

Continuous Delivery Essentials – Deploying on K8s



Course Outline



- How to differentiate CI from CD
- Understand the steps involved in typical deployments
- Learn industry best practices for deployments
- Explore Harness's capabilities for automated CD
- Setup a basic Harness delegate install
- Configure a basic Kubernetes deployment using Harness

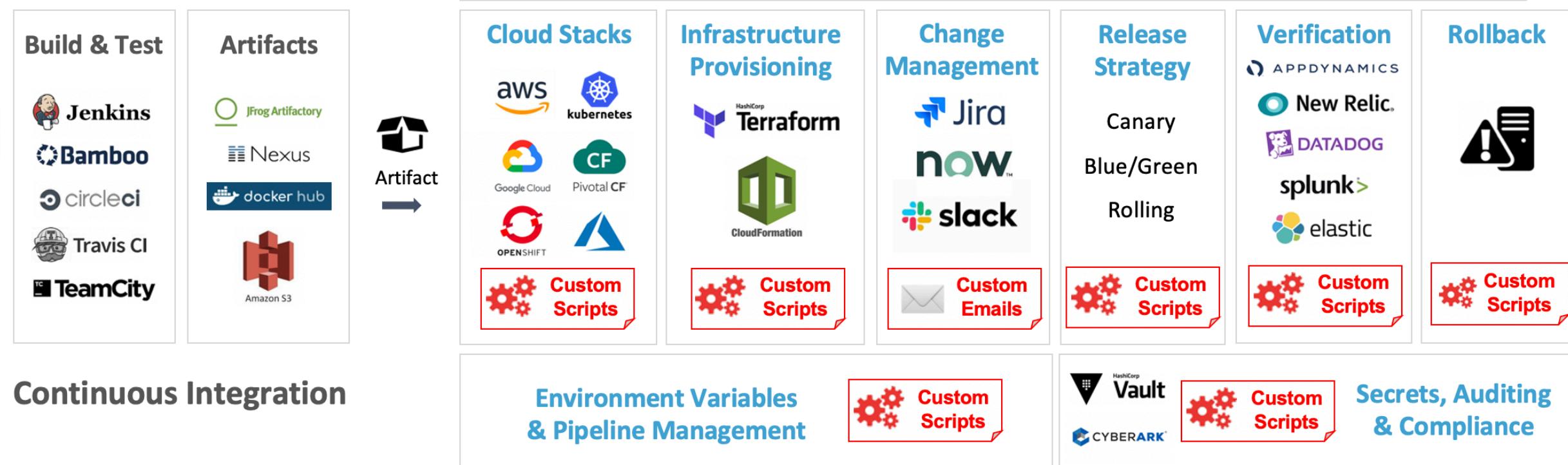


Part 1:

CI versus CD



Continuous Integration != Continuous Delivery



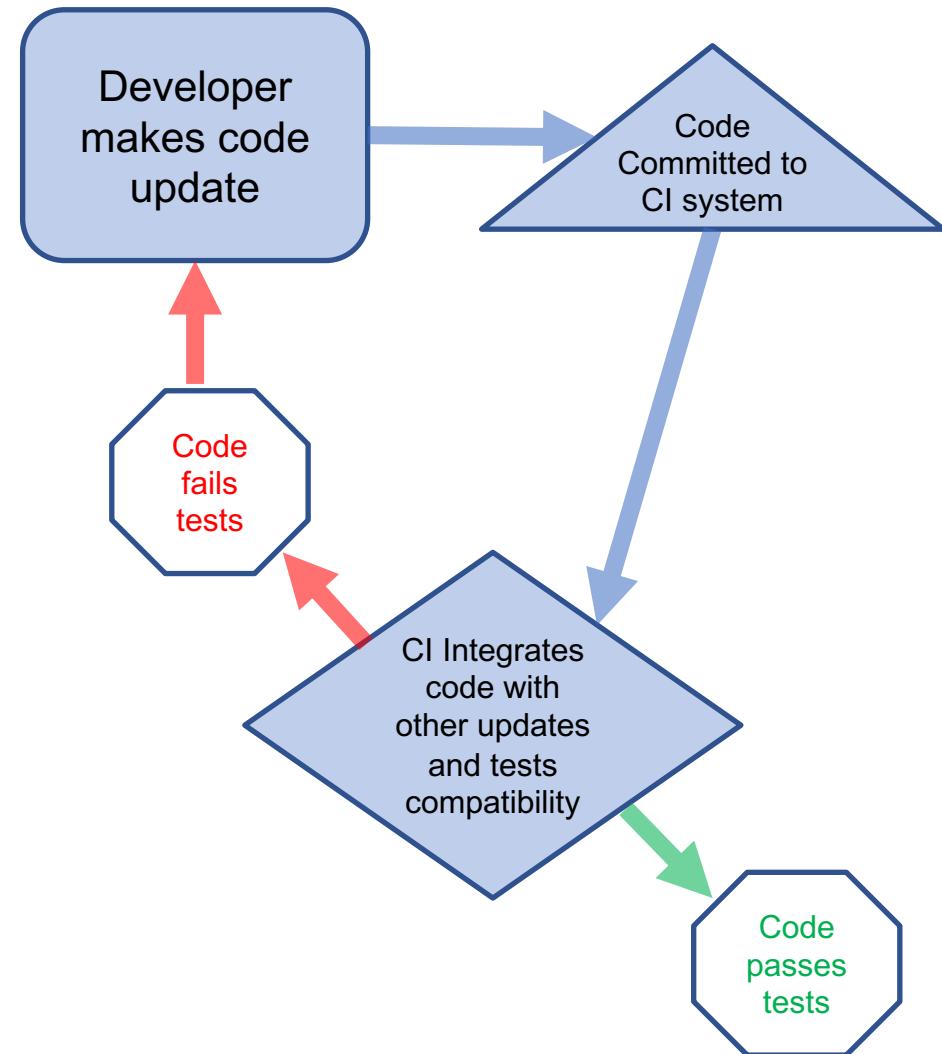
Continuous Delivery

The CI Cycle and Purpose



- CI integrates and tests small changes to the codebase from multiple developers – usually many times daily
- One goal is to keep the various parts of the project as up to date as possible. That way different developers are using the most up-to-date copies of their peer's code to make their own updates
- Another goal is to keep the main branch of the project integrated, bug free, and up to date – as “deployable” as possible

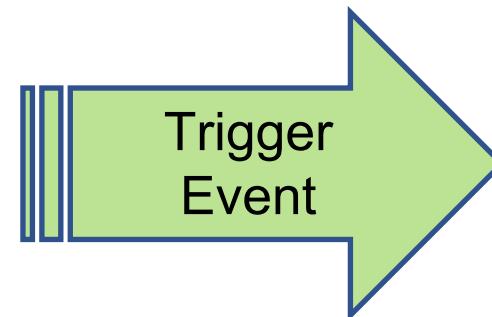
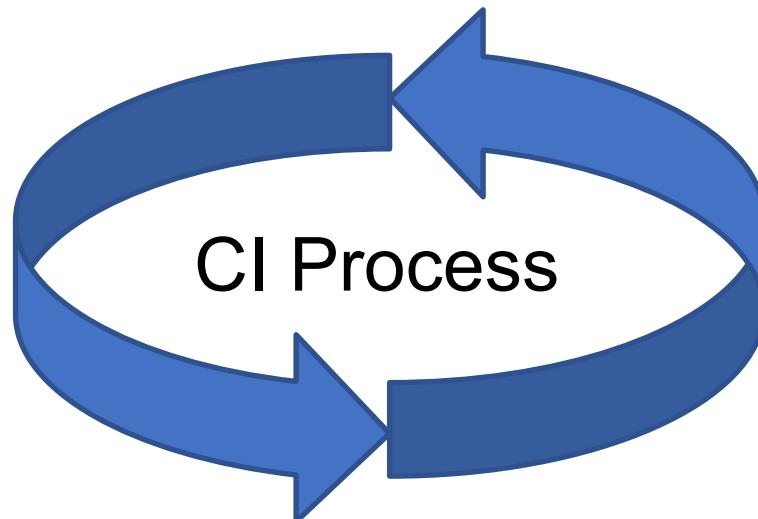
CI Process



When Does the CI Cycle End?



- Continuous Integration ends when you have a “releasable artifact”
- A releasable artifact could be many things: Docker image, VM image, AMI, .jar file, other packaged software bundle
- Not every successful pass through the CI cycle yields a releasable artifact though
- A “trigger” event ends the CI cycle and generates an artifact



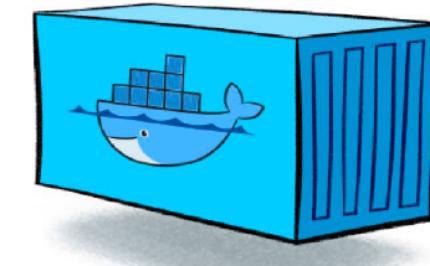
Artifact



What's an Artifact?



- Artifacts are units of deployable code – they range from large to small.
- Some examples: VM Images, AMIs, .jar files, Docker containers, serverless code, etc.



What Triggers That Trigger?



- Sometimes it's based on business needs, sometimes it's arbitrary:

We've got to get
Penultimate Fantasy 17 out
before Cyber Monday or
we're toast!

Janette McGamexec
CTO, Second To Last Games, Inc.

I say we only release
on the full moon! The
pack has spoken.

Michael J. Teenwolf
Head Developer, Werewolves of London, Ltd.

- Sometimes it's just because deploying is hard, and you can't ask your team to give up all their Saturday nights. So you settle on once a month/quarter/year.
- In a perfect world you'd release as often as you could . . .

What is Continuous Delivery?

“Continuous Delivery is the ability to get changes of all types—including new features, configuration changes, bug fixes and experiments—into production, or into the hands of users, *safely* and *quickly* in a *sustainable* way.”

Jez Humble, Founder continuousdelivery.com

How Often *Is* Continuous Deployment?



- Lots of factors determine this of course, but if your CI process is dialed in and your main branch is kept as bug free and "deployable" as possible you can really up the frequency of your deployments
- The target is to achieve an interval that meets the business AND engineering needs
- The process is to increase your deployment frequency over time. Allow "virtuous cycles" in your CI and CD systems to constantly shrink your deployment interval



Harness.io SaaS Is a Product of Continuous Deployment



- Harness deploys new features and feature updates every day at 5 PM
- Bug fixes and security updates are deployed immediately once they are fully vetted – sometimes multiple times per day



Part 2:

Typical Steps to Deployment



CD Pipelines



- Pipelines are the set of steps required to move an artifact from its repository to its final destination.
- Inside a single pipeline there can be multiple deployments, e.g. deploy to QA, deploy to Staging, deploy to prod.



Building Blocks of a CD Pipeline



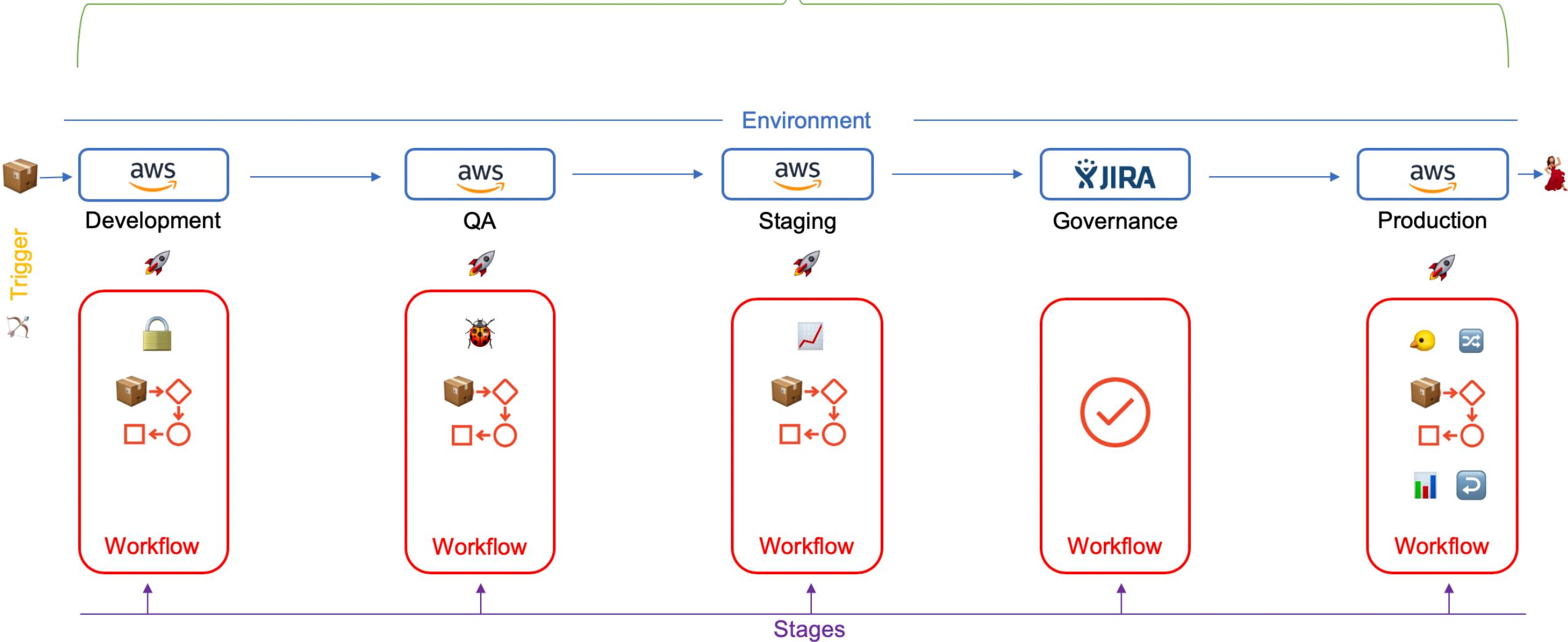
Typical deployments follow these steps:

1. Infrastructure and Platform deployment
2. Change management
3. Push Deployment
4. Verification
5. Roll Back (if necessary)





Pipeline



It All Starts With an Artifact

- Deployable artifacts live in repositories
- Besides storing and controlling access to artifacts, repositories sometimes perform other functions such as vulnerability scanning and secrets management



Artifact

Deploy to Staging with
staging certs and
secrets

Staging



Artifact

Deploy to Prod with
Prod certs and secrets

Prod



Amazon ECR



Azure Registry



Bitbucket



GitLab



GitHub



Where Do Artifacts Go?



- Artifacts get deployed to places like:
 - Staging/QA
 - Production
 - Multiple Prod locations for DR and HA



Artifact

Deploy to Staging with
staging certs and
secrets

Staging



Artifact

Deploy to Prod with
Prod certs and secrets

Prod

Moving Artifacts With Environment and Cert Variables



- As artifacts get deployed to different environments consideration must be made to ensure the right environment and cert variables get set

Set Staging Variables

```
$db = testdb.mycompany.com  
$logger = elk.mycompany.com  
$product = testcat.mycompany.com  
$cert = vault.staging.mycompany.com/ . . .
```

Set Prod Variables

```
$db = proddb.mycompany.com  
$logger = splunk.mycompany.com  
$product = prodcat.mycompany.com  
$cert = vault.prod.mycompany.com/ . . .
```



Artifact

Deploy to Staging with
staging certs and
secrets

Staging



Artifact

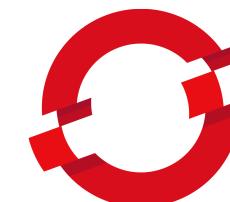
Deploy to Prod with
Prod certs and secrets

Prod

Infrastructure and Platform Deployment



- Before you can deploy your new artifact
 - it needs somewhere to deploy to:
servers, network, load balancers,
firewalls, etc.
- Before you can run those containers /
serverless workloads they need a
platform setup



Pivotal CF®



Release Strategy



- Who approves your deployment?
Sec-Ops? Product Management?
Execs? All of the above?
- What tools do they use for this approval?



Pushing the Deployment



- There are multiple ways to deploy your artifact:
 - **Rolling Upgrade** – for many PaaS's this is the default deployment. Slowly scale up the new version while simultaneously slowly scaling down the old version.
 - **Blue/Green Deployment** – stand up a complete deployment of the new version (blue) while leaving a complete copy of the old version (green) up and running.
 - **Canary Deployment** – A variation of Blue/Green. Stand up a small deployment of the new version (the canary) and direct a small portion of incoming traffic to the canary





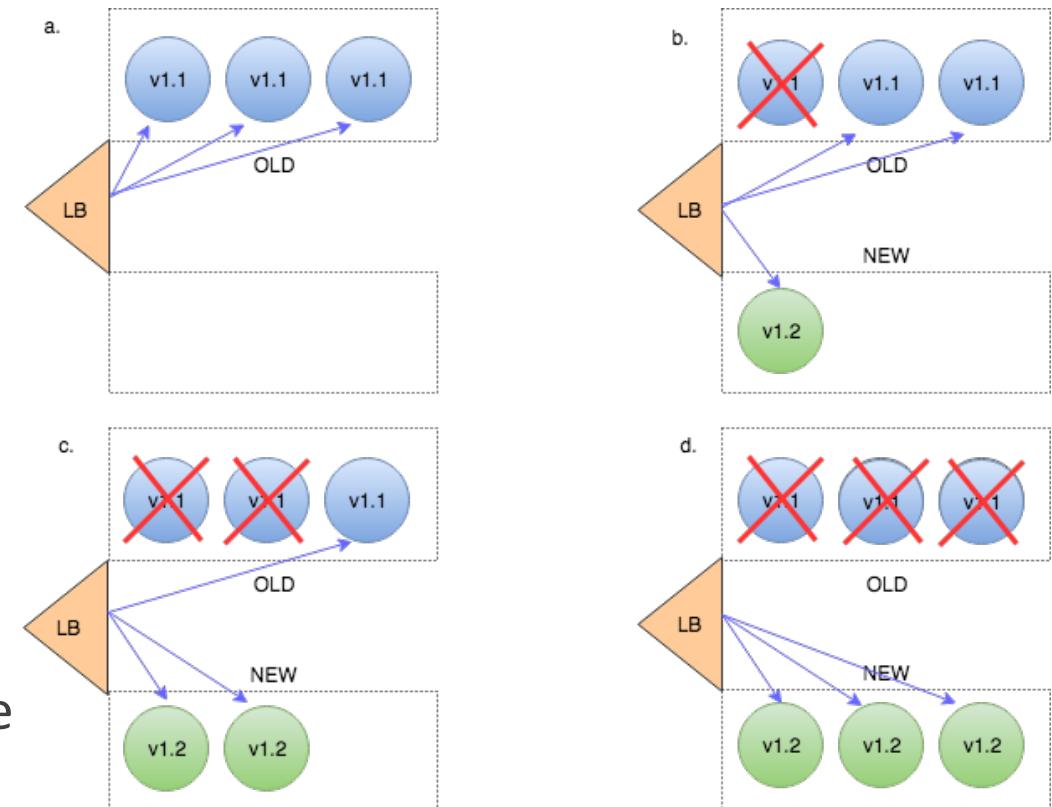
Types of Deployments – Rolling Upgrade

- **Pros:**

- Easy to do – the default upgrade for many PaaS's
- Respects hardware limitations / footprints
- Minimally disruptive *if* everything goes well

- **Cons**

- Slow – can take a long time depending on what you're deploying
- Rollback is quite difficult / time consuming / maybe impossible – roll forward more likely





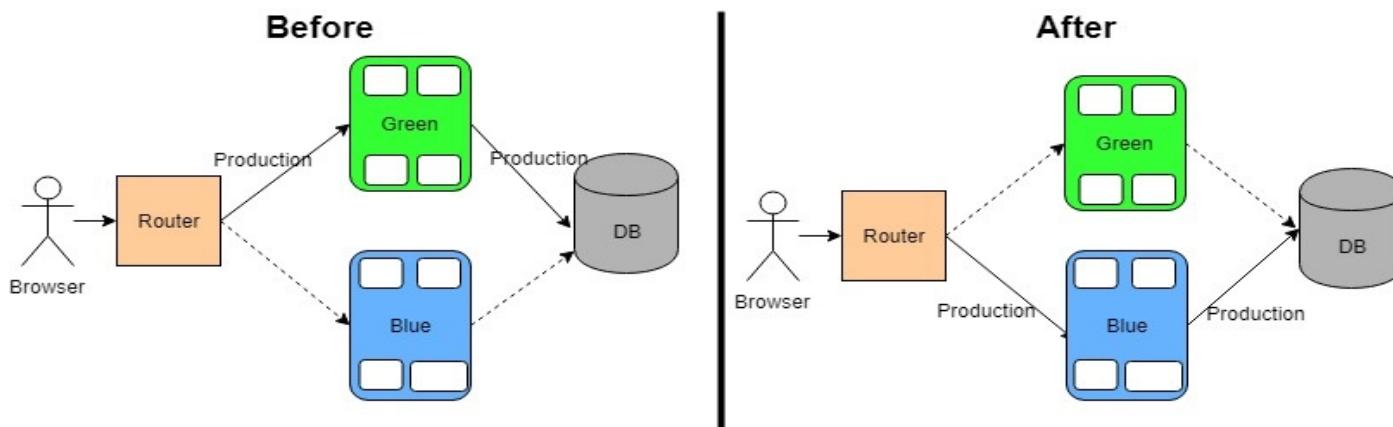
Types of Deployments – Blue / Green

- Pros:

- Very easy rollback – just flip back to the green version
- Easy networking – flip a switch on the LB

- Cons

- Slow – can take a long time to get the entire Blue version up and running
- Hardware / cloud footprint intensive – requires twice the infrastructure while both versions are up and running



Types of Deployment - Canary

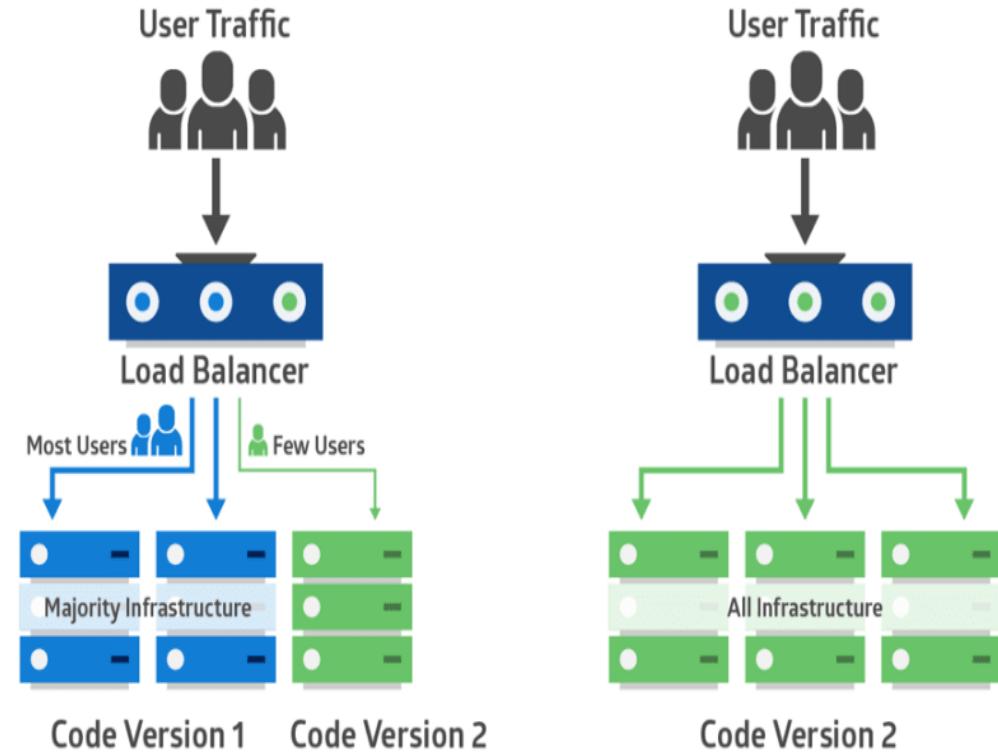


- **Pros:**

- Minimal extra hardware/cloud footprint
- Rapid deployment and feedback – canaries come up quickly and give feedback on quality of deployment quickly
- Easy-ish rollback – just kill the canary and reshape the network traffic back to the original

- **Cons**

- Very difficult to setup – requires complex networking and orchestration deployment



Deployment Verification



- When is a deployment *actually* finished? It didn't crash? It's running as well as the previous version? It's running better than the previous version? How do you know?
- APM software and log aggregation software can tell us how it went / is going



Deployment Rollbacks



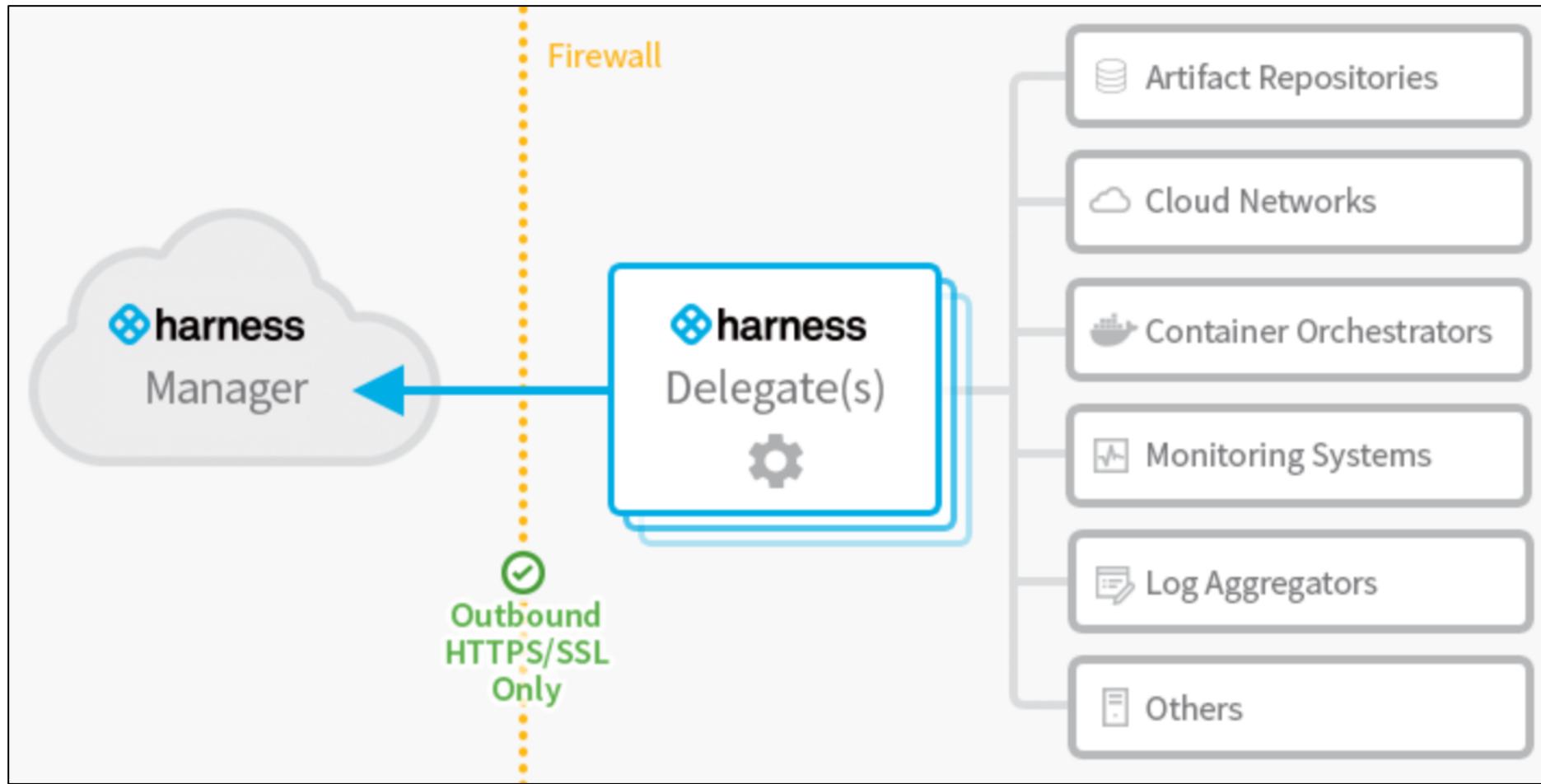
- Depending on the deployment type this might be impossible
 - Many instead do a roll-forward: Deploy the last previous known good deployment as if it were a new deployment
- Blue/Green or Canary types – just flip the networking switch back to the Green/non-canary deployment



Harness Delegates



The Harness.io Platform





Main Dashboard

See any problems detected with the Harness Delegate or other platform components.

ATTENTION NEEDED:

! 2 Setup Problems Detected

No Pending Approvals

No Pending Manual Intervention

Deployments

221

Deployments

See the total number of deployments, the most failed deployments, and the Service instances deployed each day.

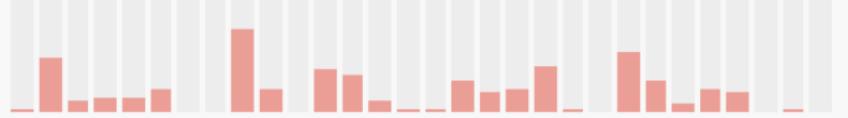


See success and failure rates for the most active Services. Click a Service name to get a detailed view of its artifacts, Environments, and deployment infrastructures.

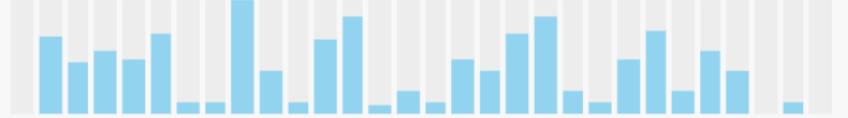
Most Active Services

Order (Retail Application)	148
Catalog (sahithi-app)	21
DockerImage (Ankita App 2)	19
MyService (CretzmanApp)	9
career-contractorbenchmarking-main-be (Mercer HR)	5
Catalog (Retail Application)	5
Todolist-Datadog (Retail Application)	3

156
Failed Deployments



743
Instances Deployed



Service Instances

See instances by Service, by Cloud Provider, and by Production versus Non-Production Environments.

29

Total



By Service



By Cloud Provider



By Production/Non-Production

20

31

Harness Manager – SaaS or On-Prem



- The “brains” of a Harness implementation
- Usually runs as SaaS managed by Harness, but can be run on-prem for high security and air-gap implementations
- All configurations and deployments are stored here
- Deployments get “kicked off” from here



Harness Delegates



- The “worker” or “proxy” of a Harness implementation
- Installed and runs inside your infrastructure, on-prem or cloud
- Multiple delegates can be run for high availability
- Multiple delegates may also be needed due to internal network policies
- Connects to the various internal systems to facilitate deployments



Delegate Connectivity



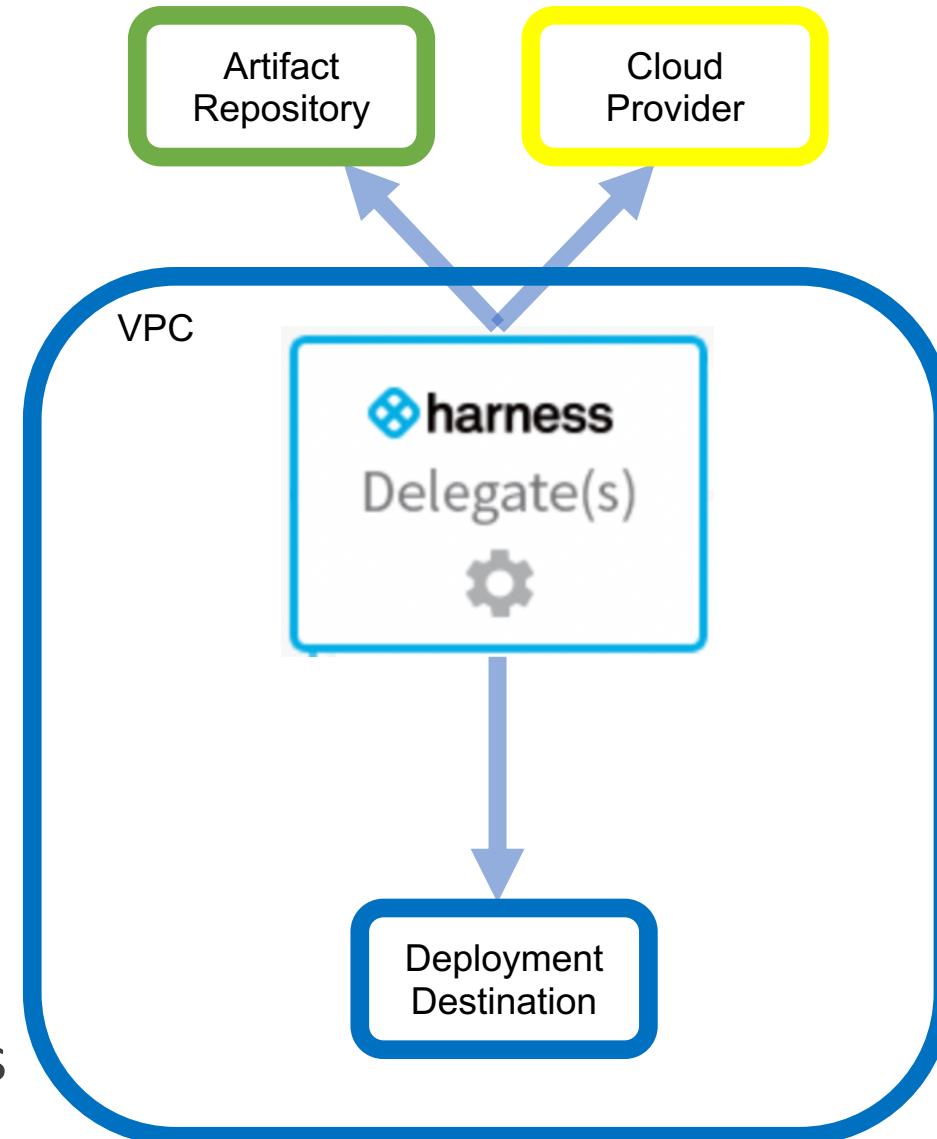
- Only outgoing connection from Delegate to Manager is HTTPS on port 443
- Delegates must also connect to other systems to deploy. For example:
API/CLI calls to cloud providers, authenticating and downloading from artifact repositories, obtaining data from verification providers such as AppDynamics or Splunk, and many more depending on the deployment and environment
- Connectivity considerations play a large role in Delegate deployment – See *Enterprise Harness Deployments* for details



Where to Install Delegates



- Where and how to run Delegates can be a complex subject that's covered in detail in the Enterprise Harness Implementations course
- For now just know that Delegates need connectivity to the artifact repository, the cloud provider, and the destination systems the artifact will deploy to
- Generally you'll want to run at least one Delegate per VPC / security zone in your systems



Delegate Run Time Specs



- Linux/UNIX server or container
- Minimum 1 CPU
- Minimum 8GB RAM. Ensure that you provide the minimum memory for the Delegate and enough memory for the host/node system. Minimum 6GB Disk space
- ulimit set to at least 10000
- Access to artifact servers, deployment environments, and cloud providers.



Delegate Run Time Specs - Cont



- Delegates do NOT need to run as root
- Delegates that run in a container do run as root by default
 - that includes Docker, K8s, and ECS.
- It's possible to run containerized Delegates as non-root, but you lose the ability to use Delegate Profiles
 - We will cover Delegate profiles later in the course. You can read about them here: https://docs.harness.io/article/h9tkwmkrm7-delegate-installation#delegate_profiles



Giving Delegates a Happy Home



- Delegates can be run as either a normal Linux process or in an Ubuntu container with an orchestrator
- The server or container running the Delegate may also need other packages installed such as kubectl, Terraform, or cloud provider CLIs
- This can be accomplished by using Delegate Profiles in Harness or configuring the underlying VM image or container the Delegate will run on



Delegate Profiles



- Delegate Profiles are scripts that run when a new Delegate in that Profile gets installed, or you add a Delegate to that profile
 - Updates to an existing Delegate Profile will cause all members to rerun their scripts
- For the container-based Delegates with root access, you can use normal Ubuntu `apt-get` commands to install packages
- For any other system follow the package install instructions for that version of Linux
- See <https://docs.harness.io/article/nxhlbgkj-common-delegate-profile-scripts>



Container Based Delegates

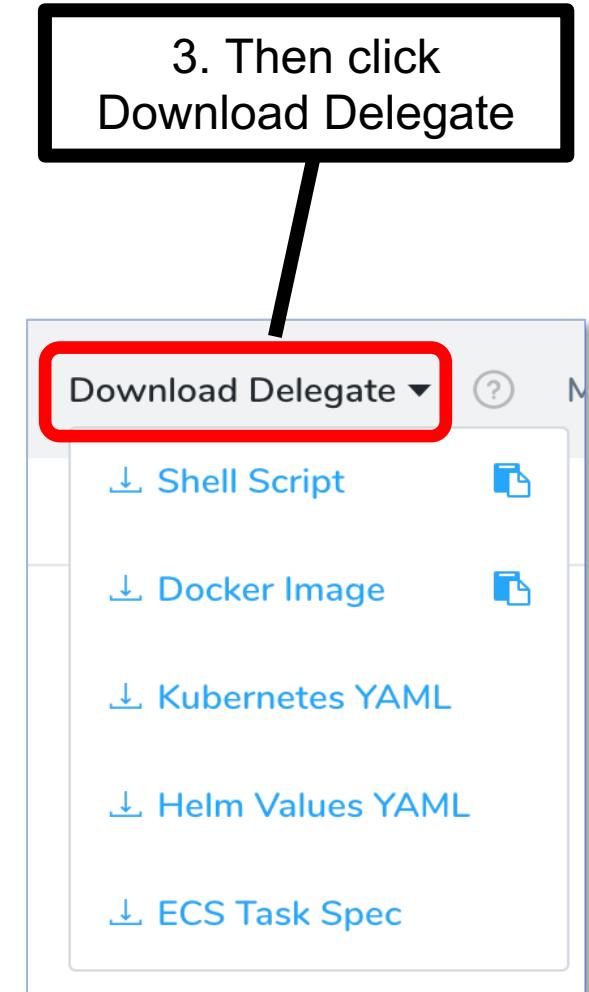
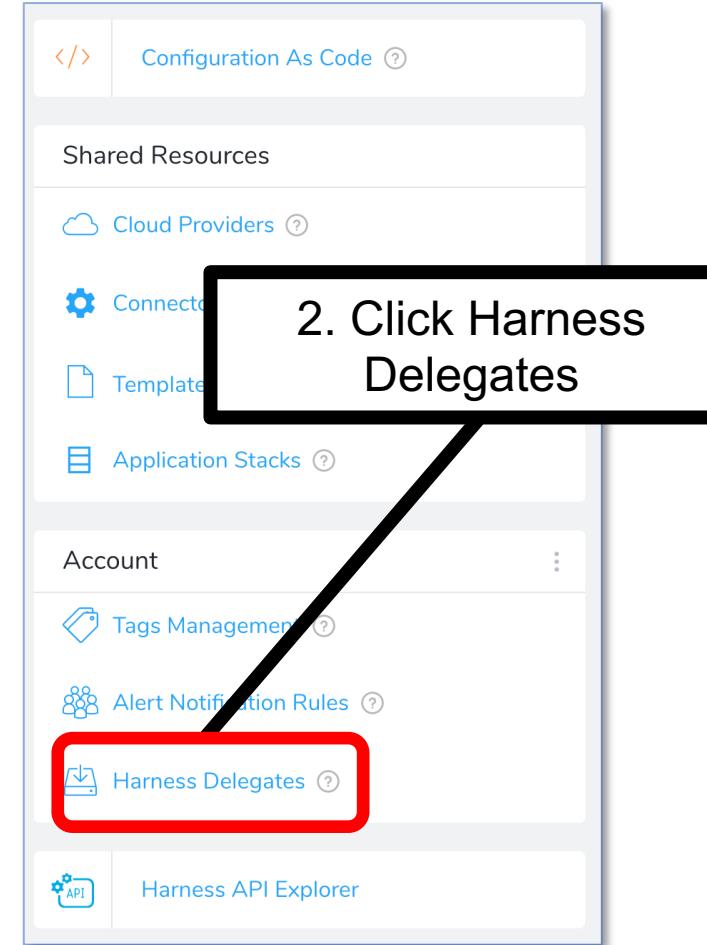
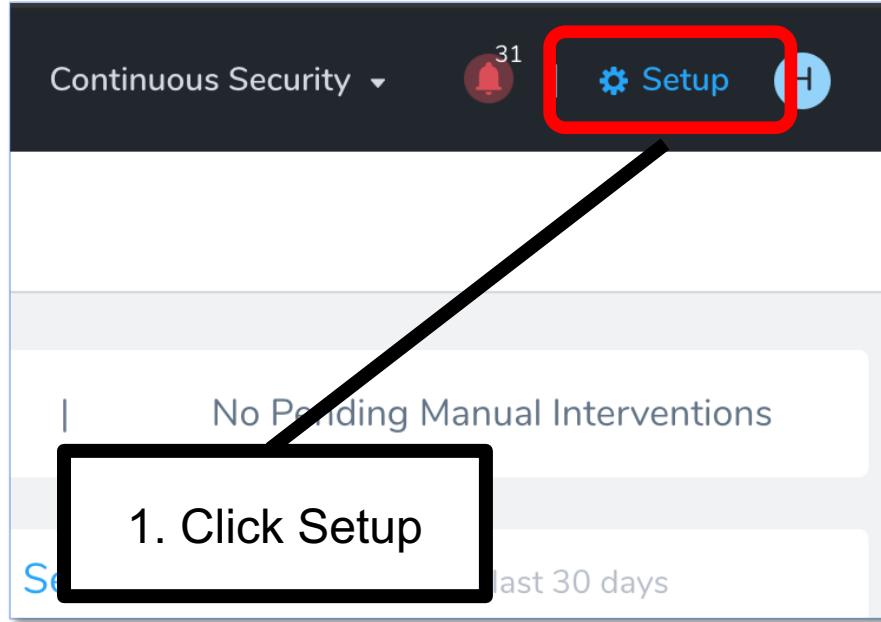


- Docker, K8s, and ECS Delegates are all based on the containerized version of the Delegate
- The Delegate runs in an Ubuntu 18.04 image
- The image can be downloaded and added to a private repository
- It can also be customized using this docker file as an example:

<https://github.com/bfeuling01/harness-io/blob/master/delegate/Dockerfile>



Installing a Delegate – Getting There

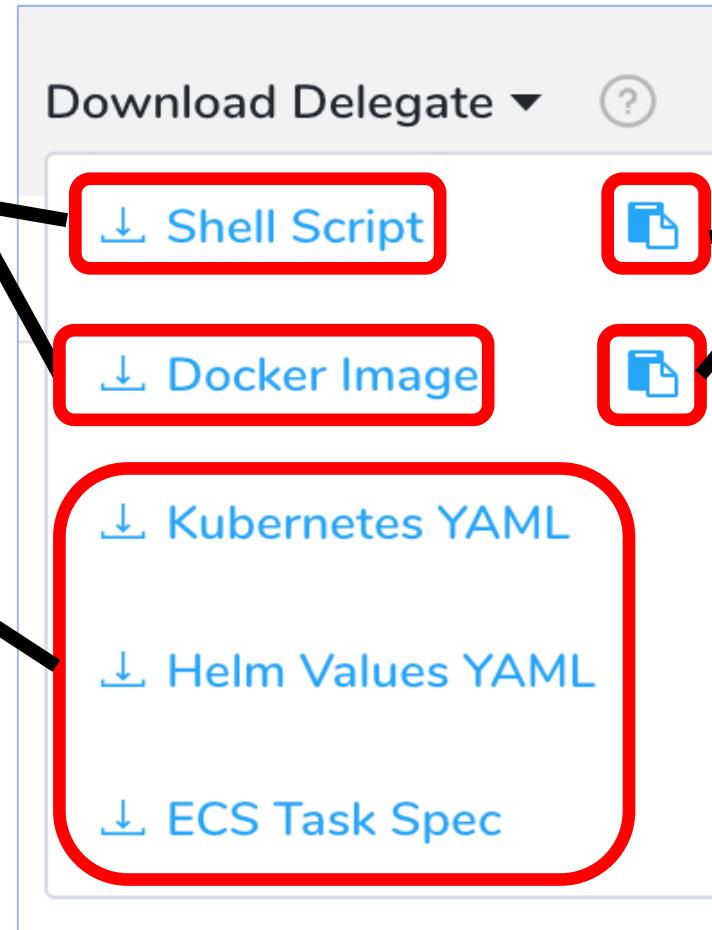


Download Delegate File or curl Command



Click to download Delegate locally

Click to download local copy of installation file for each platform



Click to copy curl command to clipboard for remote installations

```
sh-3.2$ curl -o harness-delegate.tar.gz  
"https://app.harness.io/gratis/api/delegate  
s/download?accountId=d1XXXXXXX2Gi_UFimZzDaA  
&token=eyJhbGciOiJIUzI1NiJ9.eyJyZXNvdXJjZSI  
6ImRlb..."
```



K8s Delegate Install



- To generate the correct yaml for Kubernetes Harness needs to know about the name of the delegate any profile settings

Delegate Setup ⓘ

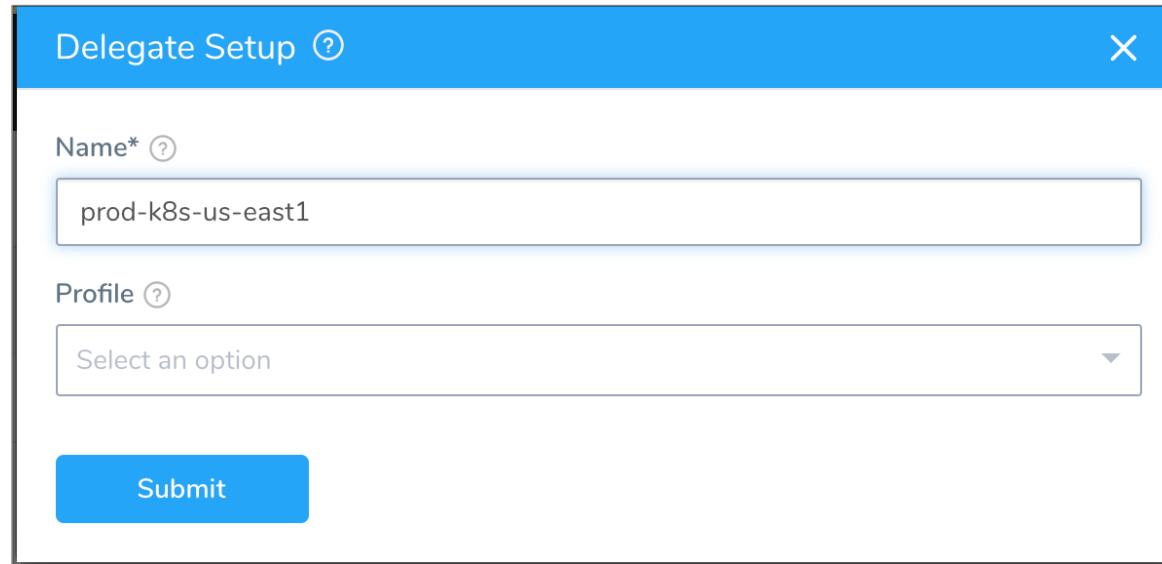
X

Name* ⓘ

Profile ⓘ

Select an option

Submit



- Download and unzip the tar.gz file like the other delegates
- The zip contains a yaml file to run your Delegate in Kubernetes

The K8s Yaml File Unpacked



- The Delegate yaml file creates the following Kubernetes structures:
 - A namespace named “harness-delegate”
 - A cluster-admin ClusterRoleBinding – this is the default, but can be changed to something more limited within the cluster if needed
 - The actual delegate process itself is a single replica set with a 1 CPU and 8 GB memory limit
 - Proxy settings can also be added to the yaml for connectivity to Harness.io Manager service



K8s Delegates



drew-k8s-testing-nkuryn-0

Hostname

drew-k8s-testing-nkuryn-0

Description

delegate used for k8s stuffs



Profile

[Install Helm](#) ▾ Last executed: 01/09/2020 03:25 pm [View Logs](#)

IP

192.168.156.4

Status

Connected

Last heartbeat

22 seconds ago - 01/09/2020 03:25 pm

Active Versions

1.0.46902

Scope Limited To

prod

[+ Add Scope](#)

Scope Excluded

[+ Add Scope](#)

Tags

k8s

prod

stuff

and

or

things

Edit



Running a Delegate With Local / Laptop K8s



- By default Harness delegates are sized for high volume / production workloads
- It's possible to run a delegate locally on your laptop if you resize the delegate footprint AND have enough resources on your laptop to give Minikube or Docker Desktop at least 8 gigs of RAM and 2 CPUs
- Change the harness-delegate.yaml memory spec from “8Gi” to “4Gi”
- See the self-guided labs that accompany this class for more details

```
minikube start --memory 8192 --cpus 2
```



Delegate Helm Install



- Similar to the K8s install
- Requires Tiller to be installed
- See the docs for further details:



<https://docs.harness.io/article/6n7fon8rit-using-the-helm-delegate>





Lab 1 Demo

Building Deployments with Harness



Hooking Harness Up to Your Environment



- For Harness to work its magic on your deployments it must make connections to various systems that will run or manage your deployments
- In Harness these take two forms: Cloud Providers and Connectors



Setting Up Cloud Providers



4

1

2

3

Setup

Cloud Providers

+ Add Cloud Provider

Use the Test button
to test the auth
credentials for the
cloud provider

Kubernetes Cluster ? X

Type: Kubernetes Cluster

Display Name*: myEKScluster

Cluster Details:

Inherit Cluster Details from selected Delegate ?
 Enter Cluster Details manually

Master URL*: https://42A7AA2CA66XXXXXX2DCAE9ACC11F.gr7.us-east-1.eks.amazonaws.com

User Name: admin Password: *****

CA Certificate: Client Certificate:

Client Key:

Client Key Passphrase: Client Key Algorithm:

Test Submit

Types of Cloud Providers



Connector

X

Type

✓

- Amazon Web Services
- Google Cloud Platform
- Physical Data Center
- Kubernetes Cluster
- Microsoft Azure
- Pivotal Cloud Foundry

- See the docs for specific settings for each provider:

<https://docs.harness.io/article/whwnovprrb-cloud-providers>

- Except Kubernetes, covered next

Kubernetes Cloud Provider - Authentication



- 4 ways to authenticate to K8s – choose only one
 - Use the authentication settings from a Harness Kubernetes Delegate already running in the cluster
 - Use simple username and password
 - Use certificates
 - Use K8s service account
- Credentials get validated when you save the K8s Cloud Provider by doing a list of services in the default Namespace – select skip verification if you have deleted the default Namespace or the credentials used don't have access to the default NS



Setting Up Connectors



Setup > Connectors

Artifact Servers ?

(1) Harness Docker Hub

Verification Providers ?

(0) None

Source Repo Providers ?

(0) None

Collaboration Providers ?

(0) None



Amazon ECR



DATADOG



Connector Specifics



Connector

Type

- SMTP
- Jira
- ServiceNow

Add Splunk Verification Provider ⓘ

Note: Non-SAML account with permission to search indices required for the APIs.
Customer needs to open ticket with the Splunk support, if not done already.

Display Name* ⓘ

Splunk

Connector

Type

- Jenkins
- Bamboo
- Docker Registry
- Nexus
- Artifactory
- SMB
- SFTP
- Helm Repository
- Azure Artifacts

- See both inline help in the Harness UI as well as the Harness Docs for specifics on setting up each different system
 - https://docs.harness.io/article/ifx4yks50s-connectors-and-providers-setup#docker_registry_across_multiple_projects

Applications to Organize Your Deployments



- Applications are arbitrary groupings of deployments inside Harness
- Use them to:
 - Create RBAC divisions for compliance, security, etc.
 - Establish other enterprise-specific organizational divisions

A screenshot of the Harness web interface showing the 'Applications' page. The page has a header with the title 'Applications' and a search bar. Below the header, it displays '4 Applications'. There are four entries, each consisting of a small icon followed by the application name and its description in parentheses.

Application	Description
</> mycompanydotcn	(中文版)
</> mycompanydotcom	(english version)
</> mycompanydotde	(deutsche version)
</> mycompanydotfr	(version française)

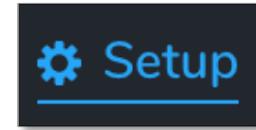
Creating an Application



1

2

3



+ Add Application

Application

Name* ⓘ

mycompanydotcom 2

Description

english version

Set up Git Sync

Submit



Inside an Application



mycompanydotcom [?](#) english version

[Application Defaults](#)



Services [?](#)

(0)



Environments [?](#)

(0)



Workflows [?](#)

(0)



Pipelines [?](#)

(0)



Triggers [?](#)

(0)



Infrastructure Provisioners [?](#)

(0)



What Are Services

- Services are the individual components / microservices that make up your stack
- In Kubernetes usually represented by a group of yaml's to define the particular service parameters
- In Helm represented by a chart
- See docs for more details

https://docs.harness.io/article/i3n6qr8p5i-deployments-overview#deployment_guides

Add Service

Name* ⓘ

Description

Deployment Type* ⓘ

- Kubernetes
- Amazon ECR Container Services (ECR)
- Helm
- Kubernetes
- Pivotal Cloud Foundry
- Secure Shell (SSH)
- Windows Remote Management (WinRM)



Setting Up Harness Kubernetes Service* - Artifact Source



Setup > mycompanydotde > Services > webFrontEnd

Service Overview

Name	webFrontEnd
Deployment Type	Kubernetes
Artifact Type	Docker Image
Description	The front end part of the stack
Artifact Source	+ Add Artifact Source
Tags	

Manifests

- Search
- Files
 - templates

Artifact Source - Docker Registry

Display Name Auto Generate Name
(Name will be auto generated)

Source Server* ⓘ
Harness Docker Hub

Docker Image Name* ⓘ
latest/nginx

Submit

* Please note that a “Harness Kubernetes Service” is not the same thing as a “Kubernetes service”

Setting Up Harness Kubernetes Service – Manifests Editor



Manifests ?

The screenshot shows the Harness Manifests Editor interface. On the left, there's a sidebar with a search bar and a three-dot menu icon. Below that is a tree view of files under 'templates': 'deployment.yaml' (selected), 'namespace.yaml', 'service.yaml', and 'values.yaml'. A context menu is open over 'deployment.yaml' with options: '+ Add File', 'Rename File', and 'Delete File'. The main right-hand panel shows the YAML code for 'deployment.yaml' with line numbers 1 through 13. The code uses go text template syntax to define a ConfigMap and a Secret based on environment variables.

```
</> deployment.yaml
```

```
1 {{- if .Values.env.config}}
```

```
2 apiVersion: v1
```

```
3 kind: ConfigMap
```

```
4 metadata:
```

```
5 | name: {{.Values.name}}
```

```
6 data:
```

```
7 {{.Values.env.config | toYaml | indent 2}}
```

```
8 ---
```

```
9 {{- end}}
```

```
10
```

```
11 {{- if .Values.env.secrets}}
```

```
12 apiVersion: v1
```

```
13 kind: Secret
```

- Comes preconfigured to use the go text template package
- <https://godoc.org/text/template>



Service Specific Configurations



Configuration ?



Config Variables ?

+ Add Variable

Config Files ?

+ Add File

Values YAML Override

+ Add Values



- It's possible to specify service level variables – both plain text and encrypted
- It's also possible to include service-specific config files and values yaml overrides



Environments



- If Harness Services are what you are deploying, Environments are where you are deploying them to
- For example: Prod, QA, Staging, multi-geo sites for HA and DR, etc.

Environment ⓘ

Name* ⓘ

Description

Environment Type* ⓘ

Submit

Setting Up Environments for Kubernetes



Setup > mycompanydotde > Environments > dotdeQA

Environment Overview

Name: dotdeQA
Description: QA for .de domain
Environment Type: Non-Production
Tags: + Add Tag

Infrastructure Definitions

No Infrastructure Definition.

+ Add Infrastructure Definition

Service Configuration Overrides

No Service Configuration Overrides.

+ Add Configuration Overrides

Screenshot

?

Environments – Infrastructure Definition



- Infrastructure Definitions are based on which Cloud Provider Type you use
- Kubernetes Cluster Cloud Provider Type covers Helm and all flavors of Kubernetes

Cloud Provider Type*
Kubernetes Cluster
Amazon Web Services
Microsoft Azure
Google Cloud Platform
Kubernetes Cluster
Physical Data Center
Pivotal Cloud Foundry



Environment Infrastructure Definition K8s



Infrastructure Definition X

Name* ⓘ

Cloud Provider Type* ⓘ

Kubernetes Cluster x ▾

Deployment Type* ⓘ

Kubernetes x ▾

Cloud Provider* ⓘ

Kubernetes Cluster: Harness Sample K8s Cloud Pr... x ▾

Namespace ⓘ

Release Name* ⓘ

Scope to specific Services ⓘ

Submit



Make Environment-Specific Exceptions



Service Configuration Override ⓘ

Service* ⓘ

webFrontEnd(Docker Image) x ▾

Override Type Variable Override File Override Values YAML

Configuration Variable ⓘ

\$logger

Override Scope ⓘ

Entire Environment

Type

Text

Override Value* ⓘ

elk

Submit



Setting Up Workflows – Deployment Types



Workflow Type	Description
Basic	A Basic deployment selects nodes and installs a service.
Multi-Service	A Multi-Service uses one or more phases composed of separate steps.
Canary	A Canary deployment rolls out a new app version to small sets of users in separate phases, tests and verifies it at each phase, gradually rolling it out to your entire infrastructure.
Build	A Build deployment simply collects artifacts.
Rolling	A Rolling deployment lets you gradually roll out your deployment, enabling and disabling services as necessary.
Blue/Green	In a Blue/Green deployment, network traffic to your service/artifact is routed between two identical environments called blue (staging) and green (production). Both environments run simultaneously, containing different versions of the service/artifact.

Workflow Components



- Workflows are made up of different steps
- Depending on the type of deployment and cloud provider you're using there will be different steps and options available in your workflows
- See the docs for details on non-Kubernetes Workflows

Pre-deployment Steps :

+ Add Step

Deployment Phases

+ Add Phase

Post-deployment Steps :

+ Add Step



Building a Workflow



- Workflow steps and options will vary depending on your type of deployment as well where you're deploying to
- Add actions to your workflow using the Harness UI
- For Kubernetes based deployments you can run K8s specific commands and actions.

▼ Kubernetes (8)

A screenshot of the Harness UI showing a list of Kubernetes actions. The actions are displayed in a grid format with a light gray background. Each action card has a title, a star icon, and an information icon. A 'Delete' button is also present. Below the cards is a large 'Apply' button.

Canary Depl...	Blue/Green ...	Rollout Depl...	Swap Servic...	Traffic Split	Scale	Delete
★ ⓘ	★ ⓘ	★ ⓘ	★ ⓘ	★ ⓘ	★ ⓘ	★ ⓘ
Apply						
★ ⓘ						

Running Your Workflow



- Once you've built a workflow you have all the building blocks for a basic deployment
- More complex deployments group together workflows into Pipelines, which are covered in a upcoming course
- To run a just a Workflow you can start it right from the Workflow page just by clicking deploy

Workflow Overview

Name	push-nginx-local
Description	using local dev
Service	nginx-service (Docker Image)
Deployment Type	Kubernetes

▶ Deploy

Individual Deployment Specifics



Start New Deployment ⓘ

X

Application my-first-k8s-install
Workflow push-nginx-local

Artifacts*

Service Build / Version

nginx-service

Tag# 1.9.15-alpine

X

Last Deployed: [library/nginx \(build# 1.11.5-alpine\)](#)

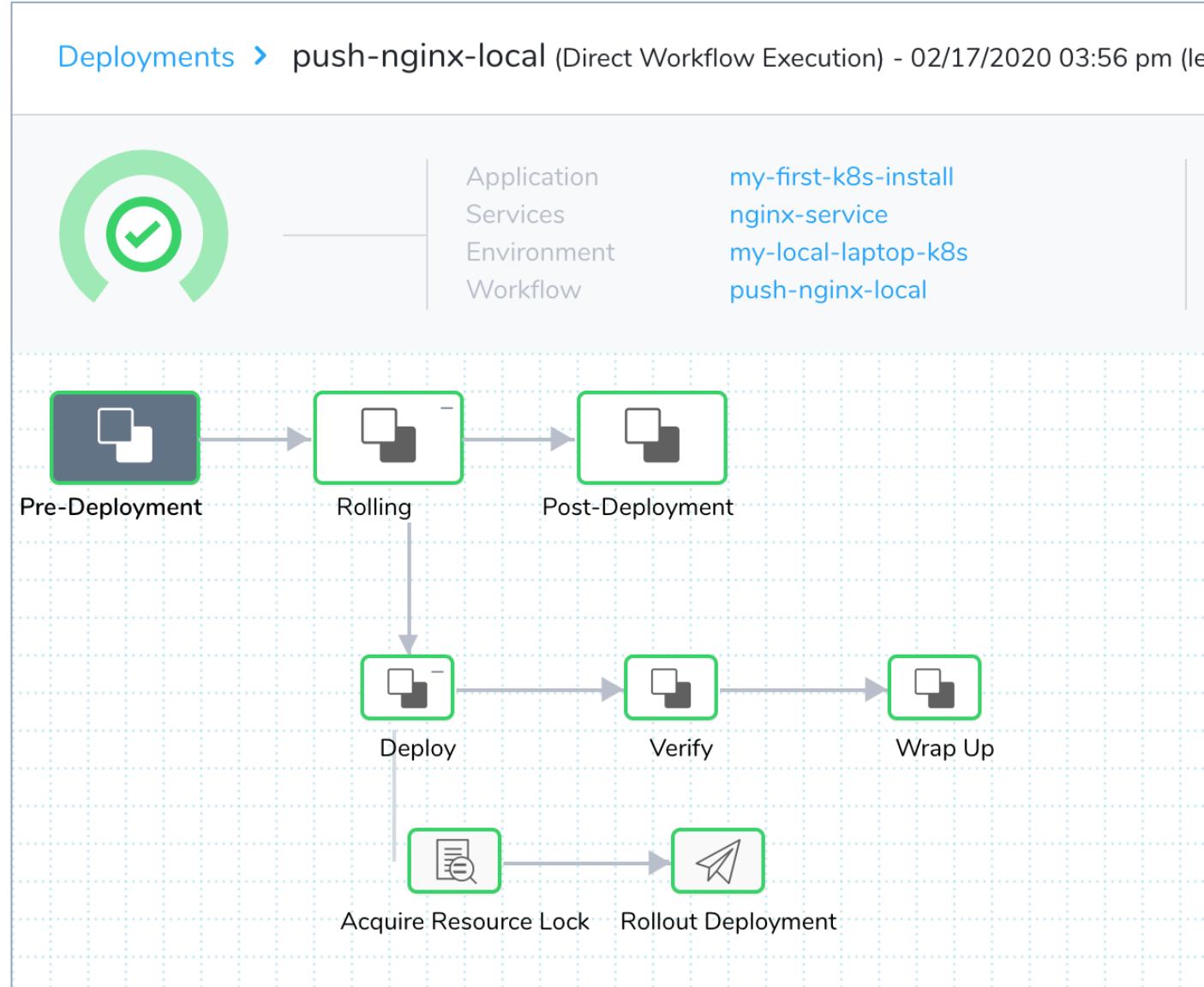
Notes

my first nginx deploy

Send notification to me only

Submit

Running a Deployment



Deployment Details



Deployment

Verify

Rollout Deployment

Details

Started At	02/17/2020 03:56:03 pm
Ended At	02/17/2020 03:56:09 pm
Delegate	my-first-delegate-nudyov-0
Cluster Name	my-first-k8s-env
Namespace	testing-local
Release Name	release-b398fa3c-0da7-3d41-b781-10bda81c3d32
Release Number	1
Load Balancer	http://localhost/

Initialize

```
INFO 2020-02-17 15:56:04    Initializing..  
INFO 2020-02-17 15:56:04  
INFO 2020-02-17 15:56:04  
INFO 2020-02-17 15:56:04    Rendering manifest files using go template  
INFO 2020-02-17 15:56:04    Only manifest files with [.yaml] or [.yml] extension will be processed  
INFO 2020-02-17 15:56:04  
INFO 2020-02-17 15:56:04    Manifests [Post template rendering] :  
INFO 2020-02-17 15:56:04  
INFO 2020-02-17 15:56:04    ---  
INFO 2020-02-17 15:56:04
```

?

Workflow Components



- Workflows are made up of different steps
- Depending on the type of deployment and cloud provider you're using there will be different steps and options available in your workflows
- See the docs for details on non-Kubernetes Workflows

Pre-deployment Steps :

+ Add Step

Deployment Phases

+ Add Phase

Post-deployment Steps :

+ Add Step



Building a Workflow



- Workflow steps and options will vary depending on your type of deployment as well where you're deploying to
- Add actions to your workflow using the Harness UI
- For Kubernetes based deployments you can run K8s specific commands and actions.

▼ Kubernetes (8)

A screenshot of the Harness UI showing a list of Kubernetes actions. The actions are displayed in a grid of cards. Each card contains the action name, a star icon, and an information icon. A dropdown arrow icon is positioned to the left of the first card. Below the grid is a large 'Apply' button with a star icon and an information icon. The background is white, and the cards have a light gray border.

Canary Depl...	Blue/Green ...	Rollout Depl...	Swap Servic...	Traffic Split	Scale	Delete
★ ⓘ	★ ⓘ	★ ⓘ	★ ⓘ	★ ⓘ	★ ⓘ	★ ⓘ
Apply						
★ ⓘ						



Running Your Workflow



- Once you've built a workflow you have all the building blocks for a basic deployment
- More complex deployments group together workflows into Pipelines, which are covered in a upcoming course
- To run a just a Workflow you can start it right from the Workflow page just by clicking deploy

Workflow Overview

Name	push-nginx-local
Description	using local dev
Service	nginx-service (Docker Image)
Deployment Type	Kubernetes

[▶ Deploy](#)

Individual Deployment Specifics



Start New Deployment ⓘ

X

Application my-first-k8s-install
Workflow push-nginx-local

Artifacts*

Service Build / Version

nginx-service

Tag# 1.9.15-alpine

X

Last Deployed: [library/nginx \(build# 1.11.5-alpine\)](#)

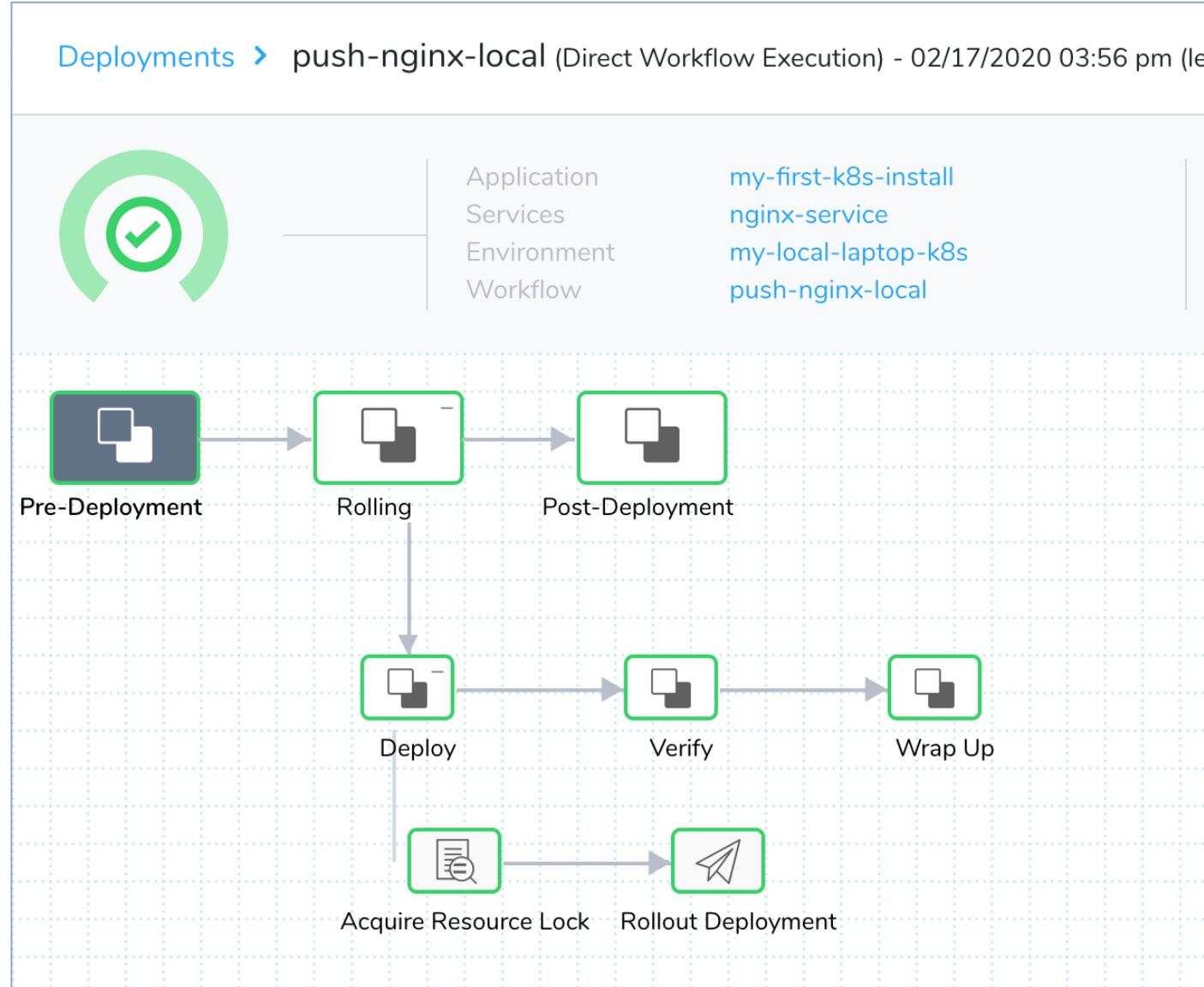
Notes

my first nginx deploy

Send notification to me only

Submit

Running a Deployment



Deployment Details



Deployment

Verify

Rollout Deployment

Details

Started At	02/17/2020 03:56:03 pm
Ended At	02/17/2020 03:56:09 pm
Delegate	my-first-delegate-nudyov-0
Cluster Name	my-first-k8s-env
Namespace	testing-local
Release Name	release-b398fa3c-0da7-3d41-b781-10bda81c3d32
Release Number	1
Load Balancer	http://localhost/

Initialize

```
INFO 2020-02-17 15:56:04 Initializing..  
INFO 2020-02-17 15:56:04  
INFO 2020-02-17 15:56:04  
INFO 2020-02-17 15:56:04 Rendering manifest files using go template  
INFO 2020-02-17 15:56:04 Only manifest files with [.yaml] or [.yml] extension will be processed  
INFO 2020-02-17 15:56:04  
INFO 2020-02-17 15:56:04 Manifests [Post template rendering] :  
INFO 2020-02-17 15:56:04 ---  
INFO 2020-02-17 15:56:04
```

?



Lab 2 Demo

