submit, can wire together a Harness Workflow to define the steps to deploy.

Setup -> Captain Canary -> Workflows + Add Workflow. The Workflow Type will be Rolling Deployment. Select the Environment, Service, and Infrastructure Definition that was created in the previous steps.

Next you can create a Harness Trigger on the presence of a new artifact in DockerHub. This will trigger the Workflow once the Docker Push is complete in the Harness CI / Drone Pipeline.Â

Setup -> Captain Canary -> Triggers + Add Trigger

Hit Next after giving a Name. DockerHub can send out a webhook or we can use a polling interval from Harness to look out for a new artifact. Without the need to manage webhooks, we can just simply configure On New Artifact. Select your artifact source and can filter on specific tags. For the example we just want to deploy what gets picked up and â.*â is fine as a filter.

Hit Next and define the Actions. Execution Type will be Workflow and will execute the Workflow that was created in the previous steps. Can leverage the Last Collected Artifact. If leveraging the example will pick up on the newest explicit tag with the same filter â.*â.

Click Next and review the Trigger then hit Submit.

Once you hit Submit, all you have to do is kick off a Harness CI / Drone build and push.Â

Harness CI / Drone Steps

If you do not have a donât have a running Harness CI / Drone instance donât worry, not difficult to accomplish. We have a detailed blog and video to get you started. For the example to work will need to be building and pushing a Docker Artifact. The example project has a simple GoLang application. The Harness CI / Drone introduction blog and video go through wiring Harness CI / Drone to a GitHub repository.Â

To kick off a Harness CI / Drone Build and Push, simply modify the configuration / Drone.yaml which Harness CI / Drone is monitoring for. To explicitly build to a specific version, can add tags to Drone.yaml which works well with the example.Â

In the example, pushing build version 1.0.2. Make sure to update your repository information also.

Can modify/increment then commit.

Once you hit Commit, Harness CI / Drone will take over.

Once the publish step is finished, a new image will be in DockerHub.

Watch the Magic

Within the polling interval, Harness will pick up on the newest build/tag and start deploying on your behalf.

With the Deployment kicked off, you have an end-to-end CI/CD pipeline. From code to production!

Partner with Harness in your CI/CD Journey

No matter where you are in your CI/CD journey, Harness can simplify and scale your capabilities. With the lightweight and convention-based Harness CI / Drone and the power of the Harness Platform, achieving CI/CD nirvana has never been easier.Â Get started with Harness CI / Drone and sign up for a Harness Account today!

Cheers,

-RaviÂ

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/harness-policy-as-code?utm_source=pmm-ff&utm_medium=how-we-use-our-own-feature-flags-at-harness

Introducing Harness Policy as Code, Powered by OPA

We're excited to announce Harness Policy as Code, powered by Open Policy Agent (OPA), a centralized policy management and rules service that empowers enterprises to centrally define and monitor policies that are enforced across all delivery pipelines and processes. Harness Policy as Code helps organizations create and enforce policies on deployments, infrastructure, and more, providing developer velocity without sacrificing compliance and standards.

Harness Policy as Code is based on OPA, an easy-to-use, extensible solution for creating and enforcing policies across the entire stack. OPA is an open source project accepted by the Cloud Native Computing Foundation (CNCF) with wide adoption across numerous software delivery use cases. Policies are written as declarative code, so they are easy to understand and modify—from simple to complex use cases.

Harness Policy as Code integrates with CI, CD, and Feature Flags enforcing automated approvals, denials, and other advanced pipeline functionality. Check out our technical documentation to learn more.

Why We Need Policy Management in Software Delivery

As DevOps is adopted within an enterprise, typically one team creates and maintains software delivery and processes. That team has full control and visibility, as they are the creators of DevOps processes within the company. As more business units adopt DevOps within the company, that originating team's manual processes can create a bottleneck, which hampers innovation by limiting team autonomy and slowing down software delivery.

Â In an effort to remove the bottleneck and increase velocity, companies can give development teams more autonomy by allowing them to drive their own DevOps processes. That decentralization of process control can lead to more risks for the company.

When governance is decentralized, development teams can miss quality checks or approvals, introduce vulnerabilities, or break compliance. Organizations need to balance autonomy *and* governance, so they can empower teams with the confidence that they are adhering to all compliance standards and security policies â all without slowing down innovation.

Compliance becomes even more critical in regulated industries, such as financial services and healthcare—not only with enterprise standards, but with third-party regulations, like SOC2, PCI, and FedRamp. It is imperative that all software delivery pipelines meet compliance standards with full auditability; otherwise, the organization is at risk of failed audits, heavy fines, and reputational damage.Â

Centralized management and governance of policies across DevOps processes allow enterprises to define standards for the entire organization while enforcing compliance with regulations. Policies enable individual teams to have autonomy over their processes with oversight and guardrails in place to prevent them from straying from standards, ensuring secure and compliant software delivery.Â

Harness Policy as Code Features

Harness Policy as Code is a centralized policy management and rules service that leverages OPA to meet compliance requirements across software delivery. HPE enables organizations to centrally define and monitor policies that are enforced across all delivery pipelines and processes.Â

Policy as Code features for writing and enforcing policies include:

- A Policy Editor that enables developers to start writing policies-as-code quickly. With a library of policies to start from and a testing terminal, developers can try out policies on real inputs during development before enabling them.

- Policies that are configured to be automatically enforced on Harness processes (e.g. on Pipeline Run, on Feature Flag save).
- The ability to set severity, so a policy violation can issue a warning or throw an error to stop processes from continuing.
- An audit trail that can maintain a full history of policy evaluations with detailed outputs for audit and compliance.

With the release of Policy as Code, policies can now be enforced on CI and CD pipelines and Feature Flags.

Pipeline policies govern the requirements of delivery pipelines, and they can be automatically enforced when the pipeline is saved or triggered, or even in the middle of pipeline execution. Policies can enforce specific pipeline configuration, advanced access control use cases, runtime validation, and more. Here are some examples of what the Policy as Code can do:

- Require an approval step before deployment to production.
- Forbid use of Shell scripts in the pipeline.
- Only allow deployment to approved namespace.
- Only allow deployments from approved container registries.
- Validate test step outcome meets minimum threshold before allowing the pipeline to continue.

Policies for Feature Flags are enforced when the flag is updated or toggled on/off, enabling policies for adhering to standards, flag process, and hygiene. This includes:

- Only allowing creation of boolean flags.
- Enforcing flag naming conventions.
- Enforcing when creating a flag the default on and off values must both be false.
- Requiring a Feature Flag be enabled in QA before it can be turned on in Production.

Policy as Code centralizes and standardizes policy management across software delivery, allowing engineering leaders to empower dev teams to own their tools and practices while ensuring that everyone is following company standards for compliance and security. With guardrails in place, security vulnerabilities won't be introduced as development teams are writing their pipelines. Leaders can rest assured that compliance standards are being met, with full auditability of policies and failures, and they can find and report breaches as early as possible with shift-left governance.

Get Started

Check out our platform governance page to learn more about how Harness' modern approach to software delivery governance empowers teams with stable processes that don't slow down delivery, or request your personalized demo today.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

[Documentation](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[Case studies](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[We want to hear from you](#)

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

[Sign up for our monthly newsletter](#)

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/feature-flags-pipelines?utm_source=pmm-ff&utm_medium=the-journey-to-using-feature-flags-at-their-full-potential

Announcing: Feature Release Workflows With Feature Flags Pipelines

We're thrilled to announce the introduction of feature release workflows for Harness Feature Flags, called Pipelines. Users of Feature Flags can now standardize the process and steps required to release a feature, and layer in automation to make feature release as simple as hitting "Go".

Feature Release Management Is A Chore - Schedules, Approvals, Monitoring - Oh My!

The most common implementation of a feature flags solution is one that is built in-house. In fact, most of the time people don't even realize they're building a feature flags solution! And that makes sense. There are a number of point problems that can be solved with feature flags, and engineering teams want to solve those. But before they know it, it's turned into another internal tool to maintain and upgrade.

What makes this difficult is two-sided:

Process is important, especially as organizations seek to maximize the value of the feature flag solution that they have built or bought. Even for those who have bought a tool, it's critical that they are able to create a standardized release process that anyone can follow, and that guarantees that any feature release has gone through the appropriate steps. And let's not forget about being able to audit releases for compliance purposes - or because something went wrong and a root cause needs to be identified.

When we look at all of these things together, it becomes obvious that all of the requirements are interrelated. Let's be clear about the problems that need to be addressed. And who knows, maybe there's a way to solve them all at once!

- Integration into the release process: monitoring, approvals, scheduling, etc.
- Standardization of feature releases across teams, products, projects, etc.
- Ability to view audit trails for compliance and root cause analysis.
- Creating a common framework that anyone can work off of.

You can imagine that being able to do this would be a massive burden taken off the shoulders of anyone wanting to use a feature flag solution, whether they're a developer or a manager. If you were able to solve these problems, you'd essentially make feature releases non-eventful.

And maybe you could even automate it?

Enter Pipelines: Feature Release Management Workflows

To solve these problems, we built Pipelines for Harness Feature Flags. Pipelines take all of the requirements and problems to be addressed above and put them into a single construct that is significantly easier to comprehend and use than the traditional hodge-podge of tools, logs, scripts, and emails.

Building Workflows for Common Release Requirements

Check out this simple pipeline below. Even though there are only three steps, it clearly outlines what you want to happen: turn on the feature; get an approval; and then release to the rest of the beta segment of your customers.

If this were your actual release process, you could easily take this pipeline and turn it into a reusable template. Now, every time you have a new feature, you could just click a button and feel good that all steps will be taken care of. Of course, these can get more complex as well. Let's take a look at what that could be:

Part of what makes Pipelines in Harness Feature Flags really useful is the deep library of integrations into tools you and your team probably use. Now we can expand the simple use case above where only an approval is required. The workflow could include updating Jira tickets, sending Slack updates, verifying logs or health metrics. Basically, anything you might need in order to release a feature.

Now you've taken all these disparate activities and rolled them into a single, manageable workflow. If something goes wrong, you can easily access the audit logs to see what changed, who changed it, when, and so on. You also have a visual pipeline, should you choose to use it, that you can use to quickly spotcheck the progress of a feature rollout or verify that the right steps are included.

Automation of Feature Releases

It's not just about eliminating the toil and headache associated with feature releases by creating templated workflows. You might have picked up on the nature of Pipelines, which is that it can be automated.

Once a Pipeline is created to meet the needs of your workflow, you can associate any feature and simply hit the proverbial "Go" button.

From there, the Harness Pipeline will take care of going through each of the steps that you've laid out. It can automatically handle rolling out the feature, pinging people for approvals, making changes in Jira, checking health metrics - you name it. Of course, if there are issues that require attention or failures in the process, you will be alerted. This leaves you to do things other than babysit the release pipeline, and you only need to be involved if something goes wrong.

In the near future, we're also looking to integrate automated health metric verification that will take this a step further. Imagine you want to roll out a feature to a subset of your users, verify that things work as expected, or that the right business metrics are moving, and then progressively roll that out to larger portions of your user base. Or, if something goes wrong, you want to take some pre-specified remediation action. All of these things will be possible, meaning that you will truly be able to simply declare what you want, and the system will take care of making it happen.

Added Value of the Harness Platform

As an end-to-end software delivery platform, it wouldn't make much sense *not* to have Harness Feature Flags have some interplay with the rest of the software delivery process. We commonly see that feature flags are an amplification method for CI/CD and in many ways are the third step of the CI/CD pipeline.

Harness is actually the only feature flag solution on the market today that natively integrates with CI/CD. Why does that matter? And what value does that provide to a product development or ops organization? And why do executives care about it too?

Let's start with a topic we've already covered - automation. The same Pipeline feature in Harness Feature Flags is also available in Harness Continuous Integration and Harness Continuous Delivery. Yes - you get a unified pipeline. Suddenly, you go from disconnected CI/CD and feature flagging to a single pipeline that shares context and can be automated from build through to individual feature delivery to end users. Can you imagine what that could mean for your software delivery process? But that's not the only reason it matters.

Analytics, governance, compliance, risk - all are other really important reasons to have CI/CD and feature flags linked up. By having shared context across the entire SDLC, you're able to do all of the above better:

- Create a canonical analytics dashboard to track software delivery metrics at each stage, and the interplay between them.
- Enforce org-wide governance policies for software delivery, and ensure that governance standards are always met.
- Leverage audit trails across every stage of software delivery to see exactly what's happening and where, and simplify external and internal audits.
- De-risk release at any stage because you're not worried about inter-tool integrations or support and critical context being lost that can impact the business or customers.

Of course, you can go into more detail about any of these topics on our blog, which speaks at length about each of these topics.

How to Get Started

If you're already a Harness Feature Flags user, you can head over to the product and you'll see a new menu item labeled "Pipelines" that you can click and get started building your own!

If you haven't signed up to use Harness yet but want to get started, you can easily sign up for a free trial. Happy developing!

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

[Documentation](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[Case studies](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?
Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/what-is-continuous-delivery?utm_source=pmm-ff&utm_medium=the-journey-to-using-feature-flags-at-their-full-potential

What is Continuous Delivery?

The rise in meal prepping or getting pre-prepared meals delivered to oneâs home has been on the rise. As software engineers we prep in a similar fashion. We build images in one of our favorite container formats let that be Docker, Mesos, or RKT, to name a few. Though as we build these up, how do we get these into running containers? Sure, we can leverage our laptop or a segregated pre-prod environment. The last thing we want to do is throw away our prepped meals because they went bad and nobody ate them.

My Image is in Nexus, but am I Done?

Congratulations, your Docker Push was successful into your artifact repository of choice e.g Nexus/Artifactory/Harbour. A great feeling as the label increments to the next version. Time for a snack and maybe tackling for lunch one of those meal preps you worked on earlier in the week. As you are heating up your meal, hit refresh on your application on your phone and the changes are not there. You jump back on Slack to check for any alerts that the deployment went through and nada. What is going on?

Ok, where is the deployment alert, really?

I did assume that Slack is part of your chat-ops stack. Though if your firm does not use some sort of chat-ops mechanism for alerts, donât fret. The salient point is that you did hard work to get an image ready to turn into a running container somewhere else than your laptop and that is not happening. There is a great goal that containers will harmonize all of our environments and allow for push or easy button deployments. Even more so if using an orchestrator like Kubernetes or Marathon this should not even be an issue for the easy button deploy; send to the orchestrator and let the orchestrator figure the deployment outright. If only things were this easy.

Not So Fast, We Still Have To Deploy

Letâs not forget the rigor and discipline of deploying software. We are certainly talented engineers bettering our craft along the way, but we donât want to be the ones on the fault end of a production outage. The adage âwork fast and break thingsâ coined by FaceBook most likely does not apply to the majority of us. Back to our lunchtime deployment. We ate our meal prep and since we are out of food are faced with a choice: find another snack or go back to our laptop and figure out what is going wrong. We log into our artifact repository to validate our latest edition is there and even for sanity sake do a Docker Pull on what we just created. When you execute Docker Run, your changes are there. Now getting them into the wild is another story (aka your pipeline).

Welcome to Your Pipeline

As we move from stage to stage in our environments, we work to build confidence that the changes we introduce will function as designed and not degrade or break our applications. There is a host of testing, code coverage, deployment, and roll back strategies to ensure this, not to mention the underlying infrastructure and potentially regulatory/change control requirements needed for a good home for your hard work. With the advent of GitOps, a commit/merge can be the trigger needed to get your changes from code to an image and eventually deployed into a container/pod in your orchestration tool of choice. With all that is required to get your image into production, is slowing down or opening up lunch for tomorrow an option as you want to work on the next set of features? The goal here is to have âwhen your changes are ready â those changes go out / be deployed. *This is continuous delivery.*Â

Nirvana Around the Corner

At Harness, we have taken the complexities of continuous delivery and address them with convention and simplicity. Anyone can build a pipeline that declaratively accomplishes the most complicated delivery tasks. Deploying a canary release to Kubernetes? No problem. We released the Community Edition of our Harness Platform to allow anyone to have the ability to have a strong continuous delivery pipeline.

Get your Community Edition now!

We would be thrilled at Harness for you to sign up for your free account to realize the dream to get your artifacts out quicker than you

can eat lunch. We have also launched the Harness Community to foster a sense of community amongst those looking for the next generation of continuous delivery. Get ship done!-Ravi

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/feature-flags-policy-governance-tutorial?utm_source=pmm-ff&utm_medium=the-journey-to-using-feature-flags-at-their-full-potential

Feature Flags Policy Governance Tutorial

Harness Policy as Code, powered by Open Policy Agent (OPA), allows you to write policies as code to automatically govern flag configuration, usage, and process. Policies are applied on every flag creation, toggle, or modification, ensuring your feature flags always stay in a compliant state.

We are proud to be the first feature management solution to support this capability. Global governance policies for feature flags are a big win for engineering organizations that need to enforce standards at scale. They allow for guardrails to be put in place across all releases to ensure standards are met, and they also automate the process, so the developer experience doesn't change. Developers simply get error messages the way they're used to during the build and test phase.Â

But this post isn't about why you should do it â it's about how you do it. In this post, we go in depth and teach you all about the Feature Flags policy governance: architecture, use cases, data available to utilize, and more.

Architecture

Here's a quick overview of how we ensure that your flags always remain in a compliant state, regardless of what changes a user makes or how they decide to make them.Â Â

Writing Policies

The Harness Policy Engine is based on OPA, an easy-to-use, extensible solution for creating and enforcing policies across the entire stack. OPA is an open-source project accepted by the Cloud Native Computing Foundation (CNCF) with wide adoption across numerous software delivery use cases within the CI/CD pipeline. Policies are written in Rego as declarative code, so they are easy to understand

and modify from simple to complex use cases.

You can find more information about writing Rego rules in the official OPA docs.[Â](#)

Once you're ready to get a policy out there, you can write your own and add it to Harness. We also provide some policies out of the box for common use cases, like enforcing naming conventions and ensuring proper promotion of code.

What Data is Available to Write Policies Against?

We know that policies not only relate to the changed entity in question, but can also depend on contextual data, such as who made the change and when. To support these use cases, we provide an ever growing collection of data to the policy engine that you can use when writing your policies. This includes:

- The full feature flag configuration for the created/updated flag. This includes flag name, description, variations, rules, flag state in every environment you have, and much more.[Â](#) This is in json format, which matches the format of our public API docs, so if you interact with your flags as code already, it will be very familiar.[Â](#)
- Which user made the change, what RBAC permissions they have, and which user groups they belong to.
- When the change was made.[Â](#)

Here's a truncated example of the metadata sent to the policy engine

Debugging Policies

Policies are written as code, and as anyone who writes code knows, there will inevitably be lots of edge cases or error scenarios you want to test while refining your policies. Having to create and configure flags in special ways just to test your policies is^Â frustrating and time consuming. For this reason, we have an integrated policy tester/debugger built right into the UI.[Â](#)

Using this debugger, you can go through a standard development process:

- **Develop:** Begin by writing your policies in the main text area. Add as many rules and helper functions as you'd like.
- **Test:** Using the testing terminal, you can get quick feedback on if the new policies you're building are having the desired effect. You can hit the **Select Input** button to populate the input field with real flag data from your own project, giving you reliable and realistic test scenarios for your use cases. You can then hit the **Test** button to run the policies and see for the given data input if it would succeed or fail. You can also manually edit this input^Â data if you'd like to hand craft particular edge cases.
- **Debug:** Once your policies are being enforced there may be a time that a user isn't sure why a change was blocked. This can be a frustrating experience if you're only provided with a vague error message such as **change forbidden**.[Â](#) Luckily, we provide all the tools for a user to view detailed information on exactly which policies failed, along with the ability to click on these policies and enter this debugger mode, viewing the policy and the exact input their change produced. This will help them verify if it's a valid rejection or if the policy itself needs to be modified going forward.

Use Cases

Harness Policy Engine supports a wide array of use cases. These range from simple sanity checks that description fields have been properly completed to complex corporate policies that prevent changes during blackout periods. Feature flag policies broadly fall into these two categories:

Flag Configuration

- Naming conventions, e.g. flag names must match jira ticket format
- Mandatory descriptions
- Only allowing boolean flags to be created (no multivariates)
- Banning certain functionality e.g. no prerequisite rules allowed
- Max number of specific target rules

Change Management^Â

- Flag cannot be enabled in production unless it is enabled in QA first
- Flag changes must be made via pipelines with an approval step
- No changes during certain time periods e.g. when tests are running or during certain mandated blackout periods
- Flags can't be enabled by the same user that created it

You can get as creative as you'd like around the rules you need to enforce. We've purposely not taken an opinionated UI wizard-based approach on how to create and combine these rules, so you're free to experiment, start small, and govern what matters to you.[Â](#)

Where Can I Try It Out?

To learn more about how Harness manages policies, check out our Policies Overview for Feature Flags documentation, or sign up for a free Feature Flags trial today.[Â](#)

â

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/progressive-delivery?utm_source=pmm-ff&utm_medium=the-journey-to-using-feature-flags-at-their-full-potential

Progressive Delivery: Everything You Need to Know

Progressive delivery is a practice that builds on the capabilities of Continuous Integration (CI) and Continuous Delivery (CD) to help you deliver with control. Software delivery helps organizations and teams get their changes out to customers quickly. Often, what gets forgotten is discussing how we ensure quality and reduce the risks of introducing said changes to customers. Delivering changes rapidly can often be at odds with not breaking things. This blog post will share what you need to know about progressive delivery as we share the ways organizations are using it today.

What is Progressive Delivery?

Progressive delivery is a practice that allows organizations to control how and when new software features or changes are delivered. It builds on the capabilities and practices of feature flag management and deployment strategies like blue-green and canary deployments. Ultimately, progressive delivery combines software development and delivery practices allowing organizations to deliver with control.

When discussing the software development life cycle (SDLC), we often mention continuous integration (CI) and continuous delivery (CD), which captures the how, when, where, and why of our software delivery. CI/CD is integral to quickly, safely, and repeatably we can get our software changes to our users and customers. Progressive delivery leverages your CI/CD pipelines alongside our software development and delivery practices to boost our software delivery capabilities.

The Key Elements of Progressive Delivery

There are several key components to achieving progressive delivery.

Source Code Management and Git Branching Strategy

Source code management is a practice that allows for the reverting, tracking, and correction of software code. A key element of progressive delivery includes application development code that is properly managed and tracked by a version control system like Git. In a basic Git-based workflow, you have the main branch from the main branch, which contains different versions of your application's code, branch from specific versions of the application code when working on a different feature. This allows for development progress to continue without the fear of losing changes as developers work on the same codebase.

Having source management is a key element for progressive delivery as it serves as the foundation for feature development, testing, and rollout.

Continuous Delivery, Continuous Integration, and Feature Flags

CI/CD is a shortened term for Continuous Integration and Continuous Delivery. CI/CD is the combination of principles, practices, and capabilities that allow for software changes of all kinds to get users in a quick, repeatable, and safe manner. This allows for software developers to integrate their feature or service changes continuously and for IT and operations teams to deliver with the standards, security, and confidence businesses need. CI/CD also serves as a foundation for your progressive delivery capabilities. Automated testing ensures that these integrations and deliveries maintain the desired quality.

You can learn more about CI/CD in this blog post.

Feature Flag Management

As a development codebase grows, so often does the development team. With new feature requests, developers and innovation syncing and aligning to a development pace can be difficult, especially for complex feature work. There are many challenges that long-running feature work face when it comes to delivering fast. Merge conflicts arise for long-lived development branches, and often, features require manual testing. Feature flags, also known as feature toggles, control the visibility of features behind a logic (often Boolean) flag. This allows teams to avoid redeploying an application to enable a feature for testing or production.

Feature flag management also has many benefits to development workflows. Contributors to a codebase don't have to wait for a feature to be fully finished to merge into the development or main branch. Developers can also test their features by controlling their feature flag and exposing their feature in a running application in a test or development environment.

A/B Testing

A/B testing, also known as split testing, is a technique that involves exposing two variants of the same application service to different segments of users. By shifting specific over to different versions of an application, teams can compare, experiment, and better understand how these changes impact a specific set of users. To effectively conduct A/B testing, it's essential to know how to create a flag that can toggle between the two variants.

Feature Release Strategies

Also known as feature rollout, these deployment or release strategies are used to change or upgrade a running instance of an application. There are three kinds of deployment strategies, rolling deployments, blue-green deployments, and canary deployments. Of these three strategies, two are progressive since they involve progressively shifting user traffic to a running service.

The Blue-Green deployment is a deployment strategy that gradually transfers user traffic from a previous version(green) of an application or service to a new release(blue) of the application or service.

The Canary Deployment is a deployment strategy that releases an application or service to a subset of users.

Both blue-green and canary deployments limit the impact of application changes. These capabilities can be beneficial if deployments are risky and have severe consequences to the user base in breaking changes.

Observability

Observability leverages metrics, tracing, and logging capabilities to provide organizations and teams with answers to questions regarding their running services without the need to deliver another instance of the application or service to answer those questions. Understanding your services interact or perform can give developers a better understanding of what is breaking and what is performant. Often we need to understand an application's performance before we can achieve the confidence needed to release a change to an entire user base, and this is where observability matters.

Implementing Progressive Delivery

The best approach to implementing progressive delivery is to build the key elements of progressive delivery incrementally. A common practice is to invest in your development workflows. This can involve introducing feature flag management capabilities through an SDK or by building it a home-grown solution. Some other organizations may need to introduce source code management practices first before even discussing A/B testing capabilities or feature flag management. If you're starting with software development and would like to read more about git-based workflows and how development and feature branching works, take a look at this source-code management guide.

Some organizations use feature flag management within their development processes already but need more capabilities around verifying deployments. It may make sense to invest efforts into monitoring or observability solutions to measure and understand application services. You can't compare or baseline performance without these capabilities. And changes go out with confidence, so this can often

be the next step in your progressive delivery journey.

And lastly, there is a deployment aspect to progressive delivery. Your platforms should support deployment strategies like canary or blue-green deployments. Deployment capabilities can come from your continuous delivery platform or other solutions like a service mesh. If you're looking for a complete solution to deploy your services to production safely, I recommend looking at Harness's CI/CD platform.

Improve your Software Delivery with Harness

Today software delivery isn't just about speed. Progressive delivery joins software development and delivery practices to better support and control your software delivery. If you're looking to move fast without breaking things and implementing any progressive delivery element quickly, I recommend trying Harness for free.

For more on progressive delivery, let's look at Feature Flags!

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/progressive-delivery?utm_source=pmm-ff&utm_medium=how-we-use-our-own-feature-flags-at-harness

Progressive Delivery: Everything You Need to Know

Progressive delivery is a practice that builds on the capabilities of Continuous Integration (CI) and Continuous Delivery (CD) to help you deliver with control. Software delivery helps organizations and teams get their changes out to customers quickly. Often, what gets forgotten is discussing how we ensure quality and reduce the risks of introducing said changes to customers. Delivering changes rapidly can often be at odds with not breaking things. This blog post will share what you need to know about progressive delivery as we share the ways organizations are using it today.

What is Progressive Delivery?

Progressive delivery is a practice that allows organizations to control how and when new software features or changes are delivered. It builds on the capabilities and practices of feature flag management and deployment strategies like blue-green and canary deployments.

Ultimately, progressive delivery combines software development and delivery practices allowing organizations to deliver with control.

When discussing the software development life cycle (SDLC), we often mention continuous integration (CI) and continuous delivery (CD), which captures the how, when, where, and why of our software delivery. CI/CD is integral to quickly, safely, and repeatably we can get our software changes to our users and customers. Progressive delivery leverages your CI/CD pipelines alongside our software development and delivery practices to boost our software delivery capabilities.

The Key Elements of Progressive Delivery

There are several key components to achieving progressive delivery.

Source Code Management and Git Branching Strategy

Source code management is a practice that allows for the reverting, tracking, and correction of software code. A key element of progressive delivery includes application development code that is properly managed and tracked by a version control system like Git. In a basic Git-based workflow, you have the main branch from the main branch, which contains different versions of your application's code, branch from specific versions of the application code when working on a different feature. This allows for development progress to continue without the fear of losing changes as developers work on the same codebase.

Having source management is a key element for progressive delivery as it serves as the foundation for feature development, testing, and rollout.

Continuous Delivery, Continuous Integration, and Feature Flags

CI/CD is a shortened term for Continuous Integration and Continuous Delivery. CI/CD is the combination of principles, practices, and capabilities that allow for software changes of all kinds to get users in a quick, repeatable, and safe manner. This allows for software developers to integrate their feature or service changes continuously and for IT and operations teams to deliver with the standards, security, and confidence businesses need. CI/CD also serves as a foundation for your progressive delivery capabilities. Automated testing ensures that these integrations and deliveries maintain the desired quality.

You can learn more about CI/CD in this blog post.

Feature Flag Management

As a development codebase grows, so often does the development team. With new feature requests, developers and innovation syncing and aligning to a development pace can be difficult, especially for complex feature work. There are many challenges that long-running feature work face when it comes to delivering fast. Merge conflicts arise for long-lived development branches, and often, features require manual testing. Feature flags, also known as feature toggles, control the visibility of features behind a logic (often Boolean) flag. This allows teams to avoid redeploying an application to enable a feature for testing or production.

Feature flag management also has many benefits to development workflows. Contributors to a codebase don't have to wait for a feature to be fully finished to merge into the development or main branch. Developers can also test their features by controlling their feature flag and exposing their feature in a running application in a test or development environment.

A/B Testing

A/B testing, also known as split testing, is a technique that involves exposing two variants of the same application service to different segments of users. By shifting specific over to different versions of an application, teams can compare, experiment, and better understand how these changes impact a specific set of users. To effectively conduct A/B testing, it's essential to know how to create a flag that can toggle between the two variants.

Feature Release Strategies

Also known as feature rollout, these deployment or release strategies are used to change or upgrade a running instance of an application. There are three kinds of deployment strategies, rolling deployments, blue-green deployments, and canary deployments. Of these three strategies, two are progressive since they involve progressively shifting user traffic to a running service.

The Blue-Green deployment is a deployment strategy that gradually transfers user traffic from a previous version(green) of an application or service to a new release(blue) of the application or service.

The Canary Deployment is a deployment strategy that releases an application or service to a subset of users.

Both blue-green and canary deployments limit the impact of application changes. These capabilities can be beneficial if deployments are risky and have severe consequences to the user base in breaking changes.

Observability

Observability leverages metrics, tracing, and logging capabilities to provide organizations and teams with answers to questions regarding their running services without the need to deliver another instance of the application or service to answer those questions. Understanding your services interact or perform can give developers a better understanding of what is breaking and what is performant. Often we need to understand an application's performance before we can achieve the confidence needed to release a change to an entire user base, and

this is where observability matters.

Implementing Progressive Delivery

The best approach to implementing progressive delivery is to build the key elements of progressive delivery incrementally. A common practice is to invest in your development workflows. This can involve introducing feature flag management capabilities through an SDK or by building it a home-grown solution. Some other organizations may need to introduce source code management practices first before even discussing A/B testing capabilities or feature flag management. If you're starting with software development and would like to read more about git-based workflows and how development and feature branching works, take a look at this source-code management guide.

Some organizations use feature flag management within their development processes already but need more capabilities around verifying deployments. It may make sense to invest efforts into monitoring or observability solutions to measure and understand application services. You can't compare or baseline performance without these capabilities. And changes go out with confidence, so this can often be the next step in your progressive delivery journey.

And lastly, there is a deployment aspect to progressive delivery. Your platforms should support deployment strategies like canary or blue-green deployments. Deployment capabilities can come from your continuous delivery platform or other solutions like a service mesh. If you're looking for a complete solution to deploy your services to production safely, I recommend looking at Harness's CI/CD platform.

Improve your Software Delivery with Harness

Today software delivery isn't just about speed. Progressive delivery joins software development and delivery practices to better support and control your software delivery. If you're looking to move fast without breaking things and implementing any progressive delivery element quickly, I recommend trying Harness for free.

For more on progressive delivery, let's look at Feature Flags!

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

[Documentation](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[Case studies](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[We want to hear from you](#)

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

[Sign up for our monthly newsletter](#)

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

5 Feature Flag Use Cases You May Not Have Thought Of

We talk to folks about feature flags every day, and usually we hear that people have pretty clear ideas about how feature flags can create value for them. However, when we start talking about feature flag use cases and the kinds of ideas we're building Harness Feature Flags to address, a lot of teams tell us that they'd never considered using feature flags for some of these purposes.

So, based on our own experience, we wanted to share some super valuable ways to consider using feature flags that you may not have thought of yet.

1- APIs and Backend

When people think about feature flags, they often think about visible, frontend changes. But, with our server-side SDKs, they can just as easily roll out backend changes with flags.

As one clear example, by using feature flags, you can more easily control migrating backend infrastructure in an experimental way - with less coordination around deployments and rollbacks.

To use a logging service as an example, if you were migrating logs from one service to another but wanted to turn it on and off to collect errors and bugfix, by using feature flags you can turn the traffic on and off easily without any need to deploy or roll back in real time as you switch the code paths out.

2- Performance Refactoring

Sometimes, when you build the first version of a feature, the biggest optimization is not around streamlining your queries or optimizing for pure performance. And, sometimes, as a result of this, your frontend engineers are quite insistent that they be given some time to greatly reduce the number of calls the frontend is making to the backend - not that this has ever happened to us, of course.

By using feature flags for this kind of refactor, you get two huge new benefits. First, you can easily test it both in prod and up to prod to verify that the intended result is not impacted by your refactor. But, just as importantly, you can also start to quantify the impact of that change by turning it on for a subset of your traffic first, see how your metrics change, then expand it over time. All of this, just like in the above scenario, without re-deploying or rolling back as you go. As far as feature flag use cases go, this is pretty invaluable.

3- Ops Toggles

Ops use cases are not the first things that come to mind with feature flags, but the more you think about it, the more you might see that a significant amount of ops work can be permanently wired up to flags to provide a better UX and a more manageable experience for critical op tasks.

Think about an archiving system that runs 24/7, but occasionally needs to be toggled off during outages caused by upstream issues or high load. Many ops teams have scripts or runbooks for this, or they just know what code to change. Turning this into a feature flag means teams start to actually build control panels for critical kill switches, and settings that are much easier to use, document, and communicate.

4- Compliance

Here at Harness, we run different environments for different needs. These needs can be government, European regulation, state regulation, and on-prem. Some of these environments need things modified slightly, or turned off entirely.

Feature flags provide a great solution. Want to turn something on only in Europe to help with GDPR? Flag it and target your European users! Want to turn something off for government or on-premise users? Feature flag management software makes this easy, without the need to maintain and release tons of different versions of your code - or having to layer logic inside your application. Feature flags also make this sort of control explicit. Because it's visible, auditable, and controlled by RBAC, you will keep your security and compliance teams happier and less in the dark.

5- Free Trials

A common request in the sales cycle is that a potential customer would like to try out a feature on a high-level plan, or maybe even try out the entire product without them needing the matching license.

If you've ever been on the engineering side of building these controls out, it can be a huge pain - and very little of it is value-enhancing to the core product experience. Messy and inconsistent internal admin panes begin to sprawl: sales and support teams often are terrified of using these important-feeling systems.

Feature flags let you easily provide control for this sort of account management without the need for complex custom logic or the overhead of custom admin portals. Simply wrap features in flags and let your account teams manage them!

Conclusion

As more organizations adopt feature flags, we will begin to see feature flags as more than just software switches to hide or activate

features. Feature flags really serve entire organizations in new and powerful ways, simplifying life for engineering and product teams while bringing other stakeholders closer to the product than ever before.

Want to try feature flags and see how it could change your life? Request a demo today - who knows, maybe you'll discover new feature flag use cases to serve your organization!

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/what-are-feature-flags?utm_source=pmm-ff&utm_medium=the-journey-to-using-feature-flags-at-their-full-potential

What are Feature Flags and How Do We Use Them?

Building software is a great experiment. Many times we are required to do what has not been done before; this is core to innovation work. However, a lot of the iteration and experimenting tends to happen before software is shipped. That's where the concept of test in production comes into play. Building something from scratch or even jumping into an older project mid-stream requires lots of iteration.Â

Once we deploy, for many software engineers, the experimentation on the existing tends to drop off and onwards to the next set of features in the backlog. Product owners/managers typically collect feedback to help prioritize the next set of features. From a software engineering perspective this can feel a little âchicken or egg-ishâ - without a feature actually deployed in a production environment to get feedback, how do we if know we are on the mark?

Letâs illustrate this with an example. Customers are asking for an updated UI view for their dashboards; they want something that surfaces the most relevant information faster. Now that the product team has collected those requirements, itâs up to engineering to figure out how to best build it. There might be a few potential solutions:

With some potential solutions decided, now itâs up to the engineering team to decide which of the three to implement. And before a customer ever sees it, they have to choose 1 of 3, build out the MVP over the next few sprints, and then ship it. This is the first time theyâll get any feedback, and if itâs not the right solution then it has to be built again!

Now, imagine if this new feature cycle could be improved for product development teams. If youâve been paying attention to developer tools over the last few years, you might have seen a new category popping up: feature flags. Feature flags as a concept were born to

increase development velocity and to solve this very problem through practices like progressive delivery.

While feature flags as a mature category of hosted tooling is new, feature flags - and what they solve - have been around a little longer. Let's take a look at what feature flags have grown into, and how to use them.

The Story of Feature Flags

Feature flags, at their core, are the ability to wrap different versions of your code in conditional statements that you can turn on and off at will. You may have previously thought of them as app configs, or even part of your company's rules engine.

With Harness Feature Flags, we've taken these concepts you may be familiar with and taken them even further - adding more sophisticated rules, governance, user management, an intuitive UI, and more.

Feature Flags lets you separate feature release from code deployment, build more effective strategies for rolling things out and testing with your users, and have more insight than ever into how changes impact your application performance.

Ultimately, feature flags give engineering teams the ability to deliver more features, with less risk. For leaders, it's a great way to improve engineering velocity and developer experience, and to reduce the risk associated with feature development. And for developers, it's a great way to work more efficiently, with less stress - think fewer late nights before a deployment, and fewer firefights post-deploy.

What Are Feature Flags (or Feature Toggles)?

Feature flags are a way that developers can conditionally turn certain sections of their code on or off. You can think of feature flags as extending Continuous Delivery into Continuous Deployment - a way to put changes into production behind a flag and turn them on in a controlled way later (or hide and remove them in the same way).

But you can also think of feature flags as a new way to think about building and releasing applications - by clearly defining features and components that can be changed and toggled on and off individually, you allow for experimentation, controlled rollouts to different users, as well letting more people (like PMs, sales, and support) turn things on and off for customers.

Together, this results in giving more power and control to developers to create more and better features faster, while at the same time ensuring that engineering is not the bottleneck for the business in having to make changes for individual customers or ensuring a perfect production deployment for every new feature or feature update.

If we want to simplify even further, feature flags essentially create private swim lanes for developers where they can ship a feature directly to customers and then have control over who sees it, get feedback, and turn it on and off as needed. It's like a light switch (a pretty complex one)!

Who Uses Feature Flags?

People oftentimes consider feature flags either an engineering tool, or a tool for product managers. The reality is, it's both. Flags can help software development and DevOps teams lower their overhead and increase their velocity, and they can help product managers better control releases, coordinating launch timings and creating a feedback loop more effectively.

Research indicates that 95% of engineering leaders want to implement a feature flag solution for themselves. This comes on the back of 97% of leaders saying they're under pressure to deliver faster, and 65% of whom say they find it difficult to achieve that velocity increase in a safe manner. It's abundantly clear that engineering teams are likely the biggest beneficiaries of implementing and using a feature flag solution, and indeed, they tend to be the primary users. And when they do implement a feature flag solution, they end up delivering 66% more features per application per year!

Feature flags are great for users across the organization beyond product and engineering. For sales and support teams, feature flags provide a way for them to directly manage betas and new features for their customers, and to track down who's using what in case of any issues. And for management, flags can provide new forms of visibility into what's happening in development and how new features are being tested with users.

On a fun fact kind of note and since we're in the "who uses feature flags" section, it's cool to note that big companies like Netflix, Flickr, Google, Reddit, and more use feature flags! They're definitely worth looking into and absolutely are an important part of the software delivery lifecycle.

Why You Should Consider Feature Flags

Whether or not the use of feature flags is important to your software delivery lifecycle comes down to what it is that you need to accomplish. While feature flags are a great tool to have for any team, they are particularly useful if you have these problems:

Problems in the Business

- The risk associated with new feature releases ends up slowing down developers since they have to thoroughly validate design and implementation before releasing.
- There are often multiple possible versions of how a new feature will look and function, and it's up to the developer to choose one (not always the right one) and implement it.
- The decision of when to release new software features is controlled by developers instead of the business.

- Sunsetting old features introduces risk to newer features that might have dependencies on the old code.
- It's all or nothing - a feature is rolled out to everyone at the same time instead of doing a feature rollout incrementally for verification purposes.
- When teams are beholden to engineering release cycles (e.g. once a month), multiple features are released simultaneously, introducing complexity and risk that can result in deployment war rooms, large rollbacks, and dissatisfied customers.

Problems in Engineering

- Releasing a feature takes too much coordination and time across stakeholders.
- Merging new features, or managing separate feature branches for changes, is cumbersome and complex.
- You want to test changes to see how they impact your application and end-users before rolling them out more widely.
- You want to provide access to new features, or to test changes with certain portions of the user base sooner - by location, by customer plans, by customers opting into beta testing, or by any other criteria or logic you can think up.
- You want to reduce the risk of rolling out new code by having instant kill switches that don't require rollbacks or new deployments to fix a disrupted environment.

To simplify, we can consider that feature flags are a great solution if you're trying to do the following things:

- Increase engineering velocity and ship more features.
- Decrease feature development risk by A/B testing multiple solutions with customers.
- Reduce the risk and toil associated with production failures and rollbacks.
- Decouple engineering deployments from feature unveilings to customers.

Different Types of Feature Flags/Toggles

It's natural to want to put all feature flag use cases in a single bucket, but it's more helpful to instead view flags themselves as a means to an end and as having categories of use cases.

Here are some examples of the different kinds of flags that could be used in a system:

- **Release features** - Using flags to let the right people decide who to turn a feature on for, and when, without complex engineering deployment coordination required.
- **Experiment** - Using flags to learn something about how a change impacts things. This can be a technical experiment, such as how a new UI impacts server load, or it can be a user experiment such as how a new button impacts conversion.
- **Ops flags** - Letting you have permanent control around key parts of your app so you can turn them on and off as necessary without a full new deployment cycle. Add in RBAC, and this is a robust way to control who has access to modify your production application.
- **Control access** - Using flags as a way to manage betas, early adopter list, trial access, and more.

Implementing Feature Flags

Implementing feature flags can seem really simple at first, but it turns out that like with any system, the devil is in the details. Building an awesome feature flag solution in-house certainly has its perks, but it can be deceptively difficult. In particular, we've seen three distinct sets of issues arise with feature flag solutions as an organization's implementation matures.

When a feature flag solution is **nonexistent or immature** and an organization is standing it up for the first time, they tend to run into issues with finding value in the tool quickly. Especially with the need for such a solution and the pressure to make it happen quickly, it's critical that value is demonstrated quickly, otherwise it may end up leaving a sour taste. Within that, there's a dichotomy of architecting for eventual scale versus getting an MVP out the door to prove value. This often results in complex design or architecture that isn't scalable and only works for small, targeted use cases.

As feature flag solutions enter a **middle stage of maturity**, or as they **start to scale**, the problem is quite obvious: scaling is hard. Especially if built in-house, ensuring that the tool can be used by multiple users and multiple teams for a variety of use cases, all while maintaining a common set of best practices and building on a single infrastructure can be incredibly daunting. And as this is being figured out and teams have code in production, it can quickly result in stale code or issues keeping production tidy when code changes occur. Add onto that the need to integrate neatly into everyone's workflow and you've got a behemoth of a problem to solve.

Building on that is the scenario we often see with **highly mature** feature flag implementations, where they primarily end up as separate systems from existing software delivery or CI/CD pipelines that teams now have to learn how to use, and to figure out how to port their existing processes or pipelines into the feature flag solution. At this stage, there's the obvious cost of maintenance and future development to keep up with new changes or requirements. Technical debt is bound to build up by this point, and it can become unmanageable without a dedicated team - and perhaps even its own codebase.

Sure, 95% of engineering leaders may want a feature flag implementation, but as it turns out, it's not so easy to build something that works well over time.

Build or Buy: Feature Flag Management

You may be thinking, "Feature flags seem pretty simple, can't we just build this ourselves?"

The real answer? It's complicated.

We've seen in most cases that an internally-built tool will end up causing a lot of overhead and putting a very low ceiling on the benefits you can receive from feature flags. It may be good for your first couple of flags, but it can be hard to make an internal system scalable, performant, and robust for all the things flags can do as your organization gets more and more comfortable using them - and more reliant on them.

For a deeper look at the build vs buy conversation, you can check out an article we wrote earlier this month, Feature Flags: Should I Build or Buy?

Why Use Harness for Feature Flag Management

Harness provides a feature flag platform made for getting you up and running quickly while being able to scale to any level. We make it easy to get started and quickly see the benefits of feature flags. With Harness Feature Flags, that means you can get your first flag out the door in minutes by following three steps:

- Create a free trial account.
- Create a new flag in just 3 clicks.
- Add one of our SDKs to your code in a few minutes.

And, you've got a flag working!

If that's not quite clear, then it's worth knowing that the fundamental way flags work in an environment like Harness is that SDKs, added to your application as a library, will receive updates on flag statuses to evaluate with your users (you should choose between real-time streaming or pulling updates on intervals). These SDKs are secure, collecting no data you don't want to share, and they are designed for performance and stability.Â

Harness Feature Flags is fast, secure, and focused on helping teams get the most out of feature flags - from your first initial use all the way to global governance, sophisticated reusable templates, and developer-first experiences.

And if you're wondering how we stack up against LaunchDarkly, Optimizely, Cloudbees Feature Management, Split, and other FF tools - those pages are being worked on right now and we'll update this post with the links when they're live!Â

If you'd like a little more information on our architecture and on how to get started, you can check out the feature flag documentation or request a demo.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/what-are-feature-flags?utm_source=pmm-ff&utm_medium=safe-testing-in-production-with-harness-feature-flags

What are Feature Flags and How Do We Use Them?

Building software is a great experiment. Many times we are required to do what has not been done before; this is core to innovation work. However, a lot of the iteration and experimenting tends to happen before software is shipped. That's where the concept of test in production comes into play. Building something from scratch or even jumping into an older project mid-stream requires lots of iteration.Â

Once we deploy, for many software engineers, the experimentation on the existing tends to drop off and onwards to the next set of features in the backlog. Product owners/managers typically collect feedback to help prioritize the next set of features. From a software engineering perspective this can feel a little âchicken or egg-ishâ - without a feature actually deployed in a production environment to get feedback, how do we know we are on the mark?

Let's illustrate this with an example. Customers are asking for an updated UI view for their dashboards; they want something that surfaces the most relevant information faster. Now that the product team has collected those requirements, it's up to engineering to figure out how to best build it. There might be a few potential solutions:

With some potential solutions decided, now it's up to the engineering team to decide which of the three to implement. And before a customer ever sees it, they have to choose 1 of 3, build out the MVP over the next few sprints, and then ship it. This is the first time they'll get any feedback, and if it's not the right solution then it has to be built again!

Now, imagine if this new feature cycle could be improved for product development teams. If you've been paying attention to developer tools over the last few years, you might have seen a new category popping up: feature flags. Feature flags as a concept were born to increase development velocity and to solve this very problem through practices like progressive delivery.

While feature flags as a mature category of hosted tooling is new, feature flags - and what they solve - have been around a little longer. Let's take a look at what feature flags have grown into, and how to use them.

The Story of Feature Flags

Feature flags, at their core, are the ability to wrap different versions of your code in conditional statements that you can turn on and off at will. You may have previously thought of them as app configs, or even part of your company's rules engine.Â

With Harness Feature Flags, we've taken these concepts you may be familiar with and taken them even further - adding more sophisticated rules, governance, user management, an intuitive UI, and more.Â

Feature Flags lets you separate feature release from code deployment, build more effective strategies for rolling things out and testing with your users, and have more insight than ever into how changes impact your application performance.

Ultimately, feature flags give engineering teams the ability to deliver more features, with less risk. For leaders, it's a great way to improve engineering velocity and developer experience, and to reduce the risk associated with feature development. And for developers, it's a great way to work more efficiently, with less stress - think fewer late nights before a deployment, and fewer firefights post-deploy.Â

What Are Feature Flags (or Feature Toggles)?

Feature flags are a way that developers can conditionally turn certain sections of their code on or off. You can think of feature flags as extending Continuous Delivery into Continuous Deployment - a way to put changes into production behind a flag and turn them on in a controlled way later (or hide and remove them in the same way).

But you can also think of feature flags as a new way to think about building and releasing applications - by clearly defining features and components that can be changed and toggled on and off individually, you allow for experimentation, controlled rollouts to different users, as well letting more people (like PMs, sales, and support) turn things on and off for customers.

Together, this results in giving more power and control to developers to create more and better features faster, while at the same time ensuring that engineering is not the bottleneck for the business in having to make changes for individual customers or ensuring a perfect production deployment for every new feature or feature update.

If we want to simplify even further, feature flags essentially create private swim lanes for developers where they can ship a feature directly to customers and then have control over who sees it, get feedback, and turn it on and off as needed. It's like a light switch (a pretty complex one)!

Who Uses Feature Flags?

People oftentimes consider feature flags either an engineering tool, or a tool for product managers. The reality is, it's both. Flags can help software development and DevOps teams lower their overhead and increase their velocity, and they can help product managers

better control releases, coordinating launch timings and creating a feedback loop more effectively.

Research indicates that 95% of engineering leaders want to implement a feature flag solution for themselves. This comes on the back of 97% of leaders saying they're under pressure to deliver faster, and 65% of whom say they find it difficult to achieve that velocity increase in a safe manner. It's abundantly clear that engineering teams are likely the biggest beneficiaries of implementing and using a feature flag solution, and indeed, they tend to be the primary users. And when they do implement a feature flag solution, they end up delivering 66% more features per application per year!

Feature flags are great for users across the organization beyond product and engineering. For sales and support teams, feature flags provide a way for them to directly manage betas and new features for their customers, and to track down who's using what in case of any issues. And for management, flags can provide new forms of visibility into what's happening in development and how new features are being tested with users.

On a fun fact kind of note and since we're in the 'who uses feature flags' section, it's cool to note that big companies like Netflix, Flickr, Google, Reddit, and more use feature flags! They're definitely worth looking into and absolutely are an important part of the software delivery lifecycle.

Why You Should Consider Feature Flags

Whether or not the use of feature flags is important to your software delivery lifecycle comes down to what it is that you need to accomplish. While feature flags are a great tool to have for any team, they are particularly useful if you have these problems:

Problems in the Business

- The risk associated with new feature releases ends up slowing down developers since they have to thoroughly validate design and implementation before releasing.
- There are often multiple possible versions of how a new feature will look and function, and it's up to the developer to choose one (not always the right one) and implement it.
- The decision of when to release new software features is controlled by developers instead of the business.
- Sunsetting old features introduces risk to newer features that might have dependencies on the old code.
- It's all or nothing - a feature is rolled out to everyone at the same time instead of doing a feature rollout incrementally for verification purposes.
- When teams are beholden to engineering release cycles (e.g. once a month), multiple features are released simultaneously, introducing complexity and risk that can result in deployment war rooms, large rollbacks, and dissatisfied customers.

Problems in Engineering

- Releasing a feature takes too much coordination and time across stakeholders.
- Merging new features, or managing separate feature branches for changes, is cumbersome and complex.
- You want to test changes to see how they impact your application and end-users before rolling them out more widely.
- You want to provide access to new features, or to test changes with certain portions of the user base sooner - by location, by customer plans, by customers opting into beta testing, or by any other criteria or logic you can think up.
- You want to reduce the risk of rolling out new code by having instant kill switches that don't require rollbacks or new deployments to fix a disrupted environment.

To simplify, we can consider that feature flags are a great solution if you're trying to do the following things:

- Increase engineering velocity and ship more features.
- Decrease feature development risk by A/B testing multiple solutions with customers.
- Reduce the risk and toil associated with production failures and rollbacks.
- Decouple engineering deployments from feature unveilings to customers.

Different Types of Feature Flags/Toggles

It's natural to want to put all feature flag use cases in a single bucket, but it's more helpful to instead view flags themselves as a means to an end and as having categories of use cases.

Here are some examples of the different kinds of flags that could be used in a system:

- **Release features** - Using flags to let the right people decide who to turn a feature on for, and when, without complex engineering deployment coordination required.
- **Experiment** - Using flags to learn something about how a change impacts things. This can be a technical experiment, such as how a new UI impacts server load, or it can be a user experiment such as how a new button impacts conversion.
- **Ops flags** - Letting you have permanent control around key parts of your app so you can turn them on and off as necessary without a full new deployment cycle. Add in RBAC, and this is a robust way to control who has access to modify your production application.
- **Control access** - Using flags as a way to manage betas, early adopter list, trial access, and more.

Implementing Feature Flags

Implementing feature flags can seem really simple at first, but it turns out that like with any system, the devil is in the details. Building an awesome feature flag solution in-house certainly has its perks, but it can be deceptively difficult. In particular, we've seen three

distinct sets of issues arise with feature flag solutions as an organization's implementation matures.

When a feature flag solution is **nonexistent or immature** and an organization is standing it up for the first time, they tend to run into issues with finding value in the tool quickly. Especially with the need for such a solution and the pressure to make it happen quickly, it's critical that value is demonstrated quickly, otherwise it may end up leaving a sour taste. Within that, there's a dichotomy of architecting for eventual scale versus getting an MVP out the door to prove value. This often results in complex design or architecture that isn't scalable and only works for small, targeted use cases.

As feature flag solutions enter a **middle stage of maturity**, or as they **start to scale**, the problem is quite obvious: scaling is hard. Especially if built in-house, ensuring that the tool can be used by multiple users and multiple teams for a variety of use cases, all while maintaining a common set of best practices and building on a single infrastructure can be incredibly daunting. And as this is being figured out and teams have code in production, it can quickly result in stale code or issues keeping production tidy when code changes occur. Add onto that the need to integrate neatly into everyone's workflow and you've got a behemoth of a problem to solve.

Building on that is the scenario we often see with **highly mature** feature flag implementations, where they primarily end up as separate systems from existing software delivery or CI/CD pipelines that teams now have to learn how to use, and to figure out how to port their existing processes or pipelines into the feature flag solution. At this stage, there's the obvious cost of maintenance and future development to keep up with new changes or requirements. Technical debt is bound to build up by this point, and it can become unmanageable without a dedicated team - and perhaps even its own codebase.

Sure, 95% of engineering leaders may want a feature flag implementation, but as it turns out, it's not so easy to build something that works well over time.

Build or Buy: Feature Flag Management

You may be thinking, "Feature flags seem pretty simple, can't we just build this ourselves?"

The real answer? It's complicated.

We've seen in most cases that an internally-built tool will end up causing a lot of overhead and putting a very low ceiling on the benefits you can receive from feature flags. It may be good for your first couple of flags, but it can be hard to make an internal system scalable, performant, and robust for all the things flags can do as your organization gets more and more comfortable using them - and more reliant on them.

For a deeper look at the build vs buy conversation, you can check out an article we wrote earlier this month, Feature Flags: Should I Build or Buy?

Why Use Harness for Feature Flag Management

Harness provides a feature flag platform made for getting you up and running quickly while being able to scale to any level. We make it easy to get started and quickly see the benefits of feature flags. With Harness Feature Flags, that means you can get your first flag out the door in minutes by following three steps:

- Create a free trial account.
- Create a new flag in just 3 clicks.
- Add one of our SDKs to your code in a few minutes.

And, you've got a flag working!

If that's not quite clear, then it's worth knowing that the fundamental way flags work in an environment like Harness is that SDKs, added to your application as a library, will receive updates on flag statuses to evaluate with your users (you should choose between real-time streaming or pulling updates on intervals). These SDKs are secure, collecting no data you don't want to share, and they are designed for performance and stability.

Harness Feature Flags is fast, secure, and focused on helping teams get the most out of feature flags - from your first initial use all the way to global governance, sophisticated reusable templates, and developer-first experiences.

And if you're wondering how we stack up against LaunchDarkly, Optimizely, Cloudbees Feature Management, Split, and other FF tools - those pages are being worked on right now and we'll update this post with the links when they're live!

If you'd like a little more information on our architecture and on how to get started, you can check out the feature flag documentation or request a demo.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/legal/AIDA-Privacy>

AIDA Data Privacy

As of

Unless otherwise agreed by you and Harness, use of the Harness AI Development Assistant (AIDA) is subject to the Harness Privacy Policy, the Acceptable Use Policy and the AIDA Terms.

What data does AIDA collect?

AIDA collects data, information, and other inputs to provide the service, some of which is then saved for further machine learning, analysis, and product improvements. AIDA collects data as described below:

Submission Data

Your submissions are transmitted to AIDA to provide Output to you. Submission data is only transmitted in real-time to return Output, and is discarded after Output is returned. AIDA does not retain submission data.Â

User Engagement Data

When transmitting submissions to AIDA it will collect usage information about events or actions generated from those submissions. These events and actions can include user actions such as output accepted and dismissed, whether the output was helpful or matched what you were looking for, and general usage data to identify metrics like latency, errors and engagement. This information may also be attributed to your Harness account, which will help Harness improve AIDA, provide support, and help troubleshoot issues. Harness will not share your Harness account information with any third party generative AI providers it integrates with.Â

How does AIDA use and share data?

User engagement data is used by Harness and other generative AI providers to provide the service and to enable improvements.

Such uses may include:

- Evaluating AIDA's effectiveness, for example, by measuring the positive impact it has on the user.
- Detecting potential abuse of AIDA or violation of the AIDA Terms or Acceptable Use Policies.
- Conducting experiments and research related to developers and their use of developer tools and services.

How can AIDA users control their data?

When you use AIDA, Harness collects and processes certain user engagement data and general usage data. This data may be shared with third parties as described above. However, Harness will not share your account information with third parties.

Please see the Harness Privacy Policy, for more information on how Harness processes and uses personal data. [Learn more](#)

Effective Date: June 19, 2023

Source URL: <https://www.harness.io/blog/introducing-harness-ci-cd-plugin-for-backstage>

Introducing Harness CI/CD Plugin for Backstage

Today, we are launching the Harness CI/CD plugin for Backstage. The plugin is open source and is one of many more plugins to follow that will integrate Harness with Backstage.

Backstage is an open platform for building developer portals. Powered by a centralized software catalog, Backstage restores order to infrastructure and enables product teams to ship high-quality code quickly without compromising autonomy. Some of the Backstage use cases include:

- Building internal developer portals
- Developer self-service hub for all kinds of automation workflows
- Selection and creation of new software components based on the company's best practices
- Focus on the services they own and see the relevant information like builds, deployments, alerts, and docs all in one place

With the Harness Open Source plugins for Backstage, developers can see the executions of Harness pipelines inside Backstage. In addition, developers can check the latest executions inside Backstage alongside their service's homepage in the software catalog.

The first version of the CI/CD plugin allows you to connect your service on Backstage with a Harness project. The plugin shows the recent executions of the pipelines inside the project. For each execution, you can see the status, details such as commit and time. You can also re-run pipelines in case the execution fails. There is also a way to search across the pipelines using the Filter on top right. If your project has a large number of pipelines, you can specify which pipelines to use for showing the executions on Backstage. You can find more configuration options in the installation instructions on the plugin repository.

This is only the beginning. We will keep shipping more plugins to integrate The Harness Platform with Backstage for use cases associated with feature flagging, managing cloud costs, and chaos engineering. We are also launching the project under Apache 2.0 license, so the open-source community is welcome to contribute at any time. We believe contributions are not only done by committing code, but also by suggesting features and giving honest feedback on how we can improve. All feedback is welcome!

If you use Harness and Backstage, try out our CI/CD plugin now. Head over to the GitHub Repository.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

[Documentation](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[Case studies](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/feature-flag-use-cases?utm_source=pmm-ff&utm_medium=how-we-use-our-own-feature-flags-at-harness

5 Feature Flag Use Cases You May Not Have Thought Of

We talk to folks about feature flags every day, and usually we hear that people have pretty clear ideas about how feature flags can create value for them. However, when we start talking about feature flag use cases and the kinds of ideas we're building Harness Feature Flags to address, a lot of teams tell us that they'd never considered using feature flags for some of these purposes.

So, based on our own experience, we wanted to share some super valuable ways to consider using feature flags that you may not have thought of yet.

1- APIs and BackendÂ

When people think about feature flags, they often think about visible, frontend changes. But, with our server-side SDKs, they can just as easily roll out backend changes with flags.

As one clear example, by using feature flags, you can more easily control migrating backend infrastructure in an experimental way - with less coordination around deployments and rollbacks.Â

To use a logging service as an example, if you were migrating logs from one service to another but wanted to turn it on and off to collect errors and bugfix, by using feature flags you can turn the traffic on and off easily without any need to deploy or roll back in real time as you switch the code paths out.

2- Performance RefactoringÂ

Sometimes, when you build the first version of a feature, the biggest optimization is not around streamlining your queries or optimizing for pure performance. And, sometimes, as a result of this, your frontend engineers are quite insistent that they be given some time to greatly reduce the number of calls the frontend is making to the backend - not that this has ever happened to us, of course. ð

By using feature flags for this kind of refactor, you get two huge new benefits. First, you can easily test it both in prod and up to prod to verify that the intended result is not impacted by your refactor. But, just as importantly, you can also start to quantify the impact of that change by turning it on for a subset of your traffic first, see how your metrics change, then expand it over time. All of this, just like in the above scenario, without re-deploying or rolling back as you go. As far as feature flag use cases go, this is pretty invaluable.

3- Ops Toggles

Ops use cases are not the first things that come to mind with feature flags, but the more you think about it, the more you might see that a significant amount of ops work can be permanently wired up to flags to provide a better UX and a more manageable experience for critical op tasks.

Think about an archiving system that runs 24/7, but occasionally needs to be toggled off during outages caused by upstream issues or high load. Many ops teams have scripts or runbooks for this, or they just know what code to change. Turning this into a feature flag

means teams start to actually build control panels for critical kill switches, and settings that are much easier to use, document, and communicate.

4- ComplianceÂ

Here at Harness, we run different environments for different needs. These needs can be government, European regulation, state regulation, and on-prem. Some of these environments need things modified slightly, or turned off entirely.

Feature flags provide a great solution. Want to turn something on only in Europe to help with GDPR? Flag it and target your European users! Want to turn something off for government or on-premise users? Feature flag management software makes this easy, without the need to maintain and release tons of different versions of your code - or having to layer logic inside your application. Feature flags also make this sort of control explicit. Because it's visible, auditable, and controlled by RBAC, you will keep your security and compliance teams happier and less in the dark.

5- Free Trials

A common request in the sales cycle is that a potential customer would like to try out a feature on a high-level plan, or maybe even try out the entire product without them needing the matching license.

If you've ever been on the engineering side of building these controls out, it can be a huge pain - and very little of it is value-enhancing to the core product experience. Messy and inconsistent internal admin panes begin to sprawl: sales and support teams often are terrified of using these important-feeling systems.

Feature flags let you easily provide control for this sort of account management without the need for complex custom logic or the overhead of custom admin portals. Simply wrap features in flags and let your account teams manage them!Â

Conclusion

As more organizations adopt feature flags, we will begin to see feature flags as more than just software switches to hide or activate features. Feature flags really serve entire organizations in new and powerful ways, simplifying life for engineering and product teams while bringing other stakeholders closer to the product than ever before.

Want to try feature flags and see how it could change your life? Request a demo today - who knows, maybe you'll discover new feature flag use cases to serve your organization!

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

[Documentation](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[Case studies](#)

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

[We want to hear from you](#)

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/harness-ci-and-harness-cd?utm_source=pmm-ff&utm_medium=the-journey-to-using-feature-flags-at-their-full-potential

Harness CI and Harness CD - Your First True CI/CD Pipeline

Harness now allows you to have both best of breed Continuous Integration and Continuous Delivery capabilities. Let's take a look at what leveraging a Harness CI / Drone Pipeline to build then leveraging Harness CD to deploy to a Kubernetes cluster.Â

The goal of this example will be to have Harness CI / Drone build and push a GoLang Docker Image to DockerHub then have Harness CD pickup on the new artifact and deploy to an awaiting Kubernetes cluster. This potentially can take you from idea to production easier than ever.Â

The Moving Pieces

There are not many moving pieces to see an end-to-end example. You need a Harness CI / Drone Instance, a Docker Hub Account, a Harness Account, and of course, a Kubernetes cluster which could even be Minikube.

Like always you can follow along with the blog and/or watch the video.

If leveraging the example Harness CI / Drone GitHub project, you will have to edit the information to be your own Docker Registry for where the Docker Push goes. I will be referencing mine below.Â

The Harness CD Piece

If this is your first time leveraging the Harness Platform, the quickest way to get started is to leverage a Kubernetes Delegate.Â

The first step of getting a Harness Delegate installed in your Kubernetes Cluster is pretty straight forward.Â

Setup -> Harness Delegates -> Download Delegates -> Kubernetes YAML

Give the Delegate a name.

Hit submit to download. Expand the tar.gz which is downloaded.

Inside the delegate folder, run `kubectl apply -f harness-delegate.yaml` to install the delegate.

In a few moments, your Harness Delegate will be available.

Next, add the Kubernetes cluster for Harness to deploy to by adding the Kubernetes cluster as a Cloud Provider.Â Â

Setup -> Cloud Providers + Add Cloud Provider

Add a Kubernetes Cluster

Give the Kubernetes cluster a name then for Cluster Details, Inherit from selected Delegate [deployed Delegate]. Hit Test then Submit and you are all wired up.

Next, creating a Harness Application is a simple process.Â

Setup -> + Add Application.

The next step inside the Application is to create a Harness Service. You can create a Service by going toÂ

Setup -> Captain Canary -> Services + Add Service. The Deployment Type will be Kubernetes.

Inside the Amazing App Service, + Add Artifact Source from a Docker Registry

Can wire to your Docker Image of choice.

Once you hit submit, the scaffolding will be there for the deployment and no need for additional configuration.

With the Service out of the way, we can wire a Harness Environment to deploy to.Â

Setup -> Captain Canary -> Environments + Add Environment

Once you hit Submit, next can wire an Infrastructure Definition.

Next click + Add Infrastructure and add the Kubernetes cluster.Â The Cloud Provider Type and Deployment Type will be Kubernetes.

Once you hit Submit, can wire together a Harness Workflow to define the steps to deploy.

Setup -> Captain Canary -> Workflows + Add Workflow. The Workflow Type will be Rolling Deployment. Select the Environment, Service, and Infrastructure Definition that was created in the previous steps.

Next you can create a Harness Trigger on the presence of a new artifact in DockerHub. This will trigger the Workflow once the Docker Push is complete in the Harness CI / Drone Pipeline.Â

Setup -> Captain Canary -> Triggers + Add Trigger

Hit Next after giving a Name. DockerHub can send out a webhook or we can use a polling interval from Harness to look out for a new artifact. Without the need to manage webhooks, we can just simply configure On New Artifact. Select your artifact source and can filter on specific tags. For the example we just want to deploy what gets picked up and â.*â is fine as a filter.

Hit Next and define the Actions. Execution Type will be Workflow and will execute the Workflow that was created in the previous steps. Can leverage the Last Collected Artifact. If leveraging the example will pick up on the newest explicit tag with the same filter â.*â.

Click Next and review the Trigger then hit Submit.

Once you hit Submit, all you have to do is kick off a Harness CI / Drone build and push.Â

Harness CI / Drone Steps

If you do not have a donât have a running Harness CI / Drone instance donât worry, not difficult to accomplish. We have a detailed blog and video to get you started. For the example to work will need to be building and pushing a Docker Artifact. The example project has a simple GoLang application. The Harness CI / Drone introduction blog and video go through wiring Harness CI / Drone to a GitHub repository.Â

To kick off a Harness CI / Drone Build and Push, simply modify the configuration / Drone.yaml which Harness CI / Drone is monitoring for. To explicitly build to a specific version, can add tags to Drone.yaml which works well with the example.Â

In the example, pushing build version 1.0.2. Make sure to update your repository information also.

Can modify/increment then commit.

Once you hit Commit, Harness CI / Drone will take over.

Once the publish step is finished, a new image will be in DockerHub.

Watch the Magic

Within the polling interval, Harness will pick up on the newest build/tag and start deploying on your behalf.

With the Deployment kicked off, you have an end-to-end CI/CD pipeline. From code to production!

Partner with Harness in your CI/CD Journey

No matter where you are in your CI/CD journey, Harness can simplify and scale your capabilities. With the lightweight and convention-based Harness CI / Drone and the power of the Harness Platform, achieving CI/CD nirvana has never been easier.Â Get started with Harness CI / Drone and sign up for a Harness Account today!

Cheers,

-RaviÂ

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/harness-cloud-asset-governance-powered-by-aida-the-path-to-a-well-managed-cloud?utm_source=pmm-ff&utm_medium=safe-testing-in-production-with-harness-feature-flags

Harness Cloud Asset Governance powered by AIDA®: The Path to a Well Managed Cloud

We've all gotten busy and let our refrigerators get a bit messy over time with forgotten and expired food which takes up space, wastes money, and could lead to health risks. And that's with only one or two people using it. Imagine how bad the problem would be with dozens of people sharing it, losing track of what is theirs, and no one taking responsibility to clean out the science experiments hatching there. Scary right?

But that is exactly what happens in your cloud environment. The ability to self provision infrastructure leads to great agility in development, at the cost of keeping everything neat and orderly. Overtime, without someone chartered to take charge of the mess, your cloud becomes cluttered with unused resources, inefficient deployments and compliance and security issues, which together waste a large part of the \$490 billion cloud computing market spend.

Wasted spend means lost opportunities for innovation. In the current economy, efficiency is paramount as tightening budgets force teams to take a hard look at their cloud efficiency. Trying to manage your cloud assets by relying on ad-hoc scripting and manual processes leads to a massive overhead for developers, and lacks centralized reporting on the how well your cloud is being managed.Â

You need a better alternative, one that automates the governance of your cloud assets, while also giving your teams strong visibility into their cloud spend and efficiency.

Introducing Harness Cloud Asset Governance

Harness Cloud Asset Governance helps customers find and eliminate cloud waste automatically, while also providing a policy engine to ensure cloud resources are in compliance with corporate standards. By leveraging policy-as-code, it automates resource optimization, security, and compliance tasks, freeing your engineers to focus on creating innovative products and services that drive your revenue.

Harness Cloud Asset Governance is built on top of the popular open source software Cloud Custodian, designed to streamline cloud asset management and governance across your multi-cloud environments. It's a widely used open-source tool backed by CNCF, that enables users to modernize cloud governance with a *governance-as-code* approach that simplifies and automates setting up the guardrails needed to proactively manage your cloud assets.

However, using Cloud Custodian at scale comes with challenges:

- **No GUI:** It's a CLI driven tool only, requiring knowledge of how to create and edit YAML files in the correct syntax.
- **No Reporting:** Without a GUI, there's not way to provide for centralized visibility of rules and enforcement across stakeholders
- **No Security/Governance:** There is no ability to apply RBAC, or have audit trails for changes made.Â
- **Operational Overhead:** As with any open source tool, it requires ongoing maintenance, high management overhead, and needs dedicated infrastructure provisioned
- **No Generative AI:** No ability to provide guidance on rules, or smart policy authoring that allows user to create rules using natural language

Harness Cloud Asset Governance leverages all of the goodness of Cloud Custodian, such as its comprehensive coverage of governance policy support across cloud providers, while eliminating all the major pain points of self-hosting Cloud Custodian. Harness provides a rich set of preconfigured governance-as-code rules that make it easy to implement out of the box, as well as introducing AI Development Assistant (AIDA®) to power Cloud Asset Governance with a natural language interface that eliminates the need to understand YAML

syntax to author policies.

Comprehensive Cloud Resource Coverage

Since we built Cloud Asset Governance on top of the open source Cloud Custodian, we can take advantage of all of the comprehensive coverage that currently exists, and new coverage as it's created. That allows us to support all major cloud assets, across all major cloud providers. Support for AWS is already generally available, and we've just launched the Azure beta availability, with GCP soon to follow.Â

Harness also offers a wide range of policies which are available out of the box, which you can leverage on day 0 to optimize your cloud resources and setup guardrails against future wastage.

AWS Resource Coverage (Comprehensive list)

- EC2 instances
- S3 buckets
- Lambda functions
- RDS (Relational Database Service) instances
- CloudFormation stacks

Azure Resource Coverage (Comprehensive list)

- Virtual Machines (VMs)
- Storage accounts
- App services
- Cosmos DB accounts
- Key Vaults

GCP Resource Coverage (Comprehensive list)

- Compute Engine instances
- Cloud Storage buckets
- App Engine applications
- Cloud SQL instances
- Cloud IAM policies

Our customers have seen immediate results from their use of Cloud Asset Governance. Jay Patel, Head of DevOps at Advanced, had this to say about it:

How Does Harness Cloud Asset Governance Work?

The possible combinations of governance-as-code rules that you can implement is nearly endless. You can keep it simple, and just look for orphaned, unused and idle resources, and shut them down. You can focus on governance and compliance, and write rules to enforce data retention policies, or ensure new assets are properly tagged. Or, you could focus on right-sizing assets such as compute, database or storage, as in the example below.Â

For some companies, data storage costs can be a significant portion of their cloud bill, especially if engineers overestimate the I/O speeds they require for their applications. Let's take Elastic Block Storage (EBS) volumes as an example. As new generations of SSDs are introduced, they become faster and less expensive than previous generations, and AWS passes these benefits to their customers with new EBS volume types.Â

EBS gp3 volumes are up to 20% less expensive than the older gp2 volume type, while also allowing independent provisioning of volume size, IOPS, and throughput. It's almost a no brainer to upgrade to gp3 as they are cheaper and offer better throughput than gp2.Â

But, how can you enforce this as a guardrail, and ensure existing and new volumes are all using less expensive, faster storage? With Harness Cloud Asset Governance, you can easily create a policy for this, but you won't have to because this policy is available out-of-the-box.Â

You simply need to pick the accounts & regions where you want to enforce the governance policy. Harness Cloud Asset Governance will automatically identify all the EBS gp2 volumes which can be auto upgraded to EBS gp3 and perform the upgrade. **All with a single click!**

If your teams are like most engineering teams, they're going to be a bit hesitant to allow rules to be implemented without at least a little bit of oversight. At least at the start. That's why Harness Cloud Asset Governance includes a "Dry Run" feature that lets you see what the rule would do, without enforcing the actions. Harness Cloud Asset Governance also keeps comprehensive logs of the actions that have been taken, so your teams have an audit trail if questions arise.

You can also set up Enforcement rules which will periodically scan through your cloud asset inventory to enforce policies automatically over time to keep your cloud assets well managed.Â

Harness AIDA® (AI Development Assistant)

Cloud Asset Governance policies play a crucial role in automatically governing cloud assets and proactively optimizing cloud costs. However, authoring these governance-as-code policies can be challenging and confusing, especially when trying to master the correct syntax for your needs.Â

The Harness AI Development Assistant (AIDA) was created to assist with understanding your existing policies, as well as creating new policies. Harness AIDA offers a user-friendly, natural language interface that serves as an excellent starting point for establishing the necessary policies for your cloud governance.Â

When you need to create a new governance policy rule, all you need to do is type in what you need, using a sentence format, such as âfind unused EBS volumes older than 90 days and delete themâ, or using our example above âfind all EBS gp2 volumes and upgrade them to EBS gp3â. Harness AIDA understands your requirements and generates customized policy rules to align with your needs.Â

How do you know that Harness AIDA created a valid rule (or that you even asked for a valid action)? Harness Cloud Asset Governance can validate the rule for you, before you ever try to implement it.

What about if you have a rule that was pre-defined, but you donât fully understand what it is going to do? Harness AIDA offers detailed descriptions of built-in rules and custom rules that others in the organization might have authored. This feature enables you to understand the purpose, scope, and implications of each rule, thereby facilitating informed decision-making during the policy enforcement process.

With AI becoming a critical capability in so much of our daily lives, itâs no surprise that our customers are taking full advantage of Harness Cloud Asset Governance, powered by AIDA. Michal Malohlava, VP of Engineering at H2O.ai said:Â

Whatâs Next?

Transform your path to a well managed cloud with Governance-as-code and try Harness Cloud Asset Governance now to receive automatic recommendations that can save you money, improve compliance, and reduce security risks. Still want to learn more? Talk to a Harness sales specialist today to get a free demo.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

6 Use Cases for Kill Switches in Production Using Feature Flags

If you've only come to feature flags recently, you may not know that feature flags as a concept has existed for quite a long time, and can easily be seen as an evolution of app config systems. But first things first - if you need a refresher on feature flags, head on over to our "What Are Feature Flags?" article now.

One of the least used - but most powerful - use cases of kill switches using feature flags can be for Ops and SRE teams. Plenty of folks never think about this potential, but it can be a way in which feature flags can most transform your engineering organization. They're a great way to increase velocity and reduce risk in software development.

Kill switches create tremendous potential both to control granular capabilities or features within an application, and also as long-lived operational switches that affect the entire application. And while feature flags are intuitively thought of as affecting frontend or client-facing applications, they can also be used just as well on the backend. Let's explore how you can leverage kill switches to change the nature of how your team can deliver software.

Use Case 1: Control in Production With Feature Toggles

Consider this the base case of implementing kill switches using feature flags. The concept is simple enough: you can rig any and all features and changes pushed to production (or any environment) with a toggle, or a kill switch. Feature flags, by nature, allow teams to isolate features within their applications so that they can be handled individually instead of as a deployment bundle. Kill switches themselves are the way in which teams can create a control to disable code in production at any time.

This basic kill switch functionality gives teams the ability to turn off a feature when it's not ready. "Not ready" can mean two things in prod:

Unfinished Features

The first condition for not being ready is what slows down many teams during the deployment process. In particular, as many teams work on different features across different applications, integration and deployment take a long time. Without having everything rolled in, release teams and the software development teams that built them both find themselves slowed down.

Using feature flags as kill switches in this context allows teams to push new features to production even as incomplete features, which can result in faster development cycles. It simply needs to be pushed live, but with the kill switch activated (feature turned off) so that it's not actually impacting anything.

Broken Features

Here's a developer favorite: it worked in all the pre-production environments, and then a bug was discovered that's now impacting production. Without the use of a feature flag, that broken feature impacting prod would now necessitate a complete rollback of the deployment - that's right, all of the good features have to be pulled off the shelves too and production is switched back to the last working version.

On the other hand, rigging each new feature or change with a feature flag enables teams to hit the kill switch as soon as an issue is found. Now, all of the features that work get to stay in prod, while the broken features are isolated out and handled as required.

Use Case 2: Incident Management

What happens when a severe issue occurs in production because of feature releases that are broken? Or when some infrastructure failure cascades across all of production? As it turns out, kill switches can be used to mitigate these incidents and improve the MTTR (mean time to resolution) for the teams responsible. Talk about stress management.

Scenario 1

In scenario 1, a broken new feature is causing an incident in production. The good news is, if that new feature was rigged up to a feature flag, the kill switch could be triggered and that feature could be turned off in production right away, resulting in minimal impact. Not only is the feature no longer affecting production, but there is also suddenly no massive response team that needs to be assembled to handle the issue, which would typically mean the whole deployment had to be pulled. Instead, the feature is turned off, the team responsible for the feature is on the hook to fix it, and the one feature fix can be rolled forward when it's ready.

Scenario 2

In scenario 2, an infrastructure failure is one example of what could happen in production. However, production outages can happen due to a variety of issues. Oftentimes, Ops teams will have fallback options or adjust turn it all off runbooks in place, and kill switches can be used here as well. If it's a new infrastructure change entirely, that whole change can be wrapped in a feature flag and a kill switch can be triggered that will failover to the old setup when an issue occurs. Some teams also create long-lived operational flags that can, say, put a whole site or application in maintenance mode, effectively triggering a kill switch for the whole application. This can be useful in P0 scenarios where things are completely broken and the preference is to not allow any access.

Use Case 3: Application Load Management

Let's say your business is an e-commerce company. During the holidays, you experience high load relative to the rest of the year. Now more than ever, the business doesn't want to have downtime or have customers experience issues! Turns out feature flagging is an incredibly low-cost way to solve this problem.

One approach to solve this problem could be to simply turn off some features that will reduce the load, and this is a great place to use a kill switch, but it might not be the most effective. Another solution might be to kill certain load balancers or servers that are useful in non-peak times, and cut over to infrastructure that's designed specifically for high load.Â

Real World Example

At a previous job, we had log data that would need to go into storage from the short term cache, so we had a system to collect and store them. During peak loads or other periods of instability (particularly upstream with AWS), we would often see this storage become either very expensive or very unreliable. So, we would turn this service off during these periods, but this was manual and involved a series of AWS commands and database changes to both disable and re-enable later on. By wiring this up to a feature flag, we were able to turn this feature off and on instantly in a much more lightweight, visible, and easy to govern way.

Use Case 4: Testing in Production

One of the most common use cases for teams doing feature flagging is testing in production. In short, this is where a specific feature (or set of features) needs to be tested against real production data. After all, pre-prod environments can only test so many things! There are two use cases that we see here:

The use of feature flags as kill switches here is a pretty simple one: have it live until the test has been run, and then turn it off. It's also possible that a feature will fail during testing and need to be turned off to minimize impact. You can dig deeper into testing in production in this dedicated blog.

Use Case 5: Progressive Delivery

Here, let's assume a basic knowledge of progressive delivery. It's becoming an increasingly popular feature release methodology, very similar to the methodology used for canary releases in Continuous Delivery.

With progressive delivery, the need for a kill switch changes slightly. Yes, the main use case is still turning things off when they break or aren't needed, but here, we have to layer in automation. By its nature, progressive delivery is something that teams run in an automated fashion. To do that, they need to pre-define in which scenarios flags are turned on, for whom, and under what conditions more users get access. Conversely, they must define failure criteria in which the kill switch is triggered and new features are turned off, or the user base is shrunk.

In Harness Feature Flags, we automate progressive delivery via the Pipeline. Teams can schedule releases, mandate approvals, integrate with plugins, create trigger events, and template rollouts - and then automate it.

Use Case 6: Personalization

Personalization is a hot topic. Kill switches using feature flags are of huge value in being able to do this well at scale. We can look at three distinct scenarios in which you can use kill switches to improve your ability to personalize experiences for customers.

Compliance & Regulation

When it comes to regulation, it's a non-negotiable item to be able to personalize something like a mobile app release to comply with the laws in various countries, or to meet the needs of clients in specific verticals (e.g. government, finance). The most common of these is data privacy - specifically, complying with GDPR.

Considering this, it can be a mess showing a button in one place but not another. Wiring key features up as permanent feature flags allows you to easily turn something on for all your North American users, but not your European users where GDPR compliance is mandatory. It's not just creating a kill switch that given context will turn off data collection in Europe, it's also being able to kill data collection for any user that decides to opt out. It's much easier than having a configuration file that tries to solve for all of the various scenarios.

User-Defined Personalization & Admin Settings

Think of this as giving control to the user on what features they want to kill. This isn't dissimilar to giving them access to admin settings on the app so they can choose to have dark mode or choose what notifications they get.

While on the frontend, users will define what they want and don't for their personal experience on the app, on the backend, it's just exposing parts of the feature flag schema to the end user and letting them make the decision on what to keep and what to kill.

Another consideration here is the use of features like screen time or parental controls, where a kill switch can be triggered to limit the access of users to specific applications altogether based on settings elsewhere on their devices. Of course, putting the onus on the user to define what they will do with such power is a whole other problem to solve.

Using Harness Feature Flags to Implement Kill Switches

Hereâs the plug: using Harness Feature Flags gives you the ability to implement all of these use cases right out of the box. When you set up feature flags in Harness, youâre able to work either in a visual UI or entirely in code to manage your kill switches. You also give yourself peace of mind when you eventually want to scale your usage of kill switches or other feature flags use cases with built-in governance, compliance, and security considerations, as well as the critical integration into CI/CD (Continuous Integration and Continuous Delivery).Â

After all, you donât want to set up a great feature flagging system that is entirely disconnected from the rest of your software delivery process. It doubles your work on access control, security, governance, and integration maintenance. You end up getting diminishing returns on your feature flagging platform, which is supposed to *accelerate* software delivery.

If you want to see how Harness can fit into your organization, you can sign up for free forever, or contact us for a personalized product demo.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/progressive-delivery?utm_source=pmm-ff&utm_medium=safe-testing-in-production-with-harness-feature-flags

Progressive Delivery: Everything You Need to Know

Progressive delivery is a practice that builds on the capabilities of Continuous Integration (CI) and Continuous Delivery (CD) to help you deliver with control. Software delivery helps organizations and teams get their changes out to customers quickly. Often, what gets forgotten is discussing how we ensure quality and reduce the risks of introducing said changes to customers. Delivering changes rapidly can often be at odds with not breaking things. Â This blog post will share what you need to know about progressive delivery as we share the ways organizations are using it today.

What is Progressive Delivery?

Progressive delivery is a practice that allows organizations to control how and when new software features or changes are delivered. It builds on the capabilities and practices of feature flag management and deployment strategies like blue-green and canary deployments. Ultimately, progressive delivery combines software development and delivery practices allowing organizations to deliver with control.

When discussing the software development life cycle (SDLC), we often mention continuous integration (CI) and continuous delivery (CD), which captures the how, when, where, and why of our software delivery. CI/CD is integral to quickly, safely, and repeatably we can get our software changes to our users and customers. Progressive delivery leverages your CI/CD pipelines alongside our software development and delivery practices to boost our software delivery capabilities.

The Key Elements of Progressive Delivery

There are several key components to achieving progressive delivery.

Source Code Management and Git Branching Strategy

Source code management is a practice that allows for the reverting, tracking, and correction of software code. A key element of progressive delivery includes application development code that is properly managed and tracked by a version control system like Git. In a basic Git-based workflow, you have the main branch from the main branch, which contains different versions of your application's code, branch from specific versions of the application code when working on a different feature. This allows for development progress to continue without the fear of losing changes as developers work on the same codebase.

Having source management is a key element for progressive delivery as it serves as the foundation for feature development, testing, and rollout.

Continuous Delivery, Continuous Integration, and Feature Flags

CI/CD is a shortened term for Continuous Integration and Continuous Delivery. CI/CD is the combination of principles, practices, and capabilities that allow for software changes of all kinds to get users in a quick, repeatable, and safe manner. This allows for software developers to integrate their feature or service changes continuously and for IT and operations teams to deliver with the standards, security, and confidence businesses need. CI/CD also serves as a foundation for your progressive delivery capabilities. Automated testing ensures that these integrations and deliveries maintain the desired quality.

You can learn more about CI/CD in this blog post.

Feature Flag Management

As a development codebase grows, so often does the development team. With new feature requests, developers and innovation syncing and aligning to a development pace can be difficult, especially for complex feature work. There are many challenges that long-running feature work face when it comes to delivering fast. Merge conflicts arise for long-lived development branches, and often, features require manual testing. Feature flags, also known as feature toggles, control the visibility of features behind a logic (often Boolean) flag. This allows teams to avoid redeploying an application to enable a feature for testing or production.

Feature flag management also has many benefits to development workflows. Contributors to a codebase don't have to wait for a feature to be fully finished to merge into the development or main branch. Developers can also test their features by controlling their feature flag and exposing their feature in a running application in a test or development environment.

A/B Testing

A/B testing, also known as split testing, is a technique that involves exposing two variants of the same application service to different segments of users. By shifting specific over to different versions of an application, teams can compare, experiment, and better understand how these changes impact a specific set of users. To effectively conduct A/B testing, it's essential to know how to create a flag that can toggle between the two variants.

Feature Release Strategies

Also known as feature rollout, these deployment or release strategies are used to change or upgrade a running instance of an application. There are three kinds of deployment strategies, rolling deployments, blue-green deployments, and canary deployments. Of these three strategies, two are progressive since they involve progressively shifting user traffic to a running service.

The Blue-Green deployment is a deployment strategy that gradually transfers user traffic from a previous version(green) of an application or service to a new release(blue) of the application or service.

The Canary Deployment is a deployment strategy that releases an application or service to a subset of users.

Both blue-green and canary deployments limit the impact of application changes. These capabilities can be beneficial if deployments are risky and have severe consequences to the user base in breaking changes.

Observability

Observability leverages metrics, tracing, and logging capabilities to provide organizations and teams with answers to questions regarding their running services without the need to deliver another instance of the application or service to answer those questions. Understanding your services interact or perform can give developers a better understanding of what is breaking and what is performant. Often we need to understand an application's performance before we can achieve the confidence needed to release a change to an entire user base, and this is where observability matters.

Implementing Progressive Delivery

The best approach to implementing progressive delivery is to build the key elements of progressive delivery incrementally. A common practice is to invest in your development workflows. This can involve introducing feature flag management capabilities through an SDK or by building it a home-grown solution. Some other organizations may need to introduce source code management practices first before even discussing A/B testing capabilities or feature flag management. If you're starting with software development and would like to read more about git-based workflows and how development and feature branching works, take a look at this source-code management guide.

Some organizations use feature flag management within their development processes already but need more capabilities around verifying deployments. It may make sense to invest efforts into monitoring or observability solutions to measure and understand application services. You can't compare or baseline performance without these capabilities. And changes go out with confidence, so this can often be the next step in your progressive delivery journey.

And lastly, there is a deployment aspect to progressive delivery. Your platforms should support deployment strategies like canary or blue-green deployments. Deployment capabilities can come from your continuous delivery platform or other solutions like a service mesh. If you're looking for a complete solution to deploy your services to production safely, I recommend looking at Harness's CI/CD platform.

Improve your Software Delivery with Harness

Today software delivery isn't just about speed. Progressive delivery joins software development and delivery practices to better support and control your software delivery. If you're looking to move fast without breaking things and implementing any progressive delivery element quickly, I recommend trying Harness for free.

For more on progressive delivery, let's look at Feature Flags!

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/targeted-rollouts-release-progression?utm_source=pmm-ff&utm_medium=the-journey-to-using-feature-flags-at-their-full-potential

Targeted Rollouts and Release Progression

When you want to try a new Harness feature out, we enable it for you via a feature flag. Our support/account team turns it on for you.Â Typically, for new features, our release process happens in multiple phases. First, we test internally in lower environments. Second, we test internally in production. Third, we test with a few beta customers in production for a period of time leading to a wider release. Finally, for some features, we release more widely - but still in beta, and often still with some restrictions on who can access them. This is all an example of using targeted rollouts via feature flags, and then progressing releases over time as you learn and respond. We will dig into that further in this blog.

Why Would I Want to Do That?

Feature flags are often thought of as being on the continuum of CI/CD, and thereâs good reason for that. Feature flags de-risk your deployments, let you merge and ship faster, and reduce the need for long-running feature branches. In this way, feature flags level off the classic goals and outcomes of CI/CD.

However, feature flags are not only an extension of CI/CD. They are also a separate process that sits *alongside* CI/CD, enabling new behaviors or *removing* those behaviors from the CI/CD loop entirely. This simplifies CI/CD and also provides you and your customers with new and better release options.Â

When thinking of feature flags this way (as a process that runs alongside CI/CD rather than one that sits on top of it), targeted releases and release progressions are high-value behaviors that immediately emerge.

Targeted releases, specifically, mean turning your change on for a specific subset of your audience initially, and then progressing it to more users if/when you're ready. Or, alternatively, turning it on for this subset and then turning it off right away if there are issues.Â This method is especially useful for reducing the amount of project management needed, where you might want to incrementally introduce changes and measure their effects with very low overhead.

Using feature flags like this allows you to stress test changes, have PMs or other folks drive alpha and beta feature feedback, control releases by customer level or region, and work closely with design partners. All without needing to rely on engineering day-to-day for a deploy/rollback cycle. Or to enable/disable things for the targeted users.

How Is it Different Than a Canary Deployment?

We previously wrote about how features flags and canaries should be thought of as working together, not an either/or. But in the context of targeted releases, itâs worth revisiting the differences between feature flags and canaries.

Canaries are a fantastic way to *test the code or the artifact* for system anomalies, performance issues, scale, and other similar metrics - and roll back as needed. However, feature flags can help you *test the change, not just the code*.

What does *testing the change and not just the code* mean, specifically? It means that the code working as intended doesnât mean the feature is working for users. Does it do what we need for our customers? Can they find it? Can they understand it? What questions do they have? Avoiding a type 2 error---where you mistakenly accept something as fine when it's not---is key.

If we're in beta of a feature and we're trying to prioritize the next batch of work in the most useful order, having this kind of feedback guiding what we do next and what has the highest impact is critical. Itâs not about âDoes the code technically work?âÂ

This is an example of where we want to think about feature flags not as an extension of CI/CD, where the point is simply releasing the change without risk, but rather as a standalone part of our software development process where we are constantly enabling and disabling things to get feedback, learn, and respond while we are building and deploying every day.

Internal Examples

Iâve already talked a little bit about how we use Harness Feature Flags internally, to turn things on for our customers. Thatâs one example of how we use targeted rollouts here at Harness. Though, it is one that often makes Slack messages difficult to parse since sometimes we use Harness Feature Flags to turn on a feature inside of Harness Feature Flags for Harness Feature Flag customers (whew!).

There are some other great ways we use targeted releases, though.

- We may turn a feature off for all on-prem customers, but not have to worry about maintaining more complex conditional code or separate builds.
- We can turn something on for our Enterprise users without yet having it wired to the account framework that enforces plan limits, which lets us learn more about the feature earlier in the process.
- If a feature is early in development, we may give access selectively to our sales engineering team so that they can work with partner customers in a hands-on way to demo the feature, get feedback, or drive conversation around where we are going.

And thatâs just how we use targeted rollouts. Weâve heard of plenty of other teams that turn things on and off per region, to test performance or scale. Weâve also heard of customers turning things on and off for specific infrastructure or clusters, to help with migrations or to get data about possible configurations. Thereâs a lot you can do that isnât immediately obvious.Â

Conclusion

Using feature flags for targeted rollouts is a powerful way to expand how your organization builds software, learns from customers, and

de-risks changes along the way.

Most organizations, when starting with feature flags, have the idea in mind that feature flags require the right use cases to be useful. They often struggle to know where to start. However, thinking of targeted releases helps explain that the more things are behind flags by default, the more options you'll have in the future.

You can't always predict what, when, and where you will want to target. The mindset of targeted releases is about knowing you have the capability to do so, so that as scenarios, questions or needs come up, your organization is able to learn and react.

Want to keep reading about feature flags? Why not check out our pieces on Kill Switches and How To Get Started With Feature Flags?

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/harness-policy-as-code?utm_source=pmm-ff&utm_medium=the-journey-to-using-feature-flags-at-their-full-potential

Introducing Harness Policy as Code, Powered by OPA

We're excited to announce Harness Policy as Code, powered by Open Policy Agent (OPA), a centralized policy management and rules service that empowers enterprises to centrally define and monitor policies that are enforced across all delivery pipelines and processes. Harness Policy as Code helps organizations create and enforce policies on deployments, infrastructure, and more, providing developer velocity without sacrificing compliance and standards.

Harness Policy as Code is based on OPA, an easy-to-use, extensible solution for creating and enforcing policies across the entire stack. OPA is an open source project accepted by the Cloud Native Computing Foundation (CNCF) with wide adoption across numerous software delivery use cases. Policies are written as declarative code, so they are easy to understand and modify—from simple to complex use cases.

Harness Policy as Code integrates with CI, CD, and Feature Flags enforcing automated approvals, denials, and other advanced pipeline functionality. Check out our technical documentation to learn more.

Why We Need Policy Management in Software Delivery

As DevOps is adopted within an enterprise, typically one team creates and maintains software delivery and processes. That team has full control and visibility, as they are the creators of DevOps processes within the company. As more business units adopt DevOps within the company, that originating team's manual processes can create a bottleneck, which hampers innovation by limiting team autonomy and slowing down software delivery.

Â In an effort to remove the bottleneck and increase velocity, companies can give development teams more autonomy by allowing them to drive their own DevOps processes. That decentralization of process control can lead to more risks for the company.

When governance is decentralized, development teams can miss quality checks or approvals, introduce vulnerabilities, or break compliance. Organizations need to balance autonomy *and* governance, so they can empower teams with the confidence that they are adhering to all compliance standards and security policies â all without slowing down innovation.

Compliance becomes even more critical in regulated industries, such as financial services and healthcareânot only with enterprise standards, but with third-party regulations, like SOC2, PCI, and FedRamp. It is imperative that all software delivery pipelines meet compliance standards with full auditability; otherwise, the organization is at risk of failed audits, heavy fines, and reputational damage.Â

Centralized management and governance of policies across DevOps processes allow enterprises to define standards for the entire organization while enforcing compliance with regulations. Policies enable individual teams to have autonomy over their processes with oversight and guardrails in place to prevent them from straying from standards, ensuring secure and compliant software delivery.Â

Harness Policy as Code Features

Harness Policy as Code is a centralized policy management and rules service that leverages OPA to meet compliance requirements across software delivery. HPE enables organizations to centrally define and monitor policies that are enforced across all delivery pipelines and processes.Â

Policy as Code features for writing and enforcing policies include:

- A Policy Editor that enables developers to start writing policies-as-code quickly. With a library of policies to start from and a testing terminal, developers can try out policies on real inputs during development before enabling them.
- Policies that are configured to be automatically enforced on Harness processes (e.g. on Pipeline Run, on Feature Flag save).
- The ability to set severity, so a policy violation can issue a warning or throw an error to stop processes from continuing.
- An audit trail that can maintain a full history of policy evaluations with detailed outputs for audit and compliance.

With the release of Policy as Code, policies can now be enforced on CI and CD pipelines and Feature Flags.

Pipeline policies govern the requirements of delivery pipelines, and they can be automatically enforced when the pipeline is saved or triggered, or even in the middle of pipeline execution. Policies can enforce specific pipeline configuration, advanced access control use cases, runtime validation, and more. Here are some examples of what the Policy as Code can do:Â

- Require an approval step before deployment to production.
- Forbid use of Shell scripts in the pipeline.
- Only allow deployment to approved namespace.
- Only allow deployments from approved container registries.
- Validate test step outcome meets minimum threshold before allowing the pipeline to continue.

Policies for Feature Flags are enforced when the flag is updated or toggled on/off, enabling policies for adhering to standards, flag process, and hygiene. This includes:

- Only allowing creation of boolean flags.
- Enforcing flag naming conventions.
- Enforcing when creating a flag the default on and off values must both be false.
- Requiring a Feature Flag be enabled in QA before it can be turned on in Production.

Policy as Code centralizes and standardizes policy management across software delivery, allowing engineering leaders to empower dev teams to own their tools and practices while ensuring that everyone is following company standards for compliance and security. With guardrails in place, security vulnerabilities won't be introduced as development teams are writing their pipelines. Leaders can rest assured that compliance standards are being met, with full auditability of policies and failures, and they can find and report breaches as early as possible with shift-left governance.

Get Started

Check out our platform governance page to learn more about how Harness' modern approach to software delivery governance empowers teams with stable processes that don't slow down delivery, or request your personalized demo today.

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/intelligent-cloud-cost-management?utm_source=pmm-ff&utm_medium=safe-testing-in-production-with-harness-feature-flags

Introducing Intelligent Cloud Cost Management

Harness Cloud Cost Management now delivers intelligent cloud cost management, providing complete cost transparency across engineering and finance with robust business intelligence (BI), and the ability to reduce wasted cloud costs by up to 75% with Intelligent Cloud AutoStopping and ML-based recommendations.

Not A Novel Problem - A Critical One

For years, it's been common knowledge that organizations regularly overspend on cloud by about 35%. While the problem of managing cloud costs isn't itself new, therein lies the problem - the problem isn't new, but the solutions to the problem may come off as stagnant.

Tooling vs. Culture

Let's talk about the state of tooling for a moment. While there are a number of good solutions out there, if we look at how cloud cost management practices are maturing, they often fall behind the curve. Indeed, tooling is not changing as quickly as cloud cost management practices! The next frontier is to make it more boring or as automated and intelligent as possible to alleviate as many of the tasks as possible. This will lead to cloud cost savings.

In addition to having the right tooling in place, enabling cultural change within an organization is a critical component of any cloud cost management solution. In 2019, the FinOps Foundation was founded, arising from repeated customer need for an open source group that supported the growing practice of FinOps, or Cloud Financial Operations. FinOps has rapidly grown as a culture, a job title, and as a methodology, not unlike the growth of DevOps. In under 3 years, the FinOps Foundation now represents over 1,500 companies, demonstrating the shift to repeatable and scalable cloud financial management as a core business practice.

Where as few as five years ago cloud cost management was largely a top-down function stemming from finance teams, it has grown to include engineering owners as well. The FinOps role itself is a cross-functional one between finance and engineering teams. And because engineers are the driver of costs, it's the case that companies are pushing more of the responsibility to manage costs on them. But this is one more responsibility to add onto their long laundry list, and so simplifying bottom-up cloud cost management becomes important to enable engineers to be effective.

This graphic outlines the top challenges for FinOps practitioners today:

Old Tools Don't Solve New Problems

To solve these problems today, companies are trying to leverage their existing cloud cost management solutions, which were primarily built to support finance and financial management/FinOps teams.

These tools generally were not architected to work in an engineering context, and can make it difficult for engineers to successfully take on responsibility for managing costs. This leaves both engineers and FinOps practitioners having to rely on multiple tools and methodologies to effectively manage their cloud costs, which creates inefficiencies and can leave significant cost savings on the table.

Achieving Cloud Cost Optimization in Organizations

Across an organization, these things are important with regards to cloud cost management:

- Achieving predictability in cloud costs.
- Top-to-bottom transparency on cloud costs, providing visibility to engineers, budget owners, financial management, and executives.
- Accurately allocating cloud costs to appropriate cost centers and budget owners.
- Creating governance and driving cultural change around cloud cost management best practices.
- Automated guardrails to minimize unexpected cost spikes and budget overruns.
- Maintaining highly performant applications and services while being cost-efficient.

The Solution

While existing tools may serve some purposes well, a modern and intelligent cloud cost management solution needs to solve three key use cases in order to address the outlined problems.

Let's take a look at how Harness Cloud Cost Management (CCM) tackles the use cases of transparency, optimization, and governance.

Cost Transparency

From the CFO to the engineer, CCM provides contextually relevant visibility into cloud costs so that they can be understood at any level of the organization, from high-level business intelligence to the individual resource level.

Costs across AWS, GCP, Azure, and Kubernetes (K8s) are all easily accessible, and using Cloud Cost Business Intelligence the data can be presented for any necessary scenario across engineers, budget owners, financial management, or executives.

Cost Optimization

Implementing cost savings shouldn't be tedious for engineers. For manual cost savings recommendations across K8s workloads and nodes, Harness provides built-in validation and cost vs. performance customization so engineers can skip straight to implementation. And with Intelligent Cloud AutoStopping, engineers can **automate** idle resource management across AWS, GCP, Azure, and K8s without taking on any risk.

In scenarios where cloud costs unexpectedly spike, CCM has FinOps and engineering teams covered with anomaly detection, which surfaces issues and their origins as soon as they happen, so customers can avoid pricey cost snowballs they won't see until the end of the month.

Cost Governance

Creating cost predictability requires process, policy, and collaboration. Tagging resources to create visibility is a big pain point for customers that CCM obviates. With cloud budgeting software and forecasting capabilities, Harness CCM keeps teams accountable for their spend and helps them to proactively avoid budget overruns.

â

Solving Practitioner Problems

Together, Harness layers intelligence into these three core pillars to help teams to alleviate the problems identified in the aforementioned graphic:

- **Getting engineers to take action** - Cutting down the effort required for an engineer to implement an optimization by surfacing the most important information as it's needed. In other cases, this means automating menial tasks like turning off compute instances that can rack up big bills over time.
- **Accurate forecasting** - Aligning cost centers to budgets and providing an accurate forecasting model that helps budget owners understand how they're trending, and when they need to take action.
- **Reducing waste or unused resources** - Providing tailored recommendations for where to optimize the infrastructure based on idle or unallocated resources. Using Cloud AutoStopping to automate detection and shutdown of idle resources to minimize wastage.
- **Container costs** - Visibility into container costs across all cloud providers and Kubernetes, showing utilized, idle, and unallocated costs. Recommendations at the Node and Workload level to optimize container costs.
- **Aligning Finance to tech teams** - Using Cloud Cost Business Intelligence and Perspectives to create cost transparency across the organization. Doing Budgeting & Forecasting to get teams on the same page about cost expectations vs. reality. Enforcing cost management policies at the infrastructure level to avoid issues altogether.

Harness CCM was built with the intention of creating collaborative cost management culture, bringing together finance, engineering, and FinOps to solve the daily challenges of managing cloud costs. Harness CCM uses intelligence to further break down those silos and simplify cloud cost management for any stakeholder involved, whether they are responsible for maintaining clear cost visibility, implementing cost savings, or for ensuring a healthy run rate of cloud costs.

With intelligent cloud cost management, Harness empowers customers to get more value out of their cloud, with less effort.

Harness: An âIntelligentâ Approach

While there are a number of good cloud cost management solutions available today, where Harness separates from the pack is in bringing intelligence and automation to the party.

More specifically, here is how Harness Cloud Cost Management differs from other solutions on the market:

- Automatically shuts down idle VMs and containers, and dynamically runs them on spot instances with no interruptions, reducing costs by up to 75%.
- Provides deep Kubernetes cost visibility & root cost analysis, and recommends ways to optimize idle/unallocated cloud spend, making containers a first-class citizen in cloud cost management.
- Intelligently achieve cost attribution, showback and chargeback by mapping organizational hierarchies to teams and projects with a modern BI experience, ultimately driving accountability.

How To Get Started

To get started with Harness Cloud Cost Management, sign up for a demo and a Harness specialist will get you going. Youâll quickly see how easy it is to get more out of your cloud, with less effort.

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: https://www.harness.io/blog/feature-flags-policy-governance-tutorial?utm_source=pmm-ff&utm_medium=how-we-use-our-own-feature-flags-at-harness

Feature Flags Policy Governance Tutorial

Harness Policy as Code, powered by Open Policy Agent (OPA), allows you to write policies as code to automatically govern flag configuration, usage, and process. Policies are applied on every flag creation, toggle, or modification, ensuring your feature flags always stay in a compliant state.

We are proud to be the first feature management solution to support this capability. Global governance policies for feature flags are a big win for engineering organizations that need to enforce standards at scale. They allow for guardrails to be put in place across all releases to ensure standards are met, and they also automate the process, so the developer experience doesn't change. Developers simply get error messages the way they're used to during the build and test phase.Â

But this post isn't about why you should do it â it's about how you do it. In this post, we go in depth and teach you all about the Feature Flags policy governance: architecture, use cases, data available to utilize, and more.

Architecture

Here's a quick overview of how we ensure that your flags always remain in a compliant state, regardless of what changes a user makes or how they decide to make them.Â Â

Writing Policies

The Harness Policy Engine is based on OPA, an easy-to-use, extensible solution for creating and enforcing policies across the entire stack. OPA is an open-source project accepted by the Cloud Native Computing Foundation (CNCF) with wide adoption across numerous software delivery use cases within the CI/CD pipeline. Policies are written in Rego as declarative code, so they are easy to understand and modify â from simple to complex use cases.

You can find more information about writing Rego rules in the official OPA docs.Â

Once you're ready to get a policy out there, you can write your own and add it to Harness. We also provide some policies out of the box for common use cases, like enforcing naming conventions and ensuring proper promotion of code.

What Data is Available to Write Policies Against?

We know that policies not only relate to the changed entity in question, but can also depend on contextual data, such as who made the change and when. To support these use cases, we provide an ever growing collection of data to the policy engine that you can use when writing your policies. This includes:

- The full feature flag configuration for the created/updated flag. This includes flag name, description, variations, rules, flag state in every environment you have, and much more.Â This is in json format, which matches the format of our public API docs, so if you interact with your flags as code already, it will be very familiar.Â
- Which user made the change, what RBAC permissions they have, and which user groups they belong to.
- When the change was made.Â

Here's a truncated example of the metadata sent to the policy engine

Debugging Policies

Policies are written as code, and as anyone who writes code knows, there will inevitably be lots of edge cases or error scenarios you want to test while refining your policies. Having to create and configure flags in special ways just to test your policies isÂ frustrating and time consuming. For this reason, we have an integrated policy tester/debugger built right into the UI.Â

Using this debugger, you can go through a standard development process:

- **Develop:** Begin by writing your policies in the main text area. Add as many rules and helper functions as you'd like.
- **Test:** Using the testing terminal, you can get quick feedback on if the new policies you're building are having the desired effect. You can hit the "Select Input" button to populate the input field with real flag data from your own project, giving you reliable and realistic test scenarios for your use cases. You can then hit the "Test" button to run the policies and see for the given data input if it would succeed or fail. You can also manually edit this inputÂ data if you'd like to hand craft particular edge cases.
- **Debug:** Once your policies are being enforced there may be a time that a user isn't sure why a change was blocked. This can be a frustrating experience if you're only provided with a vague error message such as "change forbidden." Luckily, we provide all the tools for a user to view detailed information on exactly which policies failed, along with the ability to click on these policies and enter this debugger mode, viewing the policy and the exact input their change produced. This will help them verify if it's a valid rejection or if the policy itself needs to be modified going forward.

Use Cases

Harness Policy Engine supports a wide array of use cases. These range from simple sanity checks that description fields have been properly completed to complex corporate policies that prevent changes during blackout periods. Feature flag policies broadly fall into these two categories:

Flag Configuration

- Naming conventions, e.g. flag names must match jira ticket format
- Mandatory descriptions
- Only allowing boolean flags to be created (no multivariates)
- Banning certain functionality e.g. no prerequisite rules allowed
- Max number of specific target rules

Change ManagementÂ

- Flag cannot be enabled in production unless it is enabled in QA first
- Flag changes must be made via pipelines with an approval step
- No changes during certain time periods e.g. when tests are running or during certain mandated blackout periods
- Flags can't be enabled by the same user that created it

You can get as creative as you'd like around the rules you need to enforce. We've purposely not taken an opinionated UI wizard-based approach on how to create and combine these rules, so you're free to experiment, start small, and govern what matters to you.Â

Where Can I Try It Out?

To learn more about how Harness manages policies, check out our Policies Overview for Feature Flags documentation, or sign up for a free Feature Flags trial today.Â

â

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/legal/service-packages-accelerator-premium>

Accelerator Premium

As of

This Statement of Work (this "SOW") is entered into and governed by the Master Subscription Agreement, Subscription Terms, or other

similar written agreement (the "Agreement") by and between Customer ("Customer"), and Harness Inc. ("Harness") (Customer and Harness each a "Party", and collectively, the "Parties"). This SOW and the Agreement constitute the complete agreement regarding services provided under this SOW. Except where the contrary is expressly provided, the terms and conditions of the Agreement shall prevail over any conflicting terms or conditions in this SOW.

Changes to this SOW will be processed in accordance with the procedure described below. The investigation and the implementation of changes may result in modifications to the schedule, resources, charges and/or other terms of this SOW.

Harness may utilize an employee or subcontractor (a "Partner") to provide the Services under this SOW. Harness shall remain liable or all acts and omissions of its personnel in their provision of the Services set forth in this SOW.

â

Any defined terms not specifically defined herein shall have the meaning given to them in the Agreement.

1.Â DEFINITIONSÂ

"Business Day" means any weekday, Monday through Friday, for 8 hours, generally falling between 9:00 A.M. and 5:00 P.M. in the time zone of the Harness implementation engineering team, excluding holidays and weekends, unless otherwise agreed in writing by the Parties.

"Business Week" means 5 Business Days, Monday through Friday.

"Harness Reference Architecture" means a predefined, example implementation and accompanying documentation that outlines the ideal use of Harness products, as provided and updated by Harness from time to time.

"Engagement Plan" means a project plan created by Harness after the Design & Discovery Sprint, detailing the architecture and design for the applicable software module, key project details such as milestone dates, timelines, project prerequisites, and any potential issues that may jeopardize the project timeline. This document will be used to guide the completion of this SOW.

"Project Summary" means a summary of the work completed under this SOW, which will be created by Harness upon completion of the Sprints.

â

2. PROJECT OVERVIEW

The purpose of this Accelerator (Premium) SOW is to provide Services designed to assist Customer in deploying the applicable Harness software modules, based on best practice, reference architecture and an appropriate use case for each module (use cases for all applicable modules together, the "Pilot Use Case") for a single Customer team (the "Pilot Team"). The use case selected must be (i) achievable within the project timeline, (ii) derive a first value for the customer, and (iii) Customer must be committed to deploying the use case during an agreed upon number of sprints (each, a "Sprint"). The standard timeline for the entire SOW is set forth below, in Business Weeks (the "Standard Project Duration"), and will consist of a series of Sprints, each with their own goal, as specified below. Each project begins with a Design Sprint and a Discovery Sprint, followed by one or more Implementation Sprints.

â

Harness will provide a team of service professionals, including a Solution Architect (the "SA"), and Customer Success Manager (the "CSM") who will work with the Customer for the Sprints, to implement and deliver the project as described in this SOW.

â

Harness will schedule a series of working sessions with the Customer during each Sprint. The goal of these working sessions is to provide an opportunity for live collaboration between the Parties' engineering teams. Each working session will focus on completing tasks for the current Sprint (as defined in the Engagement Plan) (detailed below), and will not exceed three (3) hours in duration per Business Day. A minimum of three (3) working sessions each Business Week is recommended. The Customer's engineer(s) should expect to perform follow-on tasks and actions from the live working sessions, and these action items will need to be completed prior to the next live session as part of the overall Sprint in order for the Sprint to move forward.

Customer shall schedule and commit its team members as appropriate to meet the Sprint schedule, and to support Harness in its delivery of the project pursuant to the Engagement Plan. Customer's commitment of its team members is essential to completing the Sprints in a timely manner.

The Accelerator (Premium) SOW is designed to provide Customer with a high level configuration summary of the Pilot Use Case, modeled after a Harness Reference Architecture. Upon completion of the project, engagement details will be documented in the Engagement Document for the customer to extend to teams beyond the pilot groups as part of the Accelerator implementation plan.

3. STANDARD TIMELINE; SCOPE OF PROJECT; SPRINT DETAILS

Timeline for Accelerator (Premium) Sprints

Sprint Name	Duration	Activities	Outcomes
-------------	----------	------------	----------

Discovery & Design Sprint	1-2 Business Weeks	3 hour planning session 3 per Business Week	Create Project Plan Begin Platform Fundamentals Scope (as applicable)
Implementation Sprint 1	2 Consecutive Business Weeks	2 hour working session 4-5 per Business Week	Complete Platform Fundamentals Scope (as applicable) Begin Module Accelerator Scope
Implementation Sprint 2	1-2 Consecutive Business Weeks	2 hour working session 4-5 per Business Week	Module Accelerator Scope
Implementation Sprint 3	1-2 Consecutive Business Weeks	2 hour working session 4-5 per Week	Complete Module Accelerator Scope

Scope of Project and Accelerator Kickoff

The procedures set forth below and in the appendices (as referenced) apply to the duration of this Accelerator (Premium) SOW, for the implementation and onboarding of multiple Harness software modules, in specific bundles as further detailed in the relevant Module Appendix (the "Product(s)"). This SOW includes either the "DevOps Accelerator" bundle or the "DevSecOps Accelerator" bundle (as specified in the applicable Order Form) and the agreed upon Pilot Use Case.

The SOW shall begin with a kickoff meeting, known as the "Accelerator Kickoff", which will be scheduled by the Harness team taking into account Customer's preferred schedule. The intent of the Accelerator Kickoff is to introduce and align stakeholders, decision makers, engineers and the Harness team to ensure clear understanding of the process to be followed and scope of this Accelerator (Premium) SOW. The Accelerator Kickoff is also intended to determine any onsite logistics (if necessary), key technical resources needed from the Customer, and general ways of working during the engagement. The outcome of this meeting should result in the scheduling of the technical planning Discovery and Design Sprint.

Sprint Details

Discovery and Design Sprints

The purpose of the Discovery & Design Sprint is to review all the technical implementation planning required to implement and onboard the Product within Customer's environment, to achieve the Pilot Use Case. During this Sprint, the Parties will also review the Product configuration requirements (for each relevant product module), refine the scope of and agree to the dates and staffing requirements for the Implementation Sprint(s). The outcome of this Discovery & Design Sprint are to ensure the Customer and Harness align the project schedules, goals, and scope of this SOW, and to produce the Engagement Plan, which will contain all such details. The Discovery & Design Sprint must be completed prior to starting the Implementation Sprints. This Sprint will cover the following, as applicable:

Discovery:

- Understand the Customer environment in which the Products will be utilized
- Identify Product(s) goals, which can include:
 - â Details about the Customer's legacy solutions being replaced/enhanced by the Product
 - â KPI metrics for project success
 - â Product utilization infrastructure
 - â Disaster recovery requirements
 - â Product logging
 - â Pilot Use Case details
 - â The end-users on Pilot Team
- Identify Products(s)(and Harness Platform) configuration goals, which can include:
 - â Single Sign On for user management
 - â RBAC Persona requirements
 - â Security Compliance requirements and stakeholders
 - â System/Network requirements and stakeholders
- Identify Product(s) adoption goals, which can include:
 - â Mapping of Customer organization structure / business unit hierarchy to Product account hierarchy
 - â Automated onboarding / propagation
 - â End-user experience standards
 - â UI, CLI, and/or code commit

Design:

- Review Product implementation designs based on reference architectures and recommended practices
- Walk through automated Product deployment methodology
- Understand how the Product will be implemented within the Customer environment (non-production)

- Establish the Product implementation goals, which can include:
 - â Zero-trust, low-trust, or high-trust security compliance
 - â Self-service onboarding automation
 - â Templatization use requirement standards
 - â Governance policy outcomes
 - â Notification and alert postures
- Identify prerequisites and/or dependencies to be addressed before Implementation Sprints can begin, this may include factors such as:
 - â Networking rules for ingress and egress
 - â Required security exceptions
 - â Delegate Architecture topology and requirements
 - â Systems in-scope for Product integration
- Requirements for Product-specific implementation that will be included in this Accelerator (Premium) SOW are set forth in the relevant Module Appendix.

Technical Implementation Planning: To be completed during both Discovery and Design Sprints

- Identify potential adoption use case(s)
- Refine and prioritize project scope
- Outline implementation prerequisites
- Identify project team members by role, organization (Harness, Customer, or Partner, and business unit)
- Determine agreed upon dates for the Implementation Sprints
- Collect onsite visit logistics (if applicable)
- Document Project details in the Engagement Plan

Sprints Deliverable: PROJECT PLAN

- Delivered within one Business Week of the completion of the Sprint activities described above.
- The Engagement Plan may contain the following, as appropriate:
 - â A high level installation architecture and design for the Product
 - â The software and staff prerequisites to support the Project
 - â Project dates and timelines, defined by sprint outlining prioritized tasks and deliverables
 - â Will highlight risks/issues and/or blockers that put the timelines at risk
 - â Prerequisites / tasks to be completed by the Customer before Implementation Sprints can begin, including providing the required people that must be available during the Sprint timelines.
- The Engagement Plan will be available via the Harness Professional Services Portal (a link to this Portal will be provided) for Customer to view and track Customerâs participation throughout the term of this SOW.

Acceptance Criteria for Deliverables; Acceptance Process:

- The Customer will have three (3) days to review the Engagement Plan and provide approval.Â If the Customer rejects any aspect of the Engagement Plan, the Parties will work together to address the issue(s) and an updated Engagement Plan will be provided for review and Customer will have an additional three (3) days to review the revised Engagement Plan.Â This approval cycle will continue up to (2) times, or until the Engagement Plan has been approved by Customer, whichever comes first.Â Â
- If Customer does not provide an affirmative approval or rejection of the Engagement Plan during the 3 day review period, or after the second approval cycle, the Engagement Plan shall be deemed automatically approved.Â
- Implementation Sprints will not be scheduled until the Engagement Plan is approved. Once approval is provided, the Implementation Sprints will be scheduled. Harness requires up to 3 Business Days to finalize the scheduling for the Implementation Sprints.

Implementation Sprint(s):

The purpose of the Implementation Sprints is to ensure the Engagement Plan as documented and agreed upon in Discovery and Design Sprint is delivered. The Implementation Sprint timelines are defined and prioritized in the Engagement Plan.

Customer Prerequisites

- Customer agrees that the following will be completed prior to the start of Implementation Sprints:
 - â Review and approval of the Engagement Plan
 - â Completion of implementation prerequisites set forth in the Engagement Plan

Platform Setup

- Single Sign-on integration with one (1) supported Identity Provider
- Implement the Pilot Account Hierarchy
- Configure Persona-based RBAC
- Secret Manager integration with one (1) supported Encrypted Secret Storage Provider

Module Pilot Implementation

- Implementation of the applicable Products based on the architecture and design outlined in the Engagement Plan, and based on the best practice scope relevant to the applicable Product (as outlined in the relevant Module Appendix).

Customer Validation

- Review implementation with Customer administrators / architects
- Review implementation with Pilot Team
- Successful production-grade Product implementation for Pilot team, with verification of success via KPIs agreed upon in Engagement Plan

Self-service Automation

- Implement the self-service automation capabilities (or other product-specific resource automation capability) specifically outlined during the technical implementation planning phase of the Discovery & Design Sprint. Harness will make reasonable efforts to incorporate any optimizations discovered during the implementation of the Pilot Use Case and any Customer feedback.

Calibrate for Adoption

- A phase of activities that include:
 - â User Training & Enablement
 - â Self-service user Onboarding
 - â Automated Org and/or Project Onboarding
 - â Automated Resource Configuration
 - â Adoption of pre-configured templates

Deliverables

- Product implementation is complete and accessible by the Customer (in a non-production environment).
- Knowledge transfer of the services rendered as part of this SOW.
- Creation of Project Summary:
 - â Customer specific notes about installation, configuration, and adoption of the Product
 - â Important findings during implementation
 - â Additional recommendations the Customer may wish to perform after the engagement (which not in scope for this SOW)

RESPONSIBILITIES; LIMITATIONS; SCHEDULING; OWNERSHIP

Customer Responsibilities:

Customer acknowledges and agrees that Harness's ability to deliver the Services is dependent upon Customer's full and timely cooperation with Harness, as well as the accuracy and completeness of any information and data Customer provides to Harness. Customer is responsible for delays and any additional costs caused by Customer's failure to comply with their responsibilities. Harness will not be responsible for any resulting fees or expenses.

- Customer will complete their required prerequisites set forth in the Engagement Plan. These prerequisites may include, but are not limited to:
 - â Permissions to manage cloud/on-premise infrastructure
 - â DNS entries
 - â TLS Certificate and related keys
 - â Cloud/on-prem network and/or proxy configuration, if needed
 - â Airgap systems, if needed
- Customer will provide all requested contact information for the Customer's key business leaders and stakeholders required for Harness to provide the Services
- Customer will coordinate communications with, and provide contacts for, all third Parties necessary for the completion of the Engagement Plan
- Customer will deliver timely responses to requests for information, technical questions and requested decisions pertaining to the project and this SOW. Harness shall not be responsible for any delays in the provision of Services due to Customer's non-compliance with this requirement
- Customer will diligently manage and supervise Customer's other contractors and service providers as necessary to support the Services
- Customer will coordinate Service implementation on third-Party-maintained hardware/software (if applicable) with Harness
- Customer will assign a designated Subject Matter Expert coordinator (SME) from Customer's staff who, on its behalf, will grant all approvals, provide information, make decisions, attend meetings, and otherwise be available to assist Harness in facilitating the delivery of the Services
- Customer will assign a designated engineer and appropriate supporting staff to implement the Engagement Plan
- Customer will assign a designated project manager to facilitate on time delivery of Engagement Plan
- Customer will ensure the availability of all hardware, firmware, and software required by the SA/SEN to deliver the Services and provide network access to integrate required tooling and systems with the Product including (but not limited to) Artifact Repositories, Version Control Systems, Identity Providers, Log Aggregators, Security Scanning Tools, Monitoring Systems, and Developer Tool platforms
- Customer will provide reasonable access and working space at the site as Harness may request

Harness Responsibilities: Harness is responsible for providing the Services outlined in Section 3 above. Where Harness is performing Services on-site for a Customer, Harness shall ensure that its personnel conduct themselves in accordance with the standard health, safety, and security policies of Customer applicable to its staff and or visitors generally, as long as such policies are provided to Harness

in writing, in advance. Customer agrees to provide written notice to Harness of any applicable non-standard policies (for example, the requirement of security clearance) prior to executing an Order Form for the Services.

Limitations/Exclusions: Customer acknowledges and agrees that following limitations apply to this SOW:

- The Product implementation will be completed using only approved methods by Harness for generally available versions of the Product
 - â Terraform will be used for automated deployment of the Product when deploying to the commercial offerings of Azure, AWS, or GCP, and manual otherwise
 - â The Product implementation will be modeled after Harness Reference Architecture and Harness best-practices
- The Module Accelerator Scope will be scoped to a single Pilot Use Case
- The implementation of the applicable Product Bundle will be limited to one (1) Pilot Team.
- Single Sign On configuration leveraging a non-tested Identity Management Platform that supports the SAML 2.0 standard can only be accommodated on a âbest effortâ basis and is not guaranteed
- Customer must make available its Identity Management Platform administrator to make necessary changes outside the Product when configuring Single Sign On
- The Product implementation in scope for this engagement will be âproduction-gradeâ and maintain a sightline to a Production implementation, BUT it is the Customer's responsibility and prerogative to promote the Pilot Use Case from a non-production environment to the Customerâs production environment.
- Governance Policies delivered as part of this engagement will be sourced from out-of-the-box examples, and any needed modifications can be accommodated on a best effort basis only and are not guaranteed.
- Customer is responsible for any system integrations that are not supported by out-of-the-box integrations. Â
- Knowledge transfer sessions will be limited to no more than two (2) sessions, totaling no more than four (4) hours in the aggregate, over the entire term of this SOW.
- The Services will be delivered as a single, continuous workstream, as set forth in the Engagement Plan.Â Environments requiring multiple engagements, phases, or timelines that exceed the standard timelines are excluded from the scope of this SOW.Â If Customer requires additional scope, the Parties must execute a Change Order or enter into a separate Order Form and Statement of Work.
- Recording of Meetings and Working Sprints is not allowed unless agreed and documented in the Engagement Plant.
- All services are provided remotely unless pre-approved travel arrangements are made with the Customer, subject to the expense reimbursement guidelines set forth in Section A 5 below.
- Any implementation of recommendations provided by Harness, such as system configuration, performance tuning, or database design improvements, which are not recommendations for the Product itself, are excluded.
- Once the Implementation Sprint is completed, Customer is responsible for maintaining any changes or improvements to any and all code artifacts, automation work, and/or documentation delivered under this SOW, including the continued support and maintenance of the foregoing.
- The following are expressly excluded from the scope of Services provided under this SOW: (i) hardware maintenance and repair; (ii) software maintenance and upgrades, (iii) instructor led training, (iv) operational runbooks or execution plans.
- Harness standard maintenance and support services are provided in accordance with Harnessâ maintenance and support policy, and such services are not part of this SOW.Â
- The Services provided under this SOW are specific to the designated Product licensed by Customer only, and do not include any consultation, guidance, or support for any other software, solutions or materials.

Scheduling/Governance of Sprints:

- A Sprint duration is defined as a minimum of one (1) Business Week, and a maximum of two (2) consecutive Business Weeks
- Each Sprint will align with a Business calendar week
- Time between Sprints cannot exceed more than two (2) Business Weeks
- Once a Sprint has started, the Customer is committed to the scope, deliverables and timeline, and the Sprint may not be suspended or delayed.
- If a Sprint is suspended or delayed due to the acts or omissions of Customer or its personnel, then the remaining work for that Sprint will be canceled, if it is the final Sprint that is being suspended. If desired, Customer can request a Change Order for the outstanding work.
 - â If the canceled Sprint is not the final Sprint, the remaining scope of work under that Sprint will be moved to the next scheduled Sprint as the first and priority tasks.
 - â The remaining Sprint scope will be worked on a best effort basis, keeping the originally scheduled Sprint dates.
 - â If the requested suspension or delay impacts the timeline of the remainder of the Engagement Plan, Customer may re-engage and reschedule the Sprint scope for the next available Sprint, working with the Harness project manager or CSM to determine the appropriate timeline given the availability of Harness resources. If there are no remaining Sprints under this SOW, Customer must request a Change Order for the additional work requested.
- If Customer needs to delay a future Sprint and deviate from the Engagement Plan, Customer shall notify their project manager or CSM in writing at least five (5) business days prior to the agreed upon start date of the Sprint. Â If Customer does not provide such notice, the Sprint will automatically be canceled and all services thereunder will be forfeited by the Customer.
 - â After the delay of a Sprint, Customer must make a written request to Harness to re-engage. Such request must be made at least fifteen (15) Business Days prior to the new requested Start Date. Harness will work with Customer to determine the new Start Date, based on Harness resource availability. The new agreed upon Start Date and any other appropriate updates will be made to the Engagement Plan. The updated Engagement Plan must be approved by Customer in accordance with the approval process outlined above. Once the Engagement Plan has been approved, the Sprint will officially be rescheduled.

*Ownership:*Â

As between Customer and Harness, Harness retains all right, title and interest in the services provided and any results of thereof including the deliverables specified in this SOW (collectively, âDeliverablesâ).Â Subject to Customerâs payment in full of all fees due

under this SOW, Harness grants Customer a limited, non-exclusive, non-transferable and non-sublicensable right and license to use the Deliverables during the applicable Subscription Term in connection with Customer's use of the applicable Products in accordance with the Agreement. For the avoidance of doubt, Deliverables do not include any Customer Confidential Information, and any Customer Confidential Information contained in the Deliverables shall remain the sole and exclusive property of Customer.

5. FEES AND EXPENSES

Customer shall pay to Harness the fees set forth in the applicable Order Form for the services. All fees are due up front, in full, and must be paid within 30 days of the invoice date. The payment of all fees is subject to the terms and conditions set forth in the Agreement.

If the work under this SOW exceeds the Standard Project Duration, or if Customer requests or requires additional hours beyond the scope of this SOW, the Parties shall execute a Change Order to this SOW as described in the Change Management section below.

Any Services performed outside of the Standard Project Duration or any additional hours of work will be provided by Harness on a time-and-materials (T&M) basis only, at an hourly rate of \$325 per hour, unless otherwise set forth in the Change Order. Customer must order a minimum of 10 hours per Business Week under such Change Order, and all additional hours purchased must be utilized within 6 months of the date of the Change Order, or else they will be forfeited. Any work provided on a T&M basis will be invoiced by Harness monthly in arrears. Once the Parties execute a Change Order for additional hours, the fees for such hours are non-cancellable and non-refundable. At the end of the six (6) month period, Harness shall invoice Customer for any remaining hours purchased.

Furthermore, if Customer changes any requirements, additional hours may be required to complete the Services, and the Parties will execute a Change Order to facilitate the change in scope. Such a Change Order may increase the Price of the Services. No additional Services shall be provided unless the Change Order is agreed to in writing by the Parties.

Expenses are not included in the fee set forth above. All expenses incurred shall be in accordance with the Customer's T&E corporate policy if provided to Harness in writing in advanced, otherwise such expenses shall be incurred in accordance with Harness's T&E policy. If travel expenses are incurred with respect to this Professional Services program, Harness will invoice for travel expenses as incurred and monthly in arrears and Customer shall pay such invoices in accordance with the terms and conditions set forth in the Agreement. For the avoidance of doubt, all travel expenses must be pre-approved by the customer prior to any trips as part of this SOW. The duration of visits requested by the Customer cannot exceed two (2) business days unless mutually agreed in writing.

6. TERM AND TERMINATION

This SOW shall terminate upon the earlier of (i) six (6) months from the date of the last signature, below, or (ii) completion of the Services described above. Customer may request to extend the termination date of this SOW by submitting a Change Request as described in the Change Management section below. If the Change Request is accepted, a reinstatement fee of up to 10% may apply.

Upon the expiration or termination of the Agreement or this SOW pursuant to the Agreement, all amounts (including expenses) owed to Harness for Services under this SOW (whether completed or not), will be immediately due and payable in full. Upon termination of the Agreement or a SOW by Customer for any reason pursuant to the Agreement, Customer shall be responsible for payment of all fees for Services rendered and expenses incurred prior to the date of termination. In addition, upon any termination or expiration of the Agreement or this SOW, Harness's obligation to provide Services shall immediately terminate.

7. CHANGE ORDERS

Unless otherwise mutually agreed by the Parties, any change or modification of this SOW or the Engagement Plan will be coordinated by the Parties in accordance with this Section 6. Either Party may initiate such requests for a change or modification (each, a "Change Request"). The Party requesting a Change Request will submit a written Change Request to the other Party in a clear and concise manner using a form substantially similar to the sample document attached hereto as Annex 1 (a "Change Order"). Upon the execution of a mutually agreed upon Change Order by both Parties, the obligations of the Parties with respect to such Change Request will be incorporated under the SOW. If a Change Order reinstates an expired SOW, a reinstatement fee of up to 10% may apply.

Source URL: <https://www.harness.io/legal/service-packages-residency-pod>

Residency Pod

As of

This Statement of Work (this "SOW") is entered into and governed by the Master Subscription Agreement, Subscription Terms, or other similar written agreement (the "Agreement") by and between Customer ("Customer"), and Harness Inc. ("Harness") (Customer and Harness each a "Party", and collectively, the "Parties"). This SOW and the Agreement constitute the complete agreement regarding services provided under this SOW. Except where the contrary is expressly provided, the terms and conditions of the Agreement shall prevail over any conflicting terms or conditions in this SOW.

Changes to this SOW will be processed in accordance with the procedure described below. The investigation and the implementation of changes may result in modifications to the schedule, resources, charges and/or other terms of this SOW.

Any defined terms not specifically defined herein shall have the meaning given to them in the Agreement.

â

1. PROJECT OVERVIEW

This SOW covers services intended to provide engineering expertise to provide strategic architecture expertise from a Residency Pod (âRPâ) with respect to utilizing the Harness Platform and the software modules licensed by Customer (the âSolution(s)â), including consulting with the Customer in connection with Customerâs deployment, management and integration of Solutions in Non-Production Environments, (the âRP Engagementâ). The RP is a set of individuals designated to the Customer account that provides strategic and tactical guidance on Customer processes and initiatives with respect to cloud transformation utilizing Harness Solutions. Harness may utilize an employee or subcontractor (a âPartnerâ) to provide the Services under this SOW. Harness shall remain liable for all acts and omissions of its personnel in their provision of the Services set forth in this SOW.

â

2. DEFINITIONS AND DELIVERY ROLES

âNon-Production Environmentâ is limited to development and test environments.

âRSAâ is the solutions architect (âSAâ) designated to the Customer account that provides strategic and tactical guidance on Customer processes and initiatives with respect to utilizing the Solutions.

âRSEâ is the solutions engineer designated to the Customer and identifies Customer challenges, captures critical business requirements, and provides pair programming, sample code and guided hands on keyboard prescriptive solutions based on established best practices if an SA is not present in the account.â

âRPEâ is the Partner Engineer assigned to the Customer that will support the tactical and strategic deliverables defined by the SA and/or SE and will establish longer term direct retainers with the Customer.

â

The RP Engagement will include up to three (3) of the roles identified and described below.

Role	Responsibilities and Skills
RSA	<ul style="list-style-type: none">• Facilitate a Technical Planning Session(s) within the first 10 days of the Retainer Term to determine specific adoption work streams for the RP to accomplish in the period.<ul style="list-style-type: none">◦ Collaborate with the Customer and the RSE assigned to the account on the adoption roadmap and technical execution plans regarding adoption and continued implementation of Solutions beyond the reference architecture and baseline implementation, including endorsed integrations with Solutions.• Guide the Customer in execution of an adoption plan created by the RSA.• Provide prescriptive guidance and hands on keyboard guidance to the Customer on deployment, management, and consumption of Solutions.• Provide strategic guidance to Customer executives and management on adoption strategies, workflows, utilization, and limited integrations of Solutions for critical business processes.• Provide assistance to help align Customer's organization, at each level, regarding their roles and responsibilities towards their Harness Solutions adoption strategy.• Work with Customer users to understand the importance of the Solutions as part of technical transformation and how to best leverage those Solutions in their day-to-day workflows.
RSE	<ul style="list-style-type: none">• Execute any guided hands on keyboard, sample code and adoption work product to help customer achieve adoption outcomes during the retainer term• Assist the RSA with documentation and demonstrations associated with adoption strategies.• Help Customer with any test plans and quality assurance with any guided solutions to customer non-production environments• Assist RSA with working session sprints during the RP Engagement.
RPE	<ul style="list-style-type: none">• Execute any guided hands on keyboard, sample code and adoption work product to help customer achieve adoption outcomes during the retainer term• Assist the RSA with consulting work streams associated with the adoption plan• Assist the RSA with delegated work product in the form of hands on keyboard guidance, enablement of customer groups on the Solutions as part of adoption goals and objectives

- Facilitate a Technical Planning Session(s) within the first 10 days of the Retainer Term to determine specific adoption work streams for the RP to accomplish in the period.
 - Collaborate with the Customer and the RSE assigned to the account on the adoption roadmap and technical execution plans regarding adoption and continued implementation of Solutions beyond the reference architecture and baseline implementation, including endorsed integrations with Solutions.
- Guide the Customer in execution of an adoption plan created by the RSA.
- Provide prescriptive guidance and hands on keyboard guidance to the Customer on deployment, management, and consumption of Solutions.
- Provide strategic guidance to Customer executives and management on adoption strategies, workflows, utilization, and limited integrations of Solutions for critical business processes.

- Provide assistance to help align Customer's organization, at each level, regarding their roles and responsibilities towards their Harness Solutions adoption strategy.
- Work with Customer users to understand the importance of the Solutions as part of technical transformation and how to best leverage those Solutions in their day-to-day workflows.
- Collaborate with the Customer and the RSE assigned to the account on the adoption roadmap and technical execution plans regarding adoption and continued implementation of Solutions beyond the reference architecture and baseline implementation, including endorsed integrations with Solutions.
- Execute any guided hands on keyboard, sample code and adoption work product to help customer achieve adoption outcomes during the retainer term
- Assist the RSA with documentation and demonstrations associated with adoption strategies.
- Help Customer with any test plans and quality assurance with any guided solutions to customer non-production environments
- Assist RSA with working session sprints during the RP Engagement.
- Execute any guided hands on keyboard, sample code and adoption work product to help customer achieve adoption outcomes during the retainer term
- Assist the RSA with consulting work streams associated with the adoption plan
- Assist the RSA with delegated work product in the form of hands on keyboard guidance, enablement of customer groups on the Solutions as part of adoption goals and objectives

â

3. SERVICES; AVAILABILITY

Services

During the Retainer Period, the RP will complete the following activities:

- **Architecture Design and Reviews:** Collaboratively design the architecture for new Harness solution and adoption use cases and review and provide feedback on existing architecture designs under use. Ensure that architectural decisions align with organizational goals and standards.
 - â Adoption Management: of up to 3 additional adopter teamsâ adoption of the Pilot Use Case onto Harness solutions (applicable to a 60 day Retainer Period) or 6 additional adopter teamsâ adoption of the Pilot Use Case (applicable to a 90 day Retainer Period).â
 - â Additional Use Cases: of up to 3 new use cases that fall within the scope found in the applicable Module Appendixâ
 - â Guided White Boarding Sessions on adoption plan tactics
 - â Guided Hands on Keyboard for configuration for adoption plans and enablement plans to Harness
 - â Guided Hands on Keyboard for pilot adoption use cases or adoption focused use cases
 - â Assisted facilitated workshops for groups to use the Harness solutions
- **Technical Mentoring:** Offer technical mentorship over the course of the retainer. Help users understand and implement Harness-module architectural patterns, best practices, and solution standards.
- **Implementation Documentation Updates:** Maintain Harness-module architectural documentation that is accessible to customer teams. Ensure that architecture diagrams, design decisions, and usage guidelines are well-documented.â
- **Adoption Issue Resolution:** Assist in troubleshooting complex technical issues (not break/fix) and provide guidance on resolving architectural challenges during the retainer period.
- **Developer Experience Change Management** - people and process transformation tactics and strategy (Partner and Harness Subject Matter Expert delivered working sessions)
 - â Phased approaches and planning
 - â Design and Discovery for net new use cases and additional Adopter-teams
- **Pod Success Management** - Weekly/Bi-Weekly cadences with Harness Customer Success and Customer teams on Pod activities and progress.

Exclusions:â The following activities are outside the scope of the RP Engagement:

- Hands-on interaction with production customer systemsâ
- Non-Harness Solutionsâ
- Harness University Enablementâ
- Services outside of the Business Day determined in the RP planning sessions
- Creation, delivery or support of custom-built modules or extensions for any Solutions
- Extensions or integrations not delivered with Solutionsâ
- Project management of Customer resources, teams, and/or other 3rd party resources outside the RP

Availability

A âBusiness Dayâ means any weekday, Monday through Friday, generally falling between 9:00 A.M. and 5:00 P.M.â in the time zone of the RP team, excluding holidays, weekends, and reasonable PTO. A âBusiness Weekâ means 5 Business Days, Monday through Friday, except company holidays in the country where the RP resource resides.â A

The RP will work only within a normal Business Day and be available for up to sixteen (16) hours per Business Week for the duration of the RP Engagement.â Harness shall have the right, in its sole discretion and for any reason, to assign a new RSA or RSE or RPE with prior written notice,â provided it does not have a materially negative impact on the velocity or quality of the work being performed.

â

4. RESPONSIBILITIES AND INTELLECTUAL PROPERTY RIGHTS

Customer acknowledges and agrees that Harness's ability to deliver on this RP Engagement is dependent upon Customer's full and timely cooperation with Harness, as well as the accuracy and completeness of any information and data Customer provides to Harness. Customer is responsible for delays and any additional costs caused by Customer's failure to comply with their responsibilities. Harness will not be responsible for any resulting fees or expenses.

Customer will promptly:

- Provide all requested and required contact information for Customer key business leaders and stakeholders.
- Coordinate necessary communications with, and provide contacts for, all third parties. Coordinate and drive any required change/design approvals.
- Deliver timely responses to requests for information, technical questions and technical decisions.
- Diligently manage and supervise Customer's other contractors and service providers
- Coordinate deployment on third-party-maintained hardware/software (if applicable) with Harness.
- Assign a designated Technical Subject Matter Expert (âSMEâ) from Customer's staff who, on its behalf,
- will grant all approvals, provide information, make decisions, attend meetings, facilitate access to applicable customer data, systems, networks or environments and otherwise be available to assist Harness in facilitating the delivery of the RP Engagement.
- For onsite delivery, provide all requisite logistical accommodations to the RP, which include but are not limited to:
 - access to adequate physical work location,
 - Customer's network, internet access, and other applicable systems and facilities.
- For remote delivery, grant Harness appropriate system access to non-production environments only for the purposes of guided advisory work.
- Ensure the availability of all hardware, firmware, and software required to deliver the RP Engagement using the Solutions.
- Provide a written acknowledgement (signoff) upon the completion of the RP Engagement.
- Provide a customer feedback survey upon completion of the RP engagement.
- Be responsible for decisions regarding Customer production environments, including the decision to implement results of the RP Engagement that are implemented in Non-Production Environment(s). For clarity, Harness disclaims all liability related to any implementation of results of this RP Engagement to Customer production environments, changes to Customer production environments, and/or decisions made by Customer regarding Customer production environments.

Harness is responsible for providing the Services outlined in Section 3 above.

As between Customer and Harness, Harness retains all right, title and interest in the services provided and any results of thereof (collectively, âDeliverablesâ). Subject to Customer's payment in full of all fees due under this SOW, Harness grants Customer a limited, non-exclusive, non-transferable and non-sublicensable right and license to use the Deliverables during the applicable Subscription Term in connection with Customer's use of the applicable Harness Solutions in accordance with the Agreement. For the avoidance of doubt, Deliverables do not include any Customer Confidential Information.

â

5. FEES AND EXPENSES

Customer shall pay to Harness the fees set forth in the applicable Order Form for the services. All fees are due up front, in full, and must be paid within 30 days of the invoice date. The payment of all fees is subject to the terms and conditions set forth in the Agreement.

Expenses are not included in the fee set forth above. All expenses incurred shall be in accordance with the Customer's T&E corporate policy if provided to Harness in writing in advanced, otherwise such expenses shall be incurred in accordance with Harness's T&E policy. If travel expenses are incurred with respect to this Professional Services program, Harness will invoice for travel expenses as incurred and monthly in arrears and Customer shall pay such invoices in accordance with the terms and conditions set forth in the Agreement. For the avoidance of doubt, all travel expenses must be pre-approved by the customer prior to any trips as part of the RP Engagement. The duration of visits requested by the Customer cannot exceed two (2) business days.

â

6. TERM AND TERMINATION

The RP Engagement shall be provided for the duration set forth on the applicable Order Form, calculated from the date Customer initiates that RP Engagement (the âRetainer Periodâ).

Customer may initiate the RP Engagement after the Start Date listed on the applicable Order Form, provided that Customer provides Harness with at least 10 business days' notice of its intended start date. The RP Engagement must be initiated prior to the End Date specified in the Order Form, or else they will automatically expire.

This SOW shall terminate upon the earlier of (i) the last day of the Retainer Period, or (ii) the End Date set forth in the applicable Order Form, unless terminated earlier by either Party in accordance with the terms of the Agreement.

Upon the expiration or termination of the Agreement or this SOW pursuant to the Agreement, all amounts (including expenses) owed to Harness for Services under this SOW (whether completed or not), will be immediately due and payable in full. Upon termination of the Agreement or a SOW by Customer for any reason pursuant to the Agreement, Customer shall be responsible for payment of all fees for Services rendered and expenses incurred prior to the date of termination. In addition, upon any termination or expiration of the Agreement or this SOW, Harness's obligation to provide Services shall immediately terminate.

â

7. CHANGE ORDERS

Unless otherwise mutually agreed by the Parties, any change or modification of this SOW will be coordinated by the Parties in accordance with this Section 7. Either Party may initiate such requests for a change or modification (each, a "**Change Request**"). The Party requesting a Change Request will submit a written Change Request to the other Party in a clear and concise manner using a form substantially similar to the sample document attached hereto as Annex 1 (a "Change Order"). Upon the execution of a mutually agreed upon Change Order by both Parties, the obligations of the Parties with respect to such Change Request will be incorporated under the SOW. If a Change Order reinstates an expired SOW, a reinstatement fee of up to 10% may apply.

Source URL: <https://www.harness.io/legal/service-packages-catalog-dollars>

Statement of Work: Service Catalog Dollars

As of Jan 12, 2024

This Statement of Work (this "SOW") is entered into and governed by the Master Subscription Agreement, Subscription Terms, or other similar written agreement (the "Agreement") by and between Customer ("Customer"), and Harness Inc. ("Harness") (Customer and Harness each a "Party", and collectively, the "Parties"). This SOW and the Agreement constitute the complete agreement regarding services provided under this SOW. Except where the contrary is expressly provided, the terms and conditions of the Agreement shall prevail over any conflicting terms or conditions in this SOW.

Changes to this SOW will be processed in accordance with the procedure described below. The investigation and the implementation of changes may result in modifications to the schedule, resources, charges and/or other terms of this SOW.

Any defined terms not specifically defined herein shall have the meaning given to them in the Agreement.

1. PROJECT OVERVIEW

This SOW covers services intended to provide engineering expertise related to the implementation of Harness's software complete platform (the "Solution"). Under this SOW, Customer may redeem the service catalog dollars ("SCDs") it has purchased under an applicable Order Form, for service packages from the Harness Professional Services catalog. Each service package fulfills a different purpose, and there are service packages to assist Customer with the implementation, deployment, migration, and adoption of the Solution. Harness may utilize an employee or subcontractor to provide the Services. Harness shall remain liable for all acts and omissions of its personnel in their provision of the Services set forth in this SOW.

2. REDEMPTION RULES

The SCDs are valid for 12 months from the Start Date set forth on the Order Form (the "Retainer Period"). Any SCDs not used during this period will automatically expire and are forfeited by Customer.

This SOW and the Retainer Period will begin with a kick-off meeting, to be scheduled at time mutually agreed upon. The purpose of this call is to align all key stakeholders and to outline the process and purposes of the SOW and various service packages.

The SOW includes the quantity of SCDs ordered by the Customer under the applicable Order Form. SCDs are non-discountable, and the service packages for which the SCD can be redeemed are not customizable.

These SCDs can be redeemed for any of the Service Packages set forth in Section 3, at the rates set forth therein, at any time during the Retainer Period. The SCDs are valid from the first day of the Retainer Term, and a portion of the SCDs will expire, if not used, in accordance with the quarterly consumption schedule set forth below. Any SCDs not redeemed prior to the quarterly expiration date will be non-redeemable.

Customer must utilize at least 25% of its total SCDs purchased every quarter (the "Minimum Spend"). The quarter shall be measured from the day the Retainer Period begins. Any portion of the Minimum Spend that is not utilized by Customer within the applicable quarter is automatically forfeited by the Customer. The table below exemplifies Minimum Spend requirements and remaining credit amounts after the completion of a quarter, based on a purchase of 100,000 SCDs. For clarity, this table is for demonstration purposes only, Customer's Minimum Spend amounts and remaining credit amounts after each quarter will depend on the actual number of SCDs purchased in the applicable Order Form.

	First Quarter	Second Quarter	Third Quarter	Fourth Quarter
Starting Balance	100,000	75,000	50,000	25,000

	First Quarter	Second Quarter	Third Quarter	Fourth Quarter
Minimum Spend Requirement	25,000	25,000	25,000	25,000
Remaining Balance	75,000	50,000	25,000	0

3. A SERVICE PACKAGE OPTIONS AND RATES

Service Package Name	SCD Rate	Duration Of Services
Accelerator Standard Implementation	35,000	4 Weeks
Accelerator Premium Implementation	60,000	8 Weeks
Residency Pod 90 days	125,000	12 Weeks
Residency Pod 60 days	75,000	8 Weeks
Harness University	5,000	1 Week
Partner-led Program/Workshop	To be specified by the parties	To be specified by the parties

For more information on each Service Package, please see the descriptions here.

4. A CUSTOMER OBLIGATIONS; REQUIREMENTS; SCHEDULING

Customer Obligations

Scheduling Service Packages

5. A TERM AND TERMINATION

This SOW shall terminate upon the earlier of (i) the date Customer has utilized all of its SCDs and the service packages redeemed have been completed, (ii) the Retainer Period ends, or (iii) the End Date set forth in the applicable Order Form, unless earlier termination by either Party in accordance with the terms of the Agreement.Â

Upon the expiration or termination of the Agreement or the Order Form, all amounts (including expenses) owed to Harness under this SOW (whether completed or not), will be immediately due and payable in full. In addition, upon any termination or expiration of the Agreement or the applicable Order Form, this SOW shall terminate and Harnessâs obligation to provide Services shall immediately terminate.Â

6. A FEES AND EXPENSES

Customer shall pay to Harness the fees set forth in the applicable Order Form for the SCDs.Â All fees are due up front, in full, and must be paid within 30 days of the invoice date.Â The payment of all fees is subject to the terms and conditions set forth in the Agreement.

Expenses are not included in the fee set forth above.Â Customer shall pay for all reasonable expenses incurred by Harness and its personnel in the performance of this SOW, provided that such expenses meet requirements set forth in the terms and conditions of the Agreement.Â Harness shall invoice Customer monthly for any applicable expenses in and Customer shall pay such invoices in accordance with the terms and conditions set forth in the Agreement.

7. A CHANGE ORDERS

Unless otherwise mutually agreed by the Parties, any change or modification of this SOW or the Project Plan will be coordinated by the Parties in accordance with this Section 6. Either Party may initiate such requests for a change or modification (each, a "**Change Request**"). The Party requesting a Change Request will submit a written Change Request to the other Party in a clear and concise manner using a form substantially similar to the sample document attached hereto as Annex 1 (a "Change Order"). Upon the execution of a mutually agreed upon Change Order by both Parties, the obligations of the Parties with respect to such Change Request will be incorporated under the SOW. Â If a Change Order reinstates an expired SOW or otherwise alters the duration of any service package, a reinstatement fee of up to 10% may apply.

Source URL: https://www.harness.io/case-studies/empower-engineers-with-cd?utm_source=internal&utm_medium=blog&utm_content=advanced-ccm-case-study

Advanced Empowers 700 Engineers With Harness

Advanced saw a 10x return on their investment in 2 months. Find out how.

About Advanced

Advanced is the UKâs third largest provider of business software and services with a Â£254m turnover, 16,000 customers and 2,200 employees. Advanced provides enterprise and market-focused solutions that allow customers to reimagine what is possible, innovate in their sectors and improve the lives of millions of people in the UK.

AWS Migration à The Trigger For CD

Advanced, like many other organizations, are migrating their traditional on-premise applications to the public cloud à more specifically AWS. With over 100 different products/teams and ~700 software engineers, an internal CI/CD initiative was kicked off earlier this year to empower all development teams with a self-service Continuous Delivery platform to help them increase velocity and reduce time-to-market.

Ad-hoc Continuous Delivery Didn't Scale

In the past, a typical deployment pipeline at Advanced would take months to build using a combination of Jenkins, AWS CloudFormation, Octopus Deploy, Puppet, SSM and Bash scripts. à Pipelines required several thousand lines of Puppet code to be maintained by DevOps engineers and teams,â said Martin Reynolds, Development and DevOps manager at Advanced.

In addition, these pipelines could take anywhere from 3 to 30 hours to execute with many manual interventions. Deployment health checks and verifications were also manual, ranging from 2 engineers for 2 hours for minor releases to 4-5 engineers for 1 day for major releases.

âIt was difficult to make Continuous Delivery scale with ad-hoc pipelinesâ said Martin. âThatâs when we decided to kick off a new CI/CD initiative internally, with the goal of making Continuous Delivery self-service for all our developers.â

It was at this point that Martins team evaluated the Harness platform.

CD as-a-Service With Harness

Today, Harness helps Advanced automate software delivery across 100+ product teams, 700+ software engineers, 6 data centers and several technology stacks/tools that include:

- AWS EC2, ECS, EKS and Lambda for public cloud
- Jenkins, GitHub, and JFrog Artifactory for Continuous Integration (CI)
- AppDynamics, CloudWatch, and Elastic/ELK Stack for monitoring

âJenkins builds it, Harness deploys it,â said Martin, âHarness makes our deployments easy.â

Jenkins builds it, Harness deploys it. Harness makes our deployments easy.

Harness provides Advanced software engineers and DevOps teams with the following capabilities:

- Intuitive Self-Service CD Platform to accelerate onboarding across teams
- Fully automated Canary deployments to reduce the risk of rollouts
- Auto-verification of deployments using machine learning to baseline performance and quality data from AppDynamics, CloudWatch, and Elastic/ELK stack
- Auto-rollback of deployments that fail performance/quality baselines
- Templatization of deployment pipelines by DevOps
- Re-use of pipeline templates by development teams

Upgrading Jenkins With Harness

Not only is Harness automating the delivery of software inside Advanced, but it also helps automate the upgrade of tooling like Jenkins.

For example, the DevOps team at Advanced would spend 2-3 hours a week upgrading Jenkins with patches and upgrades. With Harness, this process is automated and takes 15-20 minutes, saving the DevOps engineers a few hours a week.

10X ROI in Just 2 Months

The first 10 application teams and 100 services were onboarded with Harness in just 2 months, many of which were enabled on day one.

Deployment pipelines now take DevOps engineers a few hours to create/templatize with virtually no maintenance versus several months with ongoing maintenance. Developers reuse these pipeline templates for their specific services and environments without having to re-create the wheel every time, thus reducing their effort and time-to-market.

âWe saw immediate results,â said Martin. âOne of our education teams saw deployment time drop from 3-hours to 1-minute across 19 services; another team saw deployment time drop from 30 hours to under 30 minutes.â

Overall, Harness has helped Advanced reduce their average deployment time by 88% from 2 days to 2 hours, with a 90% reduction in onboarding time for dev teams from 2 days to 90 minutes. âWeâve conservatively saved 50-60% of total DevOps and engineering time spent on deployments and our previous CI/CD process,â said Martin.

Download our eBook

Taking a look at where major DevOps trends are headed, a common theme across many tools and practices is improving the Developer Experience.

Request a Demo

Request your personalized demo of Harness, The Modern Software Delivery platform.

Explore Related Content

Sensormatic optimizes retail operations with Harness SEI

Citi Improves Software Delivery Performance, Reduces Toil With Harness CD

Citi and Harness transformed the software delivery process for Citi's 20,000 engineers, slashing release times from days to 7 minutes.

Synopsys Boosts Cloud Visibility and Cost Control with Harness Cloud Cost Management

Harness helped Synopsys rein in drastic cloud spend increase and gain visibility into cloud infrastructure resourcing.

Vivun Scales DevOps With Harness

Harness enabled Vivun to evolve its DevOps strategy to achieve an on-demand release cadency and 300% improvement in DevOps engineer efficiency.

The Modern Software Delivery Platform®

Need more info? Contact Sales

Source URL: <https://www.harness.io/legal/service-packages-custom-resource-appendix>

Custom Resources

As of Jan 12, 2024

Harness Custom Resource types, by product

MODULE	RESOURCE NAME
Platform	Custom Secret Manager Connector
Platform	Custom Governance Policy
Platform	Custom SSO Provider
Platform	Custom Approval Stage
Continuous Delivery	Deployment Template
Continuous Delivery	Custom Health Source
Continuous Delivery	Custom Artifact Repository
Continuous Delivery	Custom Remote Manifest
Continuous Delivery	Custom Webhook Trigger
Continuous Delivery	Custom Approval Step
Service Reliability Engineering	Custom Health Source
Service Reliability Engineering	Custom Change Source
Security Testing Orchestration	Custom Ingest
All Modules	Custom Dashboards

Source URL: <https://www.harness.io/legal/service-packages-accelerator-standard>

Accelerator (Standard)

As of Jan 15, 2024

This Statement of Work (this "SOW") is entered into and governed by the Master Subscription Agreement, Subscription Terms, or other similar written agreement (the "Agreement") by and between Customer ("Customer"), and Harness Inc. ("Harness") (Customer and Harness each a "Party", and collectively, the "Parties"). This SOW and the Agreement constitute the complete agreement regarding services provided under this SOW. Except where the contrary is expressly provided, the terms and conditions of the Agreement shall prevail over any conflicting terms or conditions in this SOW.

Changes to this SOW will be processed in accordance with the procedure described below. The investigation and the implementation of changes may result in modifications to the schedule, resources, charges and/or other terms of this SOW.

Harness may utilize an employee or subcontractor (a "Partner") to provide the Services under this SOW. Harness shall remain liable or all acts and omissions of its personnel in their provision of the Services set forth in this SOW.

Any defined terms not specifically defined herein shall have the meaning given to them in the Agreement.

1. A DEFINITIONS

"Business Day" means any weekday, Monday through Friday, for 8 hours, generally falling between 9:00 A.M. and 5:00 P.M. in the time zone of the Harness implementation engineering team, excluding holidays and weekends, unless otherwise agreed in writing by the Parties.

"Business Week" means 5 Business Days, Monday through Friday.

"Harness Reference Architecture" means a predefined, example implementation and accompanying documentation that outlines the ideal use of Harness products, as provided and updated by Harness from time to time.

"Engagement Plan" means a project plan created by Harness after the Design & Discovery Sprint, detailing the architecture and design for the applicable software module, key project details such as milestone dates, timelines, project prerequisites, and any potential issues that may jeopardize the project timeline. This document will be used to guide the completion of this SOW.

"Project Summary" means a summary of the work completed under this SOW, which will be created by Harness upon completion of the Sprints.

2. PROJECT OVERVIEW

The purpose of this Accelerator (Standard) SOW is to provide Services designed to assist Customer in deploying the applicable Harness software module, based on best practice, reference architecture and an appropriate use case (the "Pilot Use Case") for a single Customer team (the "Pilot Team"). The use case selected must be (i) achievable within the project timeline, (ii) derive a first value for the customer, and (iii) Customer must be committed to deploying the use case during an agreed upon number of sprints (each, a "Sprint"). The standard timeline for the entire SOW is set forth below, in Business Weeks (the "Standard Project Duration"), and will consist of a series of Sprints, each with their own goal, as specified below. Each project begins with a Design Sprint and a Discovery Sprint, followed by one or more Implementation Sprints.

Harness will provide a team of service professionals, including a Solution Architect (the "SA"), and Customer Success Manager (the "CSM") who will work with the Customer for the Sprints, to implement and deliver the project as described in this SOW.

Harness will schedule a series of working sessions with the Customer during each Sprint. The goal of these working sessions is to provide an opportunity for live collaboration between the Parties' engineering teams. Each working session will focus on completing tasks for the current Sprint (as defined in the Engagement Plan) (detailed below), and will not exceed three (3) hours in duration per Business Day. A minimum of three (3) working sessions each Business Week is recommended. The Customer's engineer(s) should expect to perform follow-on tasks and actions from the live working sessions, and these action items will need to be completed prior to the next live session as part of the overall Sprint in order for the Sprint to move forward.

Customer shall schedule and commit its team members as appropriate to meet the Sprint schedule, and to support Harness in its delivery of the project pursuant to the Engagement Plan. Customer's commitment of its team members is essential to completing the Sprints in a timely manner.

The Accelerator (Standard) SOW is designed to provide Customer with a high level configuration of the Pilot Use Case, modeled after a Harness Reference Architecture. Upon completion of the project, engagement details will be documented in the Engagement Document for the customer to extend to teams beyond the pilot groups as part of the Accelerator implementation plan.

3. STANDARD TIMELINE; SCOPE OF PROJECT; SPRINT DETAILS

Timeline for Accelerator (Standard) Sprints

Sprint Name	Duration	Activities	Outcomes
Discovery & Design Sprint	1-2 Business Weeks	3 hour planning session 3 per Business Week	Create Project Plan Begin Platform Fundamentals Scope (as applicable)
Implementation Sprint 1	2 Consecutive Business Weeks	2 hour working session 4-5 per Business Week	Complete Platform Fundamentals Scope (as applicable) Begin Module Accelerator Scope
Implementation Sprint 2	1-2 Consecutive Business Weeks	2 hour working session 4-5 per Business Week	Complete Module Accelerator Scope

Scope of Project and Accelerator Kickoff

The procedures set forth below and in the appendices (as referenced) apply to the duration of this Accelerator (Standard) SOW, for the implementation and onboarding of a single Harness software module (the âProductâ). This SOW includes (A) Platform Fundamentals Scope, and (B) Module Accelerator Scope, for a single Product, for the agreed upon Pilot Use Case.

The SOW shall begin with a kickoff meeting, known as the âAccelerator Kickoffâ, which will be scheduled by the Harness team taking into account Customerâs preferred schedule. The intent of the Accelerator Kickoff is to introduce and align stakeholders, decision makers, engineers and the Harness team to ensure clear understanding of the process to be followed and scope of this Accelerator (Standard) SOW. The Accelerator Kickoff is also intended to determine any onsite logistics (if necessary), key technical resources needed from the Customer, and general ways of working during the engagement. The outcome of this meeting should result in the scheduling of the Discovery & Design Sprint.

Sprint Details

Discovery & Design Sprint

The purpose of the Discovery & Design Sprint is to review all the technical implementation planning required to implement and onboard the Product within Customerâs environment, to achieve the Pilot Use Case. During this Sprint, the Parties will also review the Product configuration requirements, refine the scope of and agree to the dates and staffing requirements for the Implementation Sprint(s). The outcome of this Discovery & Design Sprint are to ensure the Customer and Harness align the project schedules, goals, and scope of this SOW, and to produce the Engagement Plan, which will contain all such details. The Discovery & Design Sprint must be completed prior to starting the Implementation Sprints. This Sprint will cover the following, as applicable:

Discovery:

- Understand the Customer environment in which the Product will be utilized
- Identify Product goals, which can include:
 - â Details about the Customerâs legacy solutions being replaced/enhanced by the Product
 - â KPI metrics for project success
 - â Product utilization infrastructure
 - â Disaster recovery requirements
 - â Product logging
 - â Pilot Use Case details
 - â The end-users on Pilot Team
- Identify Product (and Harness Platform) configuration goals, which can include:
 - â Single Sign On for user management
 - â RBAC Persona requirements
 - â Security Compliance requirements and stakeholders
 - â System/Network requirements and stakeholders
- Identify Product adoption goals, which can include:
 - â Mapping of Customer organization structure / business unit hierarchy to Product account hierarchy
 - â Automated onboarding / propagation
 - â End-user experience standards
 - â UI, CLI, and/or code commit

Design:

- Review Product implementation designs based on reference architectures and recommended practices
- Walk through automated Product deployment methodology
- Understand how the Product will be implemented within the Customer environment (non-production)
- Establish the Product implementation goals, which can include:
 - â Zero-trust, low-trust, or high-trust security compliance
 - â Self-service onboarding automation
 - â Templatization use requirement standards
 - â Governance policy outcomes
 - â Notification and alert postures
- Identify prerequisites and/or dependencies to be addressed before Implementation Sprints can begin, this may include factors such as:
 - â Networking rules for ingress and egress
 - â Required security exceptions
 - â Delegate Architecture topology and requirements
 - â Systems in-scope for Product integration
- Requirements for Product-specific implementation that will be included in this Accelerator (Standard) SOW are set forth in the

relevant Module Appendix. Â

Technical Implementation Planning: To be completed during the Discovery & Design Sprint

- Identify potential adoption use case(s)
- Refine and prioritize project scope
- Outline implementation prerequisites
- Identify project team members by role, organization (Harness, Customer, or Partner, and business unit)
- Determine agreed upon dates for the Implementation Sprints
- Collect onsite visit logistics (if applicable)
- Document Project details in the Engagement Plan

Sprints Deliverable: PROJECT PLAN

- Delivered within one Business Week of the completion of the Sprint activities described above.
- The Engagement Plan may contain the following, as appropriate:
 - â A high level installation architecture and design for the Product
 - â The software and staff prerequisites to support the Project
 - â Project dates and timelines, defined by sprint outlining prioritized tasks and deliverables
 - â Will highlight risks/issues and/or blockers that put the timelines at risk
 - â Prerequisites / tasks to be completed by the Customer before Implementation Sprints can begin, including providing the required people that must be available during the Sprint timelines.
- The Engagement Plan will be available via the Harness Professional Services Portal (a link to this Portal will be provided) for Customer to view and track Customerâs participation throughout the term of this SOW.

*Acceptance Criteria for Deliverables; Acceptance Process:*Â

- The Customer will have three (3) days to review the Engagement Plan and provide approval.Â If the Customer rejects any aspect of the Engagement Plan, the Parties will work together to address the issue(s) and an updated Engagement Plan will be provided for review and Customer will have an additional three (3) days to review the revised Engagement Plan.Â This approval cycle will continue up to (2) times, or until the Engagement Plan has been approved by Customer, whichever comes first.Â
- If Customer does not provide an affirmative approval or rejection of the Engagement Plan during the 3 day review period, or after the second approval cycle, the Engagement Plan shall be deemed automatically approved.Â
- Implementation Sprints will not be scheduled until the Engagement Plan is approved. Once approval is provided, the Implementation Sprints will be scheduled. Harness requires up to 3 Business Days to finalize the scheduling for the Implementation Sprints.Â

Implementation Sprint(s):

The purpose of the Implementation Sprints is to ensure the Engagement Plan as documented and agreed upon in Discovery & Design Sprint is delivered. The Implementation Sprint timelines are defined and prioritized in the Engagement Plan.

Customer Prerequisites

- Customer agrees that the following will be completed prior to the start of Implementation Sprints:
 - â Review and approval of the Engagement Plan
 - â Completion of implementation prerequisites set forth in the Engagement Plan

Platform Setup

- Single Sign-on integration with one (1) supported Identity Provider
- Implement the Pilot Account Hierarchy
- Configure Persona-based RBAC
- Secret Manager integration with one (1) supported Encrypted Secret Storage Provider

Single Module Pilot Implementation

- Implementation of the Product based on the architecture and design outlined in the Engagement Plan, and based on the best practice scope relevant to the applicable Product (as outlined in the relevant Module Appendix).

Customer Validation

- Review implementation with Customer administrators / architects
- Review implementation with Pilot Team
- Successful production-grade Product implementation for Pilot team, with verification of success via KPIs agreed upon in Engagement Plan

Self-service Automation

- *Implement the self-service automation capabilities (or other product-specific resource automation capability) specifically outlined during the technical implementation planning phase of the Discovery & Design Sprint.Â Harness will make reasonable efforts to incorporate any optimizations discovered during the implementation of the Pilot Use Case and any Customer feedback.*

Calibrate for Adoption

- A phase of activities that include:
 - â User Training & Enablement
 - â Self-service user Onboarding
 - â Automated Org and/or Project Onboarding
 - â Automated Resource Configuration
 - â Adoption of pre-configured templates

Deliverables

- Product implementation is complete and accessible by the Customer (in a non-production environment).
- Knowledge transfer of the services rendered as part of this SOW.
- Creation of Project Summary:
 - â Customer specific notes about installation, configuration, and adoption of the Product
 - â Important findings during implementation
 - â Additional recommendations the Customer may wish to perform after the engagement (which not in scope for this SOW)

4.Â RESPONSIBILITIES; LIMITATIONS; SCHEDULING; OWNERSHIP

Customer Responsibilities:

Customer acknowledges and agrees that Harnessâs ability to deliver the Services is dependent uponÂ Customerâs full and timely cooperation with Harness, as well as the accuracy and completeness of any informationÂ and data Customer provides to Harness. Customer is responsible for delays and any additional costs caused byÂ Customerâs failure to comply with their responsibilities. Harness will not be responsible for any resulting fees orÂ expenses.

- Customer will complete their required prerequisites set forth in the Engagement Plan.Â These prerequisites may include, but are not limited to:
 - â Permissions to manage cloud/on-premise infrastructureÂ
 - â DNS entriesÂ
 - â TLS Certificate and related keys
 - â Cloud/on-prem network and/or proxy configuration, if needed
 - â Airgap systems, if neededÂ
- Customer will provide all requested contact information for the Customerâs key business leaders and stakeholders required for Harness to provide the Services
- Customer will coordinate communications with, and provide contacts for, all third Parties necessary for the completion of the Engagement Plan
- Customer will deliver timely responses to requests for information, technical questions and requested decisions pertaining to the project and this SOW. Harness shall not be responsible for any delays in the provision of Services due to Customerâs non-compliance with this requirement
- Customer will diligently manage and supervise Customerâs other contractors and service providers as necessary to support the Services
- Customer will coordinate Service implementation on third-Party-maintained hardware/software (if applicable) with Harness
- Customer will assign a designated Subject Matter Expert coordinator (SME) from Customerâs staff who, on its behalf, will grant all approvals, provide information, make decisions, attend meetings, and otherwise be available to assist Harness in facilitating the delivery of the Services
- Customer will assign a designated engineer and appropriate supporting staff to implement the Engagement Plan
- Customer will assign a designated project manager to facilitate on time delivery of Engagement Plan
- Customer will ensure the availability of all hardware, firmware, and software required by the SA/SEN to deliver the Services and provide network access to integrate required tooling and systems with the Product including (but not limited to) Artifact Repositories, Version Control Systems, Identity Providers, Log Aggregators, Security Scanning Tools, Monitoring Systems, and Developer Tool platforms
- Customer will provide reasonable access and working space at the site as Harness may request

â

Harness Responsibilities:

Harness is responsible for providing the Services outlined in Section 3 above. Where Harness is performing Services on-site for a Customer, Harness shall ensure that its personnel conduct themselves in accordance with the standard health, safety, and security policies of Customer applicable to its staff and or visitors generally, as long as such policies are provided to Harness in writing, in advance. Customer agrees to provide written notice to Harness of any applicable non-standard policies (for example, the requirement of security clearance) prior to executing an Order Form for the Services.

â

Limitations/Exclusions:

Customer acknowledges and agrees that following limitations apply to this SOW:

- The Product implementation will be completed using only approved methods by Harness for generally available versions of the Product
 - â Terraform will be used for automated deployment of the Product when deploying to the commercial offerings of Azure, AWS, or GCP, and manual otherwise
 - â The Product implementation will be modeled after Harness Reference Architecture and Harness best-practices
- The Module Accelerator Scope will be scoped to a single Pilot Use Case
- The single module Product implementation will be limited to one (1) Pilot Team
- Single Sign On configuration leveraging a non-tested Identity Management Platform that supports the SAML 2.0 standard can only be accommodated on a âbest effortâ basis and is not guaranteed
- Customer must make available its Identity Management Platform administrator to make necessary changes outside the Product when configuring Single Sign On
- The Product implementation in scope for this engagement will be âproduction-gradeâ and maintain a sightline to a Production implementation, BUT it is the Customer's responsibility and prerogative to promote the Pilot Use Case from a non-production environment to the Customerâs production environment.
- Governance Policies delivered as part of this engagement will be sourced from out-of-the-box examples, and any needed modifications can be accommodated on a best effort basis only and are not guaranteed.
- Customer is responsible for any system integrations that are not supported by out-of-the-box integrations. Â
- Knowledge transfer sessions will be limited to no more than two (2) sessions, totaling no more than four (4) hours in the aggregate, over the entire term of this SOW.
- The Services will be delivered as a single, continuous workstream, as set forth in the Engagement Plan.Â Environments requiring multiple engagements, phases, or timelines that exceed the standard timelines are excluded from the scope of this SOW.Â If Customer requires additional scope, the Parties must execute a Change Order or enter into a separate Order Form and Statement of Work.
- Recording of Meetings and Working Sprints is not allowed unless agreed and documented in the Engagement Plant.
- All services are provided remotely unless pre-approved travel arrangements are made with the Customer, subject to the expense reimbursement guidelines set forth in Section Â 5 below.
- Any implementation of recommendations provided by Harness, such as system configuration, performance tuning, or database design improvements, which are not recommendations for the Product itself, are excluded.
- Once the Implementation Sprint is completed, Customer is responsible for maintaining any changes or improvements to any and all code artifacts, automation work, and/or documentation delivered under this SOW, including the continued support and maintenance of the foregoing.
- The following are expressly excluded from the scope of Services provided under this SOW: (i) hardware maintenance and repair; (ii) software maintenance and upgrades, (iii) instructor led training, (iv) operational runbooks or execution plans.
- Harness standard maintenance and support services are provided in accordance with Harnessâ maintenance and support policy, and such services are not part of this SOW.Â
- The Services provided under this SOW are specific to the designated Product licensed by Customer only, and do not include any consultation, guidance, or support for any other software, solutions or materials.

â

Scheduling/Governance of Sprints:

- A Sprint duration is defined as a minimum of one (1) Business Week, and a maximum of two (2) consecutive Business Weeks
- Each Sprint will align with a Business calendar week
- Time between Sprints cannot exceed more than two (2) Business Weeks
- Once a Sprint has started, the Customer is committed to the scope, deliverables and timeline, and the Sprint may not be suspended or delayed.
 - â If a Sprint is suspended or delayed due to the acts or omissions of Customer or its personnel, then the remaining work for that Sprint will be canceled, if it is the final Sprint that is being suspended. If desired, Customer can request a Change Order for the outstanding work.
 - â If the canceled Sprint is not the final Sprint, the remaining scope of work under that Sprint will be moved to the next scheduled Sprint as the first and priority tasks.Â
 - â The remaining Sprint scope will be worked on a best effort basis, keeping the originally scheduled Sprint dates.Â
 - â If the requested suspension or delay impacts the timeline of the remainder of the Engagement Plan, Customer may re-engage and reschedule the Sprint scope for the next available Sprint, working with the Harness project manager or CSM to determine the appropriate timeline given the availability of Harness resources. If there are no remaining Sprints under this SOW, Customer must request a Change Order for the additional work requested.
- If Customer needs to delay a future Sprint and deviate from the Engagement Plan, Customer shall notify their project manager or CSM in writing at least five (5) business days prior to the agreed upon start date of the Sprint.Â If Customer does not provide such notice, the Sprint will automatically be canceled and all services thereunder will be forfeited by the Customer.
 - â After the delay of a Sprint, Customer must make a written request to Harness to re-engage.Â Such request must be made at least fifteen (15) Business Days prior to the new requested Start Date.Â Harness will work with Customer to determine the new Start Date, based on Harness resource availability. The new agreed upon Start Date and any other appropriate updates will be made to the Engagement Plan. The updated Engagement Plan must be approved by Customer in accordance with the approval process outlined above.Â Once the Engagement Plan has been approved, the Sprint will officially be rescheduled.

â

Ownership:Â

As between Customer and Harness, Harness retains all right, title and interest in the services provided and any results of thereof including the deliverables specified in this SOW (collectively, "Deliverables"). Subject to Customer's payment in full of all fees due under this SOW, Harness grants Customer a limited, non-exclusive, non-transferable and non-sublicensable right and license to use the Deliverables during the applicable Subscription Term in connection with Customer's use of the applicable Products in accordance with the Agreement. For the avoidance of doubt, Deliverables do not include any Customer Confidential Information, and any Customer Confidential Information contained in the Deliverables shall remain the sole and exclusive property of Customer.

5. FEES AND EXPENSES

Customer shall pay to Harness the fees set forth in the applicable Order Form for the services. All fees are due up front, in full, and must be paid within 30 days of the invoice date. The payment of all fees is subject to the terms and conditions set forth in the Agreement.

If the work under this SOW exceeds the Standard Project Duration, or if Customer requests or requires additional hours beyond the scope of this SOW, the Parties shall execute a Change Order to this SOW as described in the Change Management section below.

Any Services performed outside of the Standard Project Duration or any additional hours of work will be provided by Harness on a time-and-materials (T&M) basis only, at an hourly rate of \$325 per hour, unless otherwise set forth in the Change Order. Customer must order a minimum of 10 hours per Business Week under such Change Order, and all additional hours purchased must be utilized within 6 months of the date of the Change Order, or else they will be forfeited. Any work provided on a T&M basis will be invoiced by Harness monthly in arrears. Once the Parties execute a Change Order for additional hours, the fees for such hours are non-cancellable and non-refundable. At the end of the six (6) month period, Harness shall invoice Customer for any remaining hours purchased.

Furthermore, if Customer changes any requirements, additional hours may be required to complete the Services, and the Parties will execute a Change Order to facilitate the change in scope. Such a Change Order may increase the Price of the Services. No additional Services shall be provided unless the Change Order is agreed to in writing by the Parties.

Expenses are not included in the fee set forth above. All expenses incurred shall be in accordance with the Customer's T&E corporate policy if provided to Harness in writing in advanced, otherwise such expenses shall be incurred in accordance with Harness's T&E policy. If travel expenses are incurred with respect to this Professional Services program, Harness will invoice for travel expenses as incurred and monthly in arrears and Customer shall pay such invoices in accordance with the terms and conditions set forth in the Agreement. For the avoidance of doubt, all travel expenses must be pre-approved by the customer prior to any trips as part of this SOW. The duration of visits requested by the Customer cannot exceed two (2) business days unless mutually agreed in writing.

6. TERM AND TERMINATION

This SOW shall terminate upon the earlier of (i) six (6) months from the date of the last signature, below, or (ii) completion of the Services described above. Customer may request to extend the termination date of this SOW by submitting a Change Request as described in the Change Management section below. If the Change Request is accepted, a reinstatement fee of up to 10% may apply.

Upon the expiration or termination of the Agreement or this SOW pursuant to the Agreement, all amounts (including expenses) owed to Harness for Services under this SOW (whether completed or not), will be immediately due and payable in full. Upon termination of the Agreement or a SOW by Customer for any reason pursuant to the Agreement, Customer shall be responsible for payment of all fees for Services rendered and expenses incurred prior to the date of termination. In addition, upon any termination or expiration of the Agreement or this SOW, Harness's obligation to provide Services shall immediately terminate.

7. CHANGE ORDERS

Unless otherwise mutually agreed by the Parties, any change or modification of this SOW or the Engagement Plan will be coordinated by the Parties in accordance with this Section 6. Either Party may initiate such requests for a change or modification (each, a "Change Request"). The Party requesting a Change Request will submit a written Change Request to the other Party in a clear and concise manner using a form substantially similar to the sample document attached hereto as Annex 1 (a "Change Order"). Upon the execution of a mutually agreed upon Change Order by both Parties, the obligations of the Parties with respect to such Change Request will be incorporated under the SOW. If a Change Order reinstates an expired SOW, a reinstatement fee of up to 10% may apply.

Source URL: https://www.harness.io/blog/targeted-rollouts-release-progression?utm_source=pmm-ff&utm_medium=safe-testing-in-production-with-harness-feature-flags

Targeted Rollouts and Release Progression

When you want to try a new Harness feature out, we enable it for you via a feature flag. Our support/account team turns it on for you. Typically, for new features, our release process happens in multiple phases. First, we test internally in lower environments. Second, we test internally in production. Third, we test with a few beta customers in production for a period of time leading to a wider release. Finally, for some features, we release more widely - but still in beta, and often still with some restrictions on who can access them. This is all an example of using targeted rollouts via feature flags, and then progressing releases over time as you learn and respond. We will dig into that further in this blog.

Why Would I Want to Do That?

Feature flags are often thought of as being on the continuum of CI/CD, and there's good reason for that. Feature flags de-risk your deployments, let you merge and ship faster, and reduce the need for long-running feature branches. In this way, feature flags level off the classic goals and outcomes of CI/CD.

However, feature flags are not only an extension of CI/CD. They are also a separate process that sits *alongside* CI/CD, enabling new behaviors or *removing* those behaviors from the CI/CD loop entirely. This simplifies CI/CD and also provides you and your customers with new and better release options.Â

When thinking of feature flags this way (as a process that runs alongside CI/CD rather than one that sits on top of it), targeted releases and release progressions are high-value behaviors that immediately emerge.

Targeted releases, specifically, mean turning your change on for a specific subset of your audience initially, and then progressing it to more users if/when you're ready. Or, alternatively, turning it on for this subset and then turning it off right away if there are issues.Â This method is especially useful for reducing the amount of project management needed, where you might want to incrementally introduce changes and measure their effects with very low overhead.

Using feature flags like this allows you to stress test changes, have PMs or other folks drive alpha and beta feature feedback, control releases by customer level or region, and work closely with design partners. All without needing to rely on engineering day-to-day for a deploy/rollback cycle. Or to enable/disable things for the targeted users.

How Is it Different Than a Canary Deployment?

We previously wrote about how features flags and canaries should be thought of as working together, not an either/or. But in the context of targeted releases, it's worth revisiting the differences between feature flags and canaries.

Canaries are a fantastic way to *test the code or the artifact* for system anomalies, performance issues, scale, and other similar metrics - and roll back as needed. However, feature flags can help you *test the change, not just the code*.

What does *testing the change and not just the code* mean, specifically? It means that the code working as intended doesn't mean the feature is working for users. Does it do what we need for our customers? Can they find it? Can they understand it? What questions do they have? Avoiding a type 2 error---where you mistakenly accept something as fine when it's not---is key.

If we're in beta of a feature and we're trying to prioritize the next batch of work in the most useful order, having this kind of feedback guiding what we do next and what has the highest impact is critical. It's not about "Does the code technically work?"Â

This is an example of where we want to think about feature flags not as an extension of CI/CD, where the point is simply releasing the change without risk, but rather as a standalone part of our software development process where we are constantly enabling and disabling things to get feedback, learn, and respond while we are building and deploying every day.

Internal Examples

I've already talked a little bit about how we use Harness Feature Flags internally, to turn things on for our customers. That's one example of how we use targeted rollouts here at Harness. Though, it is one that often makes Slack messages difficult to parse since sometimes we use Harness Feature Flags to turn on a feature inside of Harness Feature Flags for Harness Feature Flag customers (whew!).

There are some other great ways we use targeted releases, though.

- We may turn a feature off for all on-prem customers, but not have to worry about maintaining more complex conditional code or separate builds.
- We can turn something on for our Enterprise users without yet having it wired to the account framework that enforces plan limits, which lets us learn more about the feature earlier in the process.
- If a feature is early in development, we may give access selectively to our sales engineering team so that they can work with partner customers in a hands-on way to demo the feature, get feedback, or drive conversation around where we are going.

And that's just how we use targeted rollouts. We've heard of plenty of other teams that turn things on and off per region, to test performance or scale. We've also heard of customers turning things on and off for specific infrastructure or clusters, to help with migrations or to get data about possible configurations. There's a lot you can do that isn't immediately obvious.Â

Conclusion

Using feature flags for targeted rollouts is a powerful way to expand how your organization builds software, learns from customers, and de-risks changes along the way.

Most organizations, when starting with feature flags, have the idea in mind that feature flags require the "right" use cases to be useful. They often struggle to know where to start. However, thinking of targeted releases helps explain that the more things are behind flags by default, the more options you'll have in the future.

You can't always predict what, when, and where you will want to target. The mindset of targeted releases is about knowing you have the capability to do so, so that as scenarios, questions or needs come up, your organization is able to learn and react.

Want to keep reading about feature flags? Why not check out our pieces on Kill Switches and How To Get Started With Feature Flags?

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/chaos-carnival-2023-recap>

What We Learned at Chaos Carnival 2023

We're thrilled to report that our third annual chaos engineering conference, Chaos Carnival 2023, was a success. With over 1,800 attendees worldwide, Harness brought chaos engineering practitioners and leaders together to explore the latest trends and best practices.

The theme for the 2023 conference was "Build Resilience Through Chaos," which highlighted the importance of testing and proactively improving systems to prevent failures and enable innovation. The conference brought together a diverse group of attendees â including CTOs, engineers, developers, industry experts, and analysts â to share their chaos engineering experiences and ideas.

Over two full days, attendees had the opportunity to participate in 25+ interactive talks, breakout sessions, technical workshops, and panel discussions. The event featured four keynote presentations from some of the most prominent thought leaders in the industry, including:

- **Adrian Hornsby**, Principal System Dev Engineer, AWS
- **Uma Mukkara**, Head of Chaos Engineering, Harness
- **Adrian Cockcroft**, Tech Advisor, Harness
- **Karthik Satchitanand**, Principal Engineer, Harness

Key Takeaways From Chaos Carnival 2023

One of the most important lessons learned at Chaos Carnival 2023 is the critical role that chaos engineering plays in enabling developers to focus on innovation rather than the distractions of system failures. By building resilient systems that can withstand unexpected failures and disruptions, organizations can free up resources and energy for more innovative pursuits.Â

Here are some other top takeaways:

- Chaos engineering is experiencing exponential growth as more and more organizations recognize the importance of building resilient systems that can withstand unexpected failures and disruptions.
- Documenting Chaos Engineering exercises is essential to capture best practices in your organization.

- Shift-left reliability and resiliency testing are needed across the entire software delivery lifecycle, from developers to site reliability engineers (SRE).
- Continuous resilience (introducing fault injections in the developer continuous integration and delivery (CI/CD) pipeline) is critical for enterprises.
- The emergence of new tools and techniques that are helping organizations test and proactively improve their systems without compromising reliability or resilience.
- Security chaos engineering allows users to test their security posture by running experiments that validate or invalidate security control access.
- Community and collaboration in the chaos engineering space are invaluable.

Continuing the Chaos Conversations and Learning

We encourage all attendees and readers to continue learning by exploring techniques discussed at Chaos Carnival, joining the vibrant chaos engineering slack community, and sharing their expertise with colleagues.

Community contributions and feedback play an important role in successful chaos engineering, particularly for cloud-native ecosystems. Continue the chaos conversation by connecting with the sponsors, keynote speakers, and attendees on LinkedIn, Twitter, and Slack. Networking allows us all to advance the chaos industry, learn about future events, and share thoughts with this community that is passionate for building resilience.

We want to thank all of the attendees, speakers, sponsors, and organizers who made Chaos Carnival 2023 such a success. We look forward to seeing you all at future events and continuing the conversation about how we can build resilient systems that enable innovation and growth.

Special thanks to our sponsors, Nagarro, Blameless, Grafana Labs, k6, Civo, Conf42, the Cloud Native Computing Foundation (CNCF), Elastic, Litmus Community, Poools, Solid.jobs, and DevIT.

Want to learn more about how Harness builds resiliency through chaos? Check out a demo or start your free trial today!

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Partner Code of Conduct

As of

- Partner will act in a way that reflects favorably upon Harness and in compliance with all applicable laws and regulations;
- Partner will use only those printed or electronic marketing materials provided by Harness to Partner;
- Partner will not make any statements or representations about the Software or about the scope of Harness's liability that are different from or inconsistent with those made in the Subscription Terms or otherwise authorized by Harness including, without limitation, the scope of indemnification, warranties, or support offered by Harness;Â
- Partner will reasonably cooperate and assist Harness in bringing legal action against any Customer for any activity in violation of the Subscription Terms, at Harness's sole cost and expense;A
- Partner will not issue any press release or make any other public statement regarding the parties' relationship absent Harness's prior written approval;A
- Partner will not make any representations regarding Harness on Harness's behalf;Â
- Partner shall not engage in any deceptive, misleading, illegal, or unethical practices that may be detrimental to Harness, its Software, products, or services;Â
- Partner shall comply with all applicable federal, state, and local laws and regulations while performing its obligations under this Agreement, including without limitation the U.S. Foreign Corrupt Practices Act and any similar laws or regulations in any applicable jurisdiction;
- Partner shall comply with all applicable export and import laws and regulations, including U.S. embargo and sanctions regulations and prohibitions on export.

Source URL: <https://www.harness.io/legal/service-packages-module-appendix>

Module Appendix: Module Specific Scope for Implementation

As of Jan 09, 2024

Accelerator Standard

This part of the Appendix details the scope of the Services to be provided by Harness under the SOW, on a module by module basis. The SOW provides Customer with Services in relation to "Platform Fundamentals" and one module of Customer's choice (subject to Customer licensing access to and use of such module under a valid Order Form). The details below are intended to identify what the scope of Services includes for a particular module.

Platform Fundamentals

Product Scope

- Single-sign On
- Role Based Access Control
- Harness Account Hierarchy
- Secret Manager

Implementation Scope

Resource Type	Resource Count	Dependencies
Single-sign On Integration	1	Supported SAML or OAUTH SSO Provider
Roles	-	Built-in User Roles
User Groups	-	Built-in User Groups
Custom Roles	1-5	Customer-agreed Role scoping
Custom User Groups	1-10	Customer-agreed User Group scoping
Custom Resource Groups	1-3	Customer-agreed Resource Group scoping
Custom Service Accounts	1-3	Customer-agreed Service Account use case and applicable scoping
Harness Organization Hierarchy	1-2	Customer-agreed business unit / organizational unit structure
Harness Project Hierarchy	1-5	Customer-agreed departmental / team structure
Secret Manager Integration	1	Secret Manager Connector
Secret Manager Connector	1	Supported encrypted secret storage platform

Module: Continuous Delivery

Product Scope

- Rolling Deployments

- Deployment Rollback
- Deployment Approvals
- Service manifest / configuration files
- Service artifacts and repositories / registries
- Deployment Environments and Infrastructure
- Change management integration
- Governance policies
- Software delivery metrics dashboard
- Step, Stage and Pipeline Templates
- Delegate dependencies

Implementation Scope

Resource Type	Resource Count	Dependencies
Service	1	SCM Connector Artifact Connector
SCM Connector	1	Supported SCM Platform
Artifact Connector	1	Supported Artifact Repository
Environment(s)	2	Environmental Overrides
Infrastructure Definition	1 per Environment	Deployment Host Connector
Deployment Host Connector	1	Supported Deployment Hosting Platform
Pipeline Template	1	Deployment Stage
Deployment Stage Template	1	Deployment Step Rollback Step
Deployment Step Template	0-2	Service Environment Infrastructure Definition
Harness Approval Step Template	1	Harness RBAC Group(s)
Change Management Step Template	1	Change Management Connector
Change Management Connector	1	Supported Change Management Platform
Governance Policies	2	Built-in OPA Policies
Dashboards	2	Built-in Dashboards

Module: Continuous Integration

Product Scope

- CI Build Farm Architecture
- Codebase
- Code Builds
- Build Cache
- Build Test
- Governance Policies
- Software Build Transparency Dashboard
- Step, Stage and Pipeline Templates
- Delegate dependencies

Implementation Scope

Resource Type	Resource Count	Dependencies
Codebase	1	SCM Connector
SCM Connector	1	Supported SCM Platform
Artifact Connector	1	Supported Artifact Repository
Pipeline Template	1	Build Stage
Build Stage Template	1	Build Step Test Step Push Step Cache Step
Build Step Template	1	Codebase
Test Step Template	1	Codebase
Push Step Template	1	Artifact Connector
Cache Step Template	1	Supported Cache Platform
Harness Approval Stage Template	1	Harness RBAC Group(s)
Governance Policies	2	Built-in OPA Policies
Dashboards	2	Built-in Dashboards

Module: Feature Flags

Product Scope

- SDK Installation and Configuration
- Feature Flags (Boolean and Multivariate)
- Targets and Target Groups
- Environments
- Step, Stage and Pipeline Templates
- Governance Policies
- Feature Flag Transparency Dashboard

Implementation Scope

Resource Type	Resource Count	Dependencies
Feature Flags	1-5	Environments Target Target Groups
Environments	1-4	SDK Key
SDK Key	1-4	Customer-defined Client-side / Server-side Use Case
Target	1-5	Customer-defined Flag Targets
Target Group	1-5	Customer-defined Flag Target Groups
Pipeline Template	1	Feature Flag Stage
Feature Flag Stage Template	1	Flag Configuration Step
Flag Configuration Step Template	1	Feature Flag
Harness Approval Step Template	1	Harness RBAC Group(s)
Change Management Step Template	1	Change Management Connector
Change Management Connector	1	Supported Change Management Platform
Governance Policies	2	Built-in OPA Policies
Dashboards	2	Built-in Dashboards

Module: Security Test Orchestration

Product Scope

- Security Scan Infrastructure
- Integrated Security Scans
- Change Management Integration
- Governance Policies
- Step, Stage and Pipeline Templates
- Delegate Architecture
- Security Scan Dashboards

Implementation Scope

Resource Type	Resource Count	Dependencies
Scanning Infrastructure	1	Delegate Architecture
Delegate Architecture	1	Supported Delegate Hosting Platform
Pipeline Template	1	Security Scan Stage
Security Scan Stage Template	1	Integrated Security Scan Step
Integrated Security Scan Step Template	1-3	Supported Security Scan Platform
Harness Approval Step Template	1	Harness RBAC Group(s)
Governance Policies	2	Built-in OPA Policies
Dashboards	2	Built-in Dashboards

Module: Service Reliability Management

Product Scope

- Monitoring Integration Infrastructure
- Integrated Health Sources
- Integrated Change Sources
- Service Level Objectives and Indicators
- Error Budgets
- Monitored Services
- Change Impact Analysis
- Governance Policies

- Delegate Architecture

Implementation Scope

Resource Type	Resource Count	Dependencies
Monitoring Integration Infrastructure	1	Delegate Architecture
Delegate Architecture	1	Supported Delegate Hosting Platform
SLO	1-2	SLI Error Budget Policy (optional)
Monitored Service	1-3	Monitored Service
SLI	1-2	Service Environment
Health & Change Source Connectors	1-3	Health / Change Source Connector
Integrated Health & Change Source	1-3	Integrated Health & Change Source
Integrated Health & Change Source	1-3	Supported Health and Change Data Systems
Governance Policies	2	Built-in OPA Policies

Module: Cloud Cost Management

Product Scope

- Cloud Billing Ingestion
- Cloud Provider IAM
- Cluster Metric Ingestion
- Delegate Architecture
- Perspectives, Budgets and Anomalies
- Recommendations
- Asset Governance
- Auto-stopping
- Cloud Transparency Dashboard

Implementation Scope

Resource Type	Resource Count	Dependencies	Business Value Unlocked
Cloud Cost Integration	1-5	Cloud Provider Connector	Cost Transparency
Cloud Provider Connector	1-5	Supported Cloud Provider	Unified cost management across multiple cloud providers
Cluster Cost Integration	1-5	Kubernetes Cluster Connector	Cost Visibility and Recommendations
Kubernetes Cluster Connector	1-5	Kubernetes Delegate Architecture	Cost Visibility and Recommendations
Perspective	1-3	Cloud or Cluster Cost Integration	Perspectives, Budgets and Anomalies
Budget	1-3	Cloud or Cluster Cost Integration	Cost Governance
Anomaly	1-3	Cloud or Cluster Cost Integration	Cost Anomaly Detection and Analysis
Recommendation	1 of each type	Cloud or Cluster Cost Integration	Cost Visibility and Recommendations
Auto Stopping Rule	1	Cloud or Cluster Cost Integration	Cost Optimization
Dashboards	1-3	Built-in Dashboards	Cost Visibility and Recommendations
Asset Governance Rule	1-3	Cloud or Cluster Cost Integration	Asset Governance
Asset Governance Enforcement	1-3	Cloud or Cluster Cost Integration	Asset Governance

Module: Chaos Engineering

Product Scope

- Chaos Environments and Infrastructure
- Chaos Experiments
- ChaosHubs
- GameDays

Implementation Scope

Resource Type	Resource Count	Dependencies
SCM Connector	1	Supported SCM Platform
Environment	1-3	Chaos Infrastructure
Chaos Infrastructure	1-3	Supported Infrastructure Provider
Chaos Experiment	1-5	Chaos Infrastructure
ChaosHub	1	SCM ConnectorChaos Experiment
GameDay	1	ChaosHub

Module: Software Engineering Insights

Product Scope

- Tenant Provisioning and Setup
- SSO Configuration
- Issue Management Integration
- SCM Integration
- CICD Integration
- SEI Dashboards
- Sprint Insights
- Asset-based and People-based Teams
- Trellis, Dora, and Business Alignment Profiles

Implementation Scope

Resource Type	Resource Count	Dependencies
Issue Management Integration	1-2	Supported Issue Management System
SCM Integration	1-2	Supported SCM System
CICD Integration	1-2	Supported CICD System
Default DORA Dashboard	1	Issue Management Integration SCM Integration CICD Integration
Default Sprint Insights Dashboard	1	Issue Management Integration
Default Dev Insights Dashboard	1	SCM Integration
Default Business Alignment Dashboard	1	Issue Management Integration
Default Trellis Dashboard	1	Issue Management Integration SCM Integration
Asset-Based Team	1-5	Customer-defined organizational unit
People-based Team	1-5	Customer-defined organizational unit
Team Org Category	1-3	People-based Team Asset-based Team
DORA Profile	1-3	Asset-based Team
Business Alignment Profile	1-3	Asset-based Team
Trellis Profile	1	People-based Team

ACCELERATOR PREMIUM

This part of the Appendix details the scope of the Services to be provided by Harness under the SOW, depending on the Accelerator Premium Bundle purchased by Customer under the applicable Order Form. The SOW provides Customer with Services in relation to either the (1) âDevOps Accelerator Bundleâ, which includes âPlatform Fundamentalsâ, âCD moduleâ, âCI moduleâ and âFF moduleâ; or (2) âDevSecOps Accelerator Bundleâ which includes âPlatform Fundamentalsâ, âCD moduleâ, âCI moduleâ and âSTO moduleâ.

DevOps Accelerator Bundle:

Platform Fundamentals

PF Scope

- Single-sign On
- Role Based Access Control
- Harness Account Hierarchy
- Secret Manager

PF Implementation Scope

Resource Type	Resource Count	Dependencies
---------------	----------------	--------------

Resource Type	Resource Count	Dependencies
Single-sign On Integration	1	Supported SAML or OAUTH SSO Provider
Roles	-	Built-in User Roles
User Groups	-	Built-in User Groups
Custom Roles	1-5	Customer-agreed Role scoping
Custom User Groups	1-10	Customer-agreed User Group scoping
Custom Resource Groups	1-3	Customer-agreed Resource Group scoping
Custom Service Accounts	1-3	Customer-agreed Service Account use case and applicable scoping
Harness Organization Hierarchy	1-2	Customer-agreed business unit / organizational unit structure
Harness Project Hierarchy	1-5	Customer-agreed departmental / team structure
Secret Manager Integration	1	Secret Manager Connector
Secret Manager Connector	1	Supported encrypted secret storage platform

Continuous Delivery

CD Product Scope

- Rolling Deployments
- Deployment Rollback
- Deployment Approvals
- Service manifest / configuration files
- Service artifacts and repositories / registries
- Deployment Environments and Infrastructure
- Change management integration
- Governance policies
- Software delivery metrics dashboard
- Step, Stage and Pipeline Templates
- Delegate dependencies

CD Implementation Scope

Resource Type	Resource Count	Dependencies
Service	1	SCM Connector Artifact Connector
SCM Connector	1	Supported SCM Platform
Artifact Connector	1	Supported Artifact Repository
Environment(s)	2	Environmental Overrides
Infrastructure Definition	1 per Environment	Deployment Host Connector
Deployment Host Connector	1	Supported Deployment Hosting Platform
Pipeline Template	1	Deployment Stage
Deployment Stage Template	1	Deployment Step Rollback Step
Deployment Step Template	0-2	ServiceEnvironmentInfrastructure Definition
Harness Approval Step Template	1	Harness RBAC Group(s)
Change Management Step Template	1	Change Management Connector
Change Management Connector	1	Supported Change Management Platform
Governance Policies	2	Built-in OPA Policies
Dashboards	2	Built-in Dashboards

Continuous Integration

CI Product Scope

- CI Build Farm Architecture
- Codebase
- Code Builds
- Build Cache
- Build Test
- Change Management Integration
- Governance Policies
- Software Build Transparency Dashboard
- Step, Stage and Pipeline Templates
- Delegate dependencies

CI Implementation Scope

Resource Type	Resource Count	Dependencies
Codebase	1	SCM Connector
SCM Connector	1	Supported SCM Platform
Artifact Connector	1	Supported Artifact Repository
Pipeline Template	1	Build Stage
Build Stage Template	1	Build Step Test Step Push Step Cache Step
Build Step Template	1	Codebase
Test Step Template	1	Codebase
Push Step Template	1	Artifact Connector
Cache Step Template	1	Supported Cache Platform
Harness Approval Step Template	1	Harness RBAC Group(s)
Change Management Step Template	1	Change Management Connector
Change Management Connector	1	Supported Change Management Platform
Governance Policies	2	Built-in OPA Policies
Dashboards	2	Built-in Dashboards

Feature Flags

FF Product Scope

- SDK Installation and Configuration
- Feature Flags (Boolean and Multivariate)
- Targets and Target Groups
- Environments
- Step, Stage and Pipeline Templates
- Governance Policies
- Feature Flag Transparency Dashboard

FF Implementation Scope

Resource Type	Resource Count	Dependencies
Feature Flags	1-5	Environments Target Target Groups
Environments	1-4	SDK Key
SDK Key	1-4	Customer-defined Client-side / Server-side Use Case
Target	1-5	Customer-defined Flag Targets
Target Group	1-5	Customer-defined Flag Target Groups
Pipeline Template	1	Feature Flag Stage
Feature Flag Stage Template	1	Flag Configuration Step
Flag Configuration Step Template	1	Feature Flag
Harness Approval Step Template	1	Harness RBAC Group(s)
Change Management Step Template	1	Change Management Connector
Change Management Connector	1	Supported Change Management Platform
Governance Policies	2	Built-in OPA Policies
Dashboards	2	Built-in Dashboards

DevSecOps Accelerator

Platform Fundamentals

PF Scope

- Single-sign On
- Role Based Access Control
- Harness Account Hierarchy
- Secret Manager

â

PF Implementation Scope

Resource Type	Resource Count	Dependencies
Single-sign On Integration	1	Supported SAML or OAUTH SSO Provider
Roles	-	Built-in User Roles
User Groups	-	Built-in User Groups
Custom Roles	1-5	Customer-agreed Role scoping
Custom User Groups	1-10	Customer-agreed User Group scoping
Custom Resource Groups	1-3	Customer-agreed Resource Group scoping
Custom Service Accounts	1-3	Customer-agreed Service Account use case and applicable scoping
Harness Organization Hierarchy	1-2	Customer-agreed business unit / organizational unit structure
Harness Project Hierarchy	1-5	Customer-agreed departmental / team structure
Secret Manager Integration	1	Secret Manager Connector
Secret Manager Connector	1	Supported encrypted secret storage platform

Continuous Delivery

CD Product Scope

- Rolling Deployments
- Deployment Rollback
- Deployment Approvals
- Service manifest / configuration files
- Service artifacts and repositories / registries
- Deployment Environments and Infrastructure
- Change management integration
- Governance policies
- Software delivery metrics dashboard
- Step, Stage and Pipeline Templates
- Delegate dependencies

CD Implementation Scope

Resource Type	Resource Count	Dependencies
Service	1	SCM Connector Artifact Connector
SCM Connector	1	Supported SCM Platform
Artifact Connector	1	Supported Artifact Repository
Environment(s)	2	Environmental Overrides
Infrastructure Definition	1 per Environment	Deployment Host Connector
Deployment Host Connector	1	Supported Deployment Hosting Platform
Pipeline Template	1	Deployment Stage
Deployment Stage Template	1	Deployment Step Rollback Step
Deployment Step Template	0-2	ServiceEnvironmentInfrastructure Definition
Harness Approval Step Template	1	Harness RBAC Group(s)
Change Management Step Template	1	Change Management Connector
Change Management Connector	1	Supported Change Management Platform
Governance Policies	2	Built-in OPA Policies
Dashboards	2	Built-in Dashboards

- Governance Policies
- Software Build Transparency Dashboard
- Step, Stage and Pipeline Templates
- Delegate dependencies

â

Continuous Integration

CI Product Scope

- CI Build Farm Architecture
- Codebase
- Code Builds
- Build Cache
- Build Test
- Change Management Integration

- Governance Policies
- Software Build Transparency Dashboard
- Step, Stage and Pipeline Templates
- Delegate dependencies

CI Implementation Scope

Resource Type	Resource Count	Dependencies
Codebase	1	SCM Connector
SCM Connector	1	Supported SCM Platform
Artifact Connector	1	Supported Artifact Repository
Pipeline Template	1	Build Stage
Build Stage Template	1	Build Step Test Step Push Step Cache Step
Build Step Template	1	Codebase
Test Step Template	1	Codebase
Push Step Template	1	Artifact Connector
Cache Step Template	1	Supported Cache Platform
Harness Approval Step Template	1	Harness RBAC Group(s)
Change Management Step Template	1	Change Management Connector
Change Management Connector	1	Supported Change Management Platform
Governance Policies	2	Built-in OPA Policies
Dashboards	2	Built-in Dashboards

Security Test Orchestration

STO Product Scope

- Security Scan Infrastructure
- Integrated Security Scans
- Change Management Integration
- Governance Policies
- Step, Stage and Pipeline Templates
- Delegate dependencies

STO Implementation Scope

Resource Type	Resource Count	Dependencies
Scanning Infrastructure	1	Delegate Architecture
Delegate Architecture	1	Supported Delegate Hosting Platform
Pipeline Template	1	Security Scan Stage
Security Scan Stage Template	1	Integrated Security Scan Step
Integrated Security Scan Step Template	1-3	Supported Security Scan Platform
Harness Approval Step Template	1	Harness RBAC Group(s)
Change Management Step Template	1	Change Management Connector
Change Management Connector	1	Supported Change Management Platform
Governance Policies	2	Built-in OPA Policies

Source URL: <https://www.harness.io/blog/pitfalls-adopting-internal-developer-portal>

3 Common Pitfalls When Adopting an Internal Developer Portal

Lately, the software industry has been hearing all the benefits that an internal developer portal (IDP) brings as a central component of platform engineering where developers can access tools, services, and information that they need to build and deploy applications. Implemented well, it reduces cognitive load and streamlines workflows for developers.

One of the most popular ways to build an IDP is by using the Backstage library. Backstage is growing rapidly and now has more than thousands of adopters. However, for every successful adoption, there are several others that fail.

After talking to hundreds of companies that are looking to have an IDP, I've noticed several pitfalls in this journey. Let's take a look at some of the common challenges companies face when adopting or building a developer portal to help inform your choices if you are looking to start the journey.

1. High Level of Effort and Maintenance Required

We have seen customers try to manage their developer portal without even assigning a full-time engineer, let alone creating a dedicated team. We've never seen this work well. They end up giving up on the project due to the overwhelming amount of effort required. Looking at a recent Reddit thread, the number one highlighted pitfall is underestimating the amount of effort to maintain Backstage itself.

If you follow Gartner's advice or even look at Backstage docs on strategies for adopting, you'll find that you will need an average of 2.5 engineers (a frontend engineer, a backend engineer, and at least some part-time commitment from a DevOps engineer) to get started with your first few internal teams. Plus, an additional product person always helps the team understand how to prioritize the right problems that can be solved by their internal developer portal.

Even after setting up the team, the amount of knowledge required to maintain an IDP can be significant. Since many available tools are new, they lack appropriate documentation and content to support the needs of maintainers. Even basic monthly upgrades leave people scratching their heads to figure out how exactly they should implement them. However, the payoff for the level of effort needed is huge, with streamlined workflows and less cognitive load on developers vastly improving time to production.

2. Extensive Customization is Essential, Yet Labor Intensive

No two developer portals look the same across companies. Every company has a unique set of challenges, so it's necessary to customize their developer portal. The level of customizations that are needed by teams exceeds what a SaaS platform would usually provide, as it would be very costly for them to maintain them for all users. The list includes:

- User interface (UI) components and views
- Menu and sidebar items
- Ability to pick and choose plugins or write their own
- Updating catalog with components from an existing source of truth
- Apps' theme, branding, and look-and-feel
- Unique workflows (e.g., service onboarding), and the ability to write code in them
- Search across internal tools
- Complex permissions policy around usage, such as role-based access control (RBAC), governance, etc.

Due to such requirements, there are not many providers offering a complete end-to-end IDP experience that meets enterprise needs.

3. Internal Adoption Requires Strategic Evangelism

Adopting an IDP takes time and requires a strategy. By launching a new portal, developers don't always understand why they should start using it alongside the dozens of other portals that they already visit regularly. It takes time for them to alter their ways of working. A good adoption strategy should communicate the goal of improving developer productivity and happiness.

Even in the same organization, development teams face different sets of problems. For example, some people might be interested in observing deployments for their software while others want to observe the latest builds, explore docs, check API dependencies, and so on.

Developers need to see how exactly this new portal fits their use cases and solves their problems. That's why it takes some time to work with them and show how exactly an IDP and its processes can improve their experience.

Join Us to Explore How We Can Advance the IPD Journey

Are you looking to adopt an IDP but unsure of the challenges you may face? As we keep hearing more from our customers, we're exploring ways in which Harness can help with your developer portal journey. If you're interested, let's talk. Reach out to us at idp-interest@harness.io or on our Community Slack (#internal-developer-portal) to discuss more!

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

[Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence](#)

[January 2024 Product Updates](#)

[Sign up now](#)

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

[Get a demo](#)

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/harness-expands-chaos-engineering-resiliency-features-integrated-continuous-delivery>

Harness Expands Chaos Engineering Resiliency Features with Integrated Continuous Delivery

Harness Chaos Engineering (CE) has expanded its chaos fault library and delivered native integration with the Harness Continuous Delivery (CD) module, enabling developers and SREs to test the reliability and resilience of applications in software delivery pipelines to improve velocity and minimize the risk of unplanned downtime.Â

As enterprises increasingly turn to cloud-native applications to keep pace with the speed of innovation and meet customers' expectations for reliability, developers can rapidly release code multiple times per day. With these rapid changes, complexity mounts, and teams develop a lack of understanding into how systems behave, leading to technical debt, drifting reliability metrics, and business risk.

Chaos engineering enables teams to prevent the unforeseen downtime and risk that can result from the complexity of cloud-native environments. With chaos engineering, developers test scenarios that could potentially bring an application down without impacting development speed.

âChaos engineering is the future of reliability and resiliency testing. It's the best way to ensure applications are pressure tested against all kinds of possible scenarios, from disaster recovery to handling traffic spikes,â said Uma Mukkara, Head of Chaos Engineering at Harness. âBringing these capabilities into the Harness Software Delivery Platform makes it even easier for customers to implement chaos engineering without slowing down development, increase developer productivity by improving system architecture understanding, and reduce time spent responding to production incidents. This will greatly help businesses reduce the risk of unplanned downtime and continue delivering exceptional customer experiences.â

Incorporating Resilience Best Practices with Harness Chaos Engineering

With Harness Chaos Engineering, organizations can adopt, scale, and automate software resilience best practices without negatively impacting velocity or requiring custom tooling and complex integrations. Harness CE identifies weak points within controlled systems-level experiments and equips developer and SRE teams with the information needed to prevent failures from happening in the future.Â

Harness CE, a cloud-native solution built on the open-source LitmusChaos project, enables enterprises to automate chaos experiments in the CI/CD pipeline with guardrails and resilience scoring, a quantitative measure of how resilient the target environment is when the respective chaos experiment is performed on it. As a result, deployments can be automatically rolled back if reliability issues are detected. With Harness CE's industry-leading capabilities, enterprises can build a reliable and resilient practice in days, not years.

To get started with CE, developers can create a trial account and follow simple tutorials to install, test reliability, and build resilience into the software delivery pipeline.

New Chaos Faults Expand Test Capabilities to Serverless Applications

The Chaos Engineering Enterprise ChaosHub has been expanded to 92 chaos faults that test the resilience of VMware, AWS, GCP, Azure, Kubernetes, and serverless systems.Â

As businesses adopt serverless architecture, the addition of six AWS Lambda chaos faults test the resilience of serverless processes when they fail. This helps developers understand if proper error handling is in place and/or if auto-recovery of failed transactions has been implemented.

These faults include:

- Delete Event Source Mapping
- Delete Function ConcurrenceÂ
- Toggle Event Mapping State
- Update Function Memory
- Update Function Timeout
- Update Role Permission

As IT outages reveal failure modes and new technology is adopted, Harness continues to expand its chaos fault library to ensure its customers can build resilience into their software delivery pipelines.

Building Reliable Software with Resilience Engineering

Harness CE helps enterprises achieve Continuous Resilience™, which ensures collaboration among all stakeholders of the software delivery lifecycle through the integration of Chaos Engineering with CD. As the only chaos engineering solution with native CD integration, Harness CE enables teams to maintain velocity while reducing downtime.

SREs, QA Engineers, and developers can quickly incorporate chaos experiments into their CD pipelines to continuously test and validate that their systems will maintain performance during a disruption.

Organizations can leverage Harness Chaos Engineering to:

- **Improve Developer Productivity** by empowering developers to innovate and create rather than getting distracted by incidents.Â
- **Accelerate Digital Transformation to stay ahead of competition** by driving innovation without disrupting business growth. With a deeper understanding of systems behavior, new technology and processes can be implemented more efficiently.Â
- **Improve Customer Experience** by ensuring systems are highly available and performant with faster response to outages.
- **Avoid Disasters** by understanding how systems handle failure.Â

Start Building Software Resilience Today with HarnessÂ

This native CD platform integration is the culmination of Harnessâs acquisition of ChaosNative Inc. and a major expansion of the Harness Software Delivery Platform. Customers can deploy each module independently with their existing tooling or use all of the Harness modules together to build a powerful, unified modern software delivery pipeline that spans Continuous Integration (CI), Continuous Delivery (CD), Feature Flags, Cloud Cost Management, Site Reliability Management (SRM), Security Testing Orchestration (STO), and Chaos Engineering.Â

If you are ready to see how your organization can adopt this practice and improve reliability, request a demo and sign up for the SaaS trial today!

Similar Blogs

ArgoCD, Terraform and Harness

CI/CD for Serverless

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.

Source URL: <https://www.harness.io/blog/harness-chaos-engineering-ce-key-capabilities>

Harness Chaos Engineering (CE) Key Capabilities

Chaos engineering helps organizations minimize unplanned downtime's financial and reputational impact. It also lets developers focus on software delivery rather than fire-fighting production incidents. Chaos experiments go beyond traditional unit, integration, and system tests and more closely represent random failures in a real-world production environment. This realistic environment provides insight into how systems behave, equipping teams to understand applications' and infrastructure weaknesses and proactively creating resilience to help prevent costly downtime. This blog will look closely at the product's key capabilities to see how it helps teams solve these challenges.

Harness CE provides:

- Chaos orchestration in CI/CD pipelines for Continuous Resilience™
- Unified experimentation across cloud providers and self-hosted platforms
- Steady state management for baselining and improving reliability
- Observability and ecosystem integration for visibility
- Robust experiment control methods for safe testing and automatic recovery rollbacks
- GameDay portal to proactively train your on-call team for incident response
- Enterprise dashboards, analytics, logs, and reports for clear communication
- Enterprise-grade audit trail and role-based access control (RBAC) for security
- Enterprise support to help businesses scale the practice quickly

Let's dive deeper into the capabilities that teams can leverage to increase reliability.

Chaos Orchestration in CI/CD Pipelines

Achieve Continuous Resilience™ with the native platform integration with Harness CE and Continuous Delivery (CD). Powered by the CNCF project, LitmusChaos, this integration makes it easier for Developers and SREs to test the reliability and resilience of applications in software delivery pipelines to improve overall reliability and minimize the risk of unplanned downtime.

Unified Experimentation Platform

Implement chaos engineering using our SaaS, self-hosted, on-premises, or air-gapped deployments to align with your business and security requirements. Harness supports injecting experiments into multiple platforms and environments. The Enterprise ChaosHub is a catalog of advanced experiments with coverage across VMware, AWS, GCP, Azure, Serverless and a full range of Kubernetes chaos experiments. Chaos experiments enable users to manage, edit, schedule, and run experiments within the UI for improved collaboration. Harness provides the largest and most diverse chaos experiments available today, with many more added monthly.

Chaos Orchestration and Reliability Management

Chaos orchestration enables users to build a CE practice quickly by letting the Harness solution fill the gaps in the organization's knowledge, processes, and tools. Utilize Harness CE to train new and existing employees to level everyone up on software reliability.

Roll out chaos engineering to the entire enterprise from a Git repository instead of waiting years to adopt the CE practice team by team. Start your entire enterprise on the chaos engineering practice to scale software reliability to every application. Leverage GitOps and CI/CD integrations to automate the complexity and meet developers where they are by providing declarative YAML files for chaos experiments that improve the developer experience.

The GitOps feature enables you to configure a single source of truth for your chaos experiments and execute them directly from Git, allowing a vast scope of automation in CI/CD pipelines.

A team can manage reliability through the resilience score to define, measure, and tune each experiment to track resiliency over time and automate experiment results.

Steady State Measurement

Rather than have developers manually look at monitoring dashboards and have eyes on glass with multiple browser tabs open, Harness CE provides probes that can automate the experiment's measurement. Probes are editable checks you can define for any chaos experiment to measure an experiment's success and failure conditions. Chaos Probe examples include simple querying of application health checks and system steady state metrics.

GameDay Portal

A GameDay is a series of experiments that serves a purpose, such as:

- New engineer training for on-call rotation
- Exploring unknown failure modes in a system
- Migrating a system to a new technology that enables education of that technology
- Incident re-creation to validate code fixes
- Validation of Disaster Recovery exercises

The Harness Chaos Engineering platform's GameDay feature constructs experiments to test with a team. Your GameDay is repeatable by defining it as a template. The feature enables a user to start, stop, and re-run experiments within one UI, allowing a team to test in small increments of failure. The team can also take notes and observations and create a checklist of tasks they need to complete, which can be added to a ticketing system.

Experiment Control Methods

Harness provides declarative chaos experiments to define configuration in a code repository, version, and edit through automation. This declarative approach empowers developers to build and automate reliability in their code.

Harness chaos engineering enables you to run faults in parallel (CPU fault + Memory fault) to mimic real-world events. In addition to this approach, you can run chaos experiments in parallel to model complex IT outages that often stem from multiple failure modes.

Run various experiments on different targets to simulate cascading failure across more extensive sets of services. This ability enables you to cause a network disruption on one cloud provider's availability zone and simultaneously run a resource exhaustion experiment, simulating traffic moving over to the redundant system.

Lastly, you can abort an inflight experiment that causes an impact beyond the desired test expectation. Users can manually or automatically set up abort conditions using probes defined with the tested system's health metrics and automate recovery scripts.

Observability and Ecosystem Integrations

Harness CE can send chaos metrics to popular observability and application performance monitoring (APM) solutions that enable developers to integrate with their ecosystem of reliability. This reduces developer toil because Harness CE can plug into their system. Our list includes Prometheus, Grafana, Dynatrace, Keptn, and more. Besides observability and monitoring integrations, you can integrate with load-testing tools or leverage your own test with a custom script.

Enterprise Dashboards, Analytics, and Reports

Different roles require additional views regarding dashboards and reports. Executives might want a high-level risk assessment on a single dashboard. An engineering manager might want to see the reliability status of all services. Regardless, Harness CE has all the experiment data, analytics, and reporting capabilities needed to be the centralized source for reliability.

Enterprise-Grade Audit Trails and RBAC

Harness has built a reputation in the CI/CD industry for having detailed audit trails and fine-grained RBAC. These audit trails make it quick and easy for engineering teams to pass audits, often turning what would be days of effort into just a few hours. Our fine-grained RBAC model means that you can implement a permissions system that meets your organization's needs - no matter how complex.

Enterprise Support

Harness recognizes that enterprises need to move fast and scale quickly to meet the demands of their business, so we're equipped to offer enterprise support to ensure your chaos engineering practice can begin as quickly and safely as possible. Harness CE was built by the same team of experts that created the CNCF open-source project, LitmusChaos. This team is ready to support SaaS, on-premises, self-hosted, or air-gapped installations and provide onboarding assistance, feature enhancements, chaos best practices, and custom tooling integration for CI/CD and observability platforms.

Start Improving Software Reliability Today with Harness Chaos Engineering

Getting started with chaos engineering has never been so simple. If you are ready to see how your organization can adopt this practice and improve reliability, request a demo and sign up for the SaaS trial today!

Similar Blogs

[ArgoCD, Terraform and Harness](#)

[CI/CD for Serverless](#)

Elevate Your Code Quality with Harness SEI: Mastering the Art of Software Excellence

January 2024 Product Updates

Sign up now

Sign up for our free plan, start building and deploying with Harness, take your software delivery to the next level.

Get a demo

Sign up for a free 14 day trial and take your software development to the next level

Documentation

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

Case studies

Learn intelligent software delivery at your own pace. Step-by-step tutorials, videos, and reference docs to help you deliver customer happiness.

We want to hear from you

Enjoyed reading this blog post or have questions or feedback?

Share your thoughts by creating a new topic in the Harness community forum.

Sign up for our monthly newsletter

Subscribe to our newsletter to receive the latest Harness content in your inbox every month.
