aws marketplace

EBOOK

Resilience and continuous improvement for cloud-native day 2 operations

How to maintain resilient workloads and continuously refine code for better customer experiences

aws marketplace

CLOUD-NATIVE OPS (DAY 2)

Table of contents

Modernization is a business priority today

To delight customers and win new business, organizations need to build reliable, scalable, and secure applications. That means adopting new technologies, practices, and consuming services as APIs.

As an application development professional, your goal is to deliver business value fast. Modern applications help achieve this goal by separating and decoupling the monolith into smaller functional services—or **microservices**—that focus on one thing and do it well. Each microservice often has its own data store and can be deployed and scaled independently. They represent the real world, where service boundaries equal business boundaries.

This has forced organizations to evolve by giving engineering teams the autonomy to architect, develop, deploy, and maintain each microservice. With this approach, you end up with the ability to make decisions very quickly because your decisions only impact individual services. After all, innovation requires change. You can learn faster by making lots of little changes to drive incremental innovation, rather than waiting to take one giant leap.

What is cloud-native? Why does it matter?

Cloud-native is an evolving term. The vast amount of software that's being built today needs a place to run and all the components and processes required to build an application need to fit together and work cohesively as a system.

The Cloud Native Computing Foundation (CNCF) definition states:

Cloud-native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds.

This definition has to broadly apply to everyone, but not everyone has the same capabilities. This is known as the lowest common denominator problem. It is where you try and appeal to a broader group and their capabilities, but in doing so you also need to limit the capabilities that can be leveraged.

Amazon Web Services (AWS) goes many steps further by providing a broad set of capabilities that belong to a family called serverless. Serverless technologies are more than just AWS Lambda—these services remove the heavy lifting associated with running, managing, and maintaining servers. This lets you focus on core business logic and quickly adding value.

The cloud-native journey: Day two

As we cover the different capabilities your organization needs to acquire to go fully cloud-native, it's useful to view each one as a step in a journey.

The above map is a model for how organizations typically evolve their cloud-native understanding. As your organization or team moves from stage to stage, you are gaining capabilities that make releasing new features and functionality faster, better, and cheaper. In the following sections, we'll be focusing on the continued strategies you need to adopt and maintain after you've achieved cloud-native maturity—aka "Day Two."

Challenges to productivity and continuity

In an ideal world, everything is perfect and works right all of the time: You cross great distances within a short amount of time, business goes as usual, time flows smoothly, and life goes by without any disruption. You have what you need, when you need it. A world that's always on, and always available. But that's in a perfect world.

In the real world, everything fails all of the time. Services that run the world are exposed to disruptions, security risks, and downtime. Imagine a critical service not available when it's needed most. In a world where every second counts, a service that is unavailable for even a fraction of a second can have adverse consequences.

Resilient system design and architecture reduces the possibility of such an event. Resilient infrastructure enables services to bounce back and become available within the shortest time possible. Let's dive in further and understand what resilience is, how we build resilience of the cloud, and how you can help your customers build resilience in their workloads on the cloud.

Modern organizations today face an ever-growing number of resilience-related challenges, as expectations from customers shift toward an always on, always available mindset. Remote teams and distributed systems coupled with the increasing need for velocity of new releases mean an organization and its systems need to be more resilient than ever.

Resilience is the ability of a workload to recover from infrastructure or service disruptions by dynamically acquiring computing resources to meet demand and mitigate disruptions, such as transient network issues or certain misconfigurations.

According to IDC, unplanned system downtime costs Fortune 1000 companies anywhere from $1.25 billion to $2.5 billion annually. In addition to financial impact, companies experience a negative impact to their brand when services are performing poorly or are unavailable due to outages. With resilient systems, organizations are able to offer their end users a consistent experience that they've grown to expect from their brand and increase revenue.

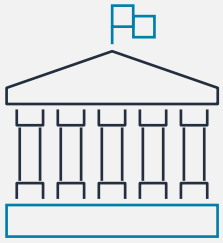
aws marketplace

What is the AWS Well-Architected Framework?

The AWS Well-Architected Framework helps you evaluate your app development practice with a set of questions and design principles applied across six pillars: Operational Excellence, Security, Reliability, Performance Efficiency, Cost Optimization, and Sustainability.

Alongside the Pillars of the AWS Well-Architected framework are AWS Well-Architected Lenses, which provide guidance with a focus on specific industry or technology domains, such as Software as a Service or Financial Services Industry (FSI) for domains, and High-Performance Computing or Serverless for technology domains.

To evaluate the health of your workloads, you answer a set of foundational questions, based on the Framework, Pillars, and Lenses. Recently it was introduced that AWS customers can now create user-defined and -managed Custom Lenses to best align with your organization's industry, operational plans, and internal processes. You can create your own question sets, add context, and identify best practices as relate to your own organization and processes.



Pillars and lenses

Design principles

Questions

Best practices

How can you use the AWS Well-Architected Framework?

Well-Architected (WA) is more than just a set of best practices or a console-based tool. **WA is a mechanism for AWS customers' cloud journey.** It allows you to:

1. **Learn strategies and best practices for architecting in the cloud.**
2. **Measure your architecture using AWS Well-Architected Framework and AWS Well-Architected Lenses, and measure your organization's best practices using Custom Lenses within the Well-Architected Tool.**
3. **Improve your cloud architectures by addressing any high-risk issues identified by leveraging improvement plans, Well-Architected Labs, AWS Partners, AWS solutions architects, and more.**

At AWS, we understand the value of educating customers on architectural best practices to help ensure they are actively thinking about foundational areas that are sometimes neglected. The AWS Well-Architected Framework provides a consistent approach to evaluating architectures and also allows you to measure your workload against your own best practices—in addition to the Well-Architected Framework—and perform associated reviews for all technology and governance needs across your organization.

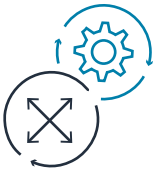
aws marketplace

Pillars of the AWS Well-Architected Framework

Creating technology solutions is a lot like constructing a physical building. If the foundation is not solid, it may cause structural problems that undermine the integrity and function of the building.

If you neglect any of the six pillars of security, reliability, performance efficiency, cost optimization, operational excellence, and sustainability when architecting technology solutions, it can become a challenge to build a system that delivers functional requirements and meets your expectations.

When you incorporate these pillars, they will help you produce stable and efficient systems, allowing you to focus on functional requirements.



Operational excellence

The ability to run and monitor systems to deliver business value and continually improve supporting processes and procedures

Cost optimization

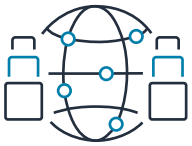
The ability to avoid or eliminate unneeded cost or suboptimal resources

Security

The ability to protect information, systems, and assets while delivering business value through risk assessments and mitigation strategies

Sustainability

Guidance on how AWS can help you reduce your footprint, and best practices you can use to improve the sustainability of your workloads



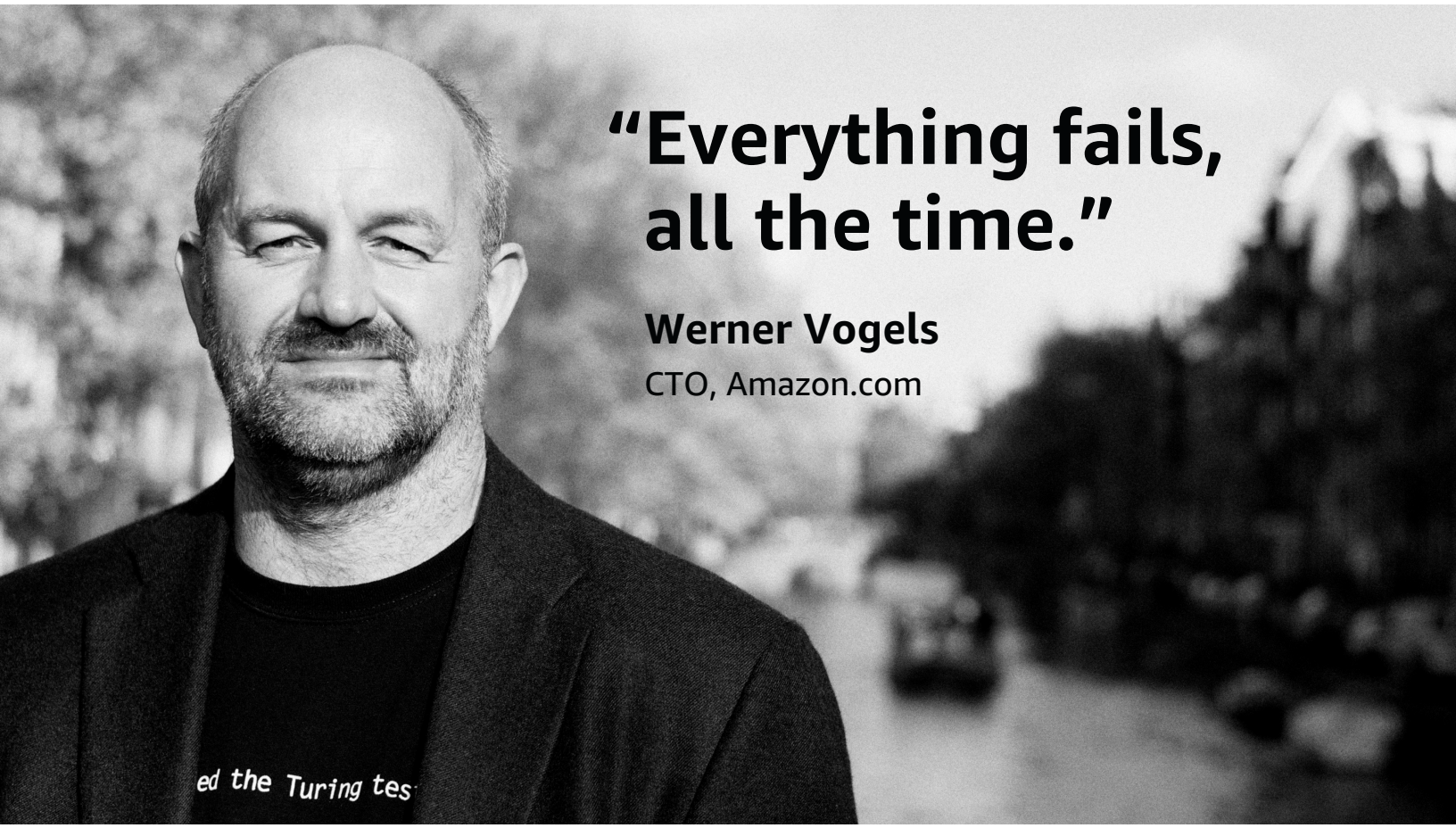
Reliability

The ability of a system to recover from infrastructure or service failures, dynamically acquire computing resources to meet demand, and mitigate disruptions such as misconfigurations or transient network issues

Performance efficiency

The ability to use computing resources efficiently to meet system requirements, and to maintain that efficiency as demand changes and technologies evolve

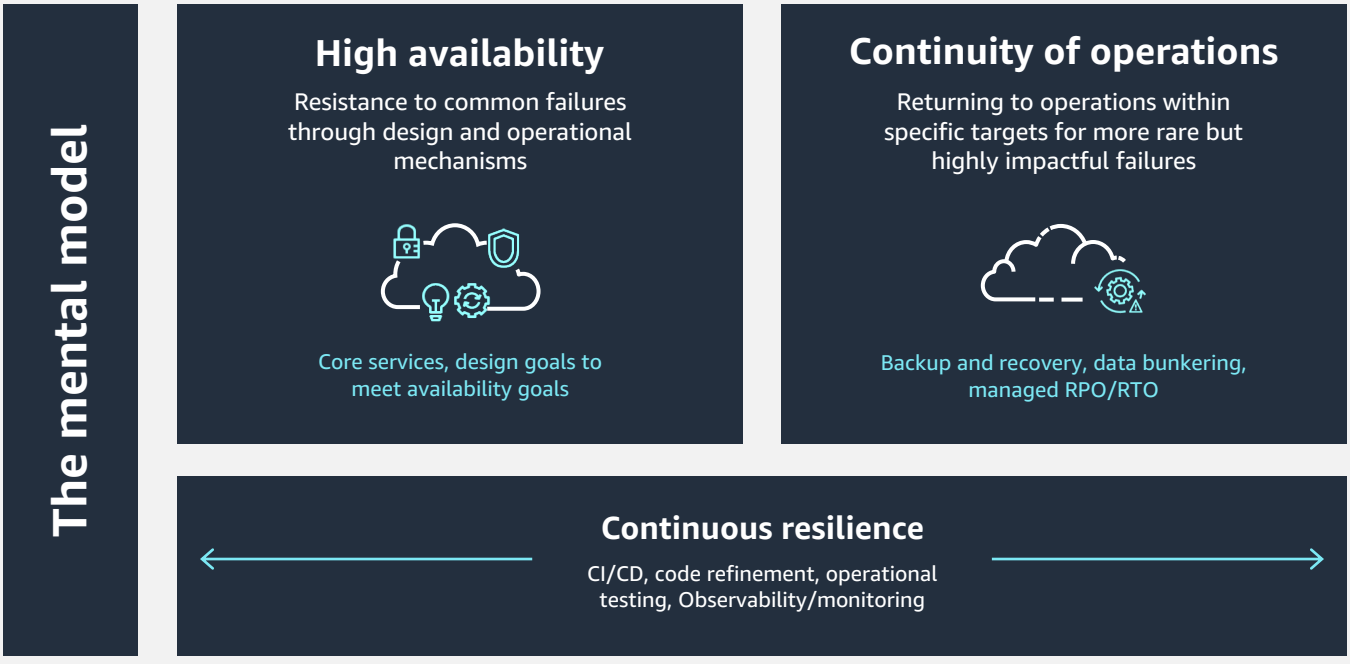
According to Werner Vogels, Chief Technical Officer of Amazon, “Everything fails, all the time.” Any complex system is likely to have a mixture of smaller, more probable failures and larger, more rare failures over its lifespan. Because of this, organizations must expect and plan for such failures and design workloads to recover from failure in a way that minimizes impact to end users.



What is resilience?

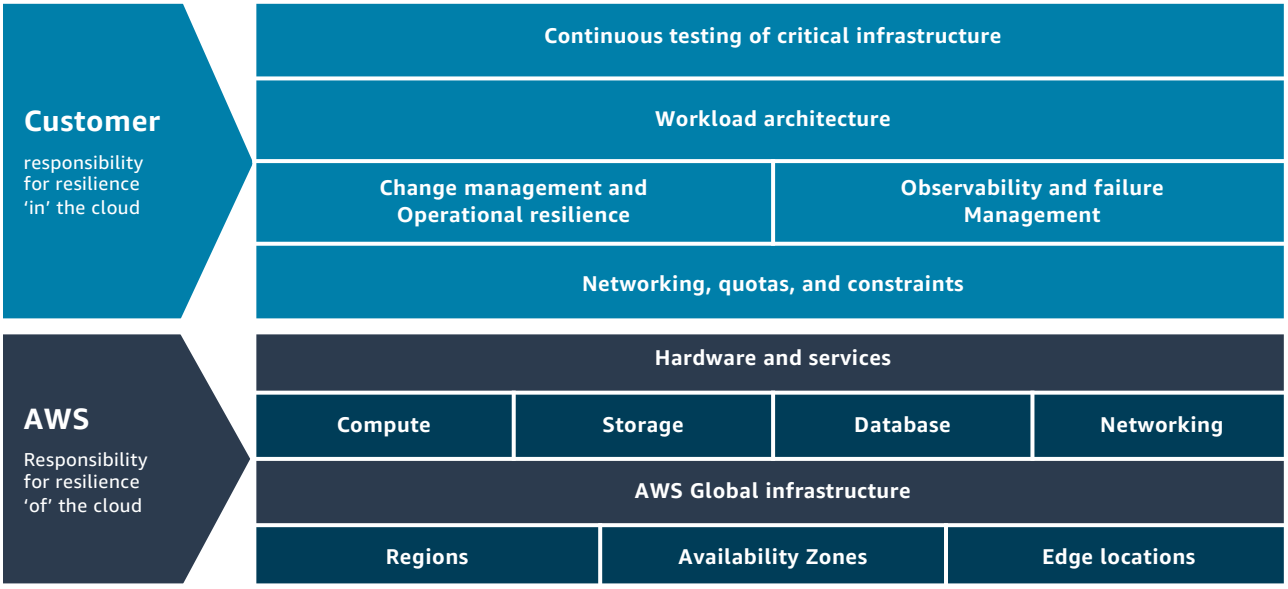
Resilience refers to the ability for workloads—a collection of resources and code that delivers business value, such as a customer-facing application or backend process—to respond and quickly recover from failures. A workload might consist of a subset of resources in a single AWS account, or be a collection of multiple resources spanning multiple accounts.

The mental model you should consider as we go through this information is to think about how you can build systems to be highly available with resistance to common failure modes and how you can recover your system if you run into those rare failure scenarios. Underpinning all of this is the idea of continuous resilience, where you are implementing DevOps practices like Continuous Integration/Continuous Delivery (CI/CD) to automate your delivery pipelines, introducing failures on an ongoing basis to test your system chain and teams for weaknesses, and implementing ongoing Observability and monitoring practices.



The shared responsibility model for resilience

Resilience is a responsibility that is shared between AWS and AWS customers. This shared model can help relieve the customer's operational burden because AWS operates, manages, and controls the components from the host operating system and virtualization layer down to the resilience of the infrastructure. The customer assumes responsibility and management of the workload they have built on AWS. Customers should carefully consider the services they choose, as their responsibilities vary depending on the services used, the integration of those services into their IT environment, and applicable laws and regulations. This differentiation of responsibility is commonly referred to as resilience of the cloud versus resilience in the cloud.



Customer responsibility (resilience in the cloud)

Customer responsibility will be determined by the AWS products that a customer selects. This determines the amount of configuration work the customer must perform as part of making their workload resilient.

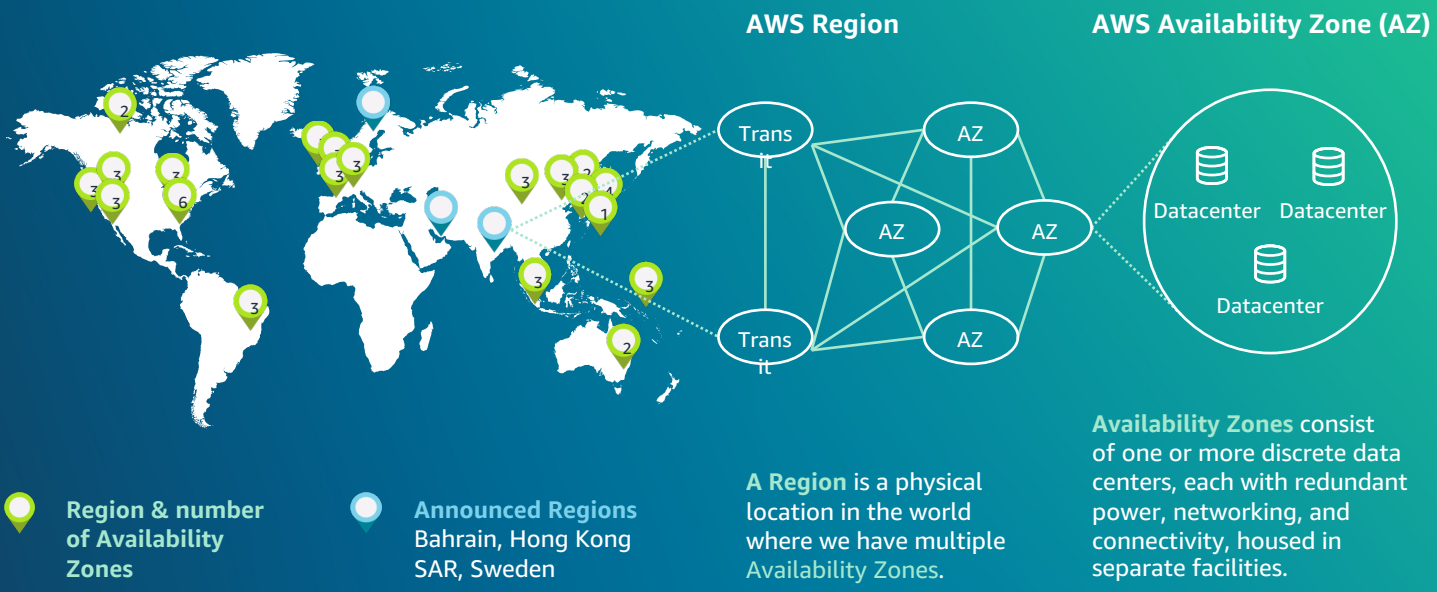
AWS responsibility (resilience of the cloud)

AWS is responsible for ensuring that the infrastructure that runs all of the services offered in the AWS Cloud is resilient. This infrastructure is composed of the hardware, software, networking, and facilities that run AWS products.

Enabling resilience of the cloud

As of this writing, AWS offers over 200 fully featured services from 84 Availability Zones (AZs) across 26 Regions, globally.

- Unlike other cloud infrastructure providers, each AWS Region has multiple Availability Zones (AZ) and each AZ has multiple physically separated data centers. By comparison, some cloud providers claim to have just over 50 regions, yet most of the regions have no AZs and just one data center, so it is not comparable to the multi-AZ, multi-data center AWS Region.
- Each Region also has two independent, fully redundant transit centers that allow traffic to cross the AWS network, enabling Regions to connect to the global network.
- AWS does not use other backbone providers for AWS traffic once it hits our backbone.



Service ownership model

AWS employs a service ownership model, which incentivizes teams to continuously improve their operations. We have organized our engineering and product management efforts to be led by small, multi-disciplinary teams with strong ownership of the services they produce. This ownership brings responsibility for not only designing and launching their service, but also operating it during production and being on-call for issues as they arise.

Operational readiness reviews

AWS is responsible for ensuring that the infrastructure that runs all of the services offered in the AWS Cloud is resilient. This infrastructure is composed of the hardware, software, networking, and facilities that run AWS products.

Safe Continuous Deployment

When AWS deploys software for service updates or rolls out new services, we use safe, Continuous Deployment pipelines. Building automated deployment safety into the release process by using extensive pre-production testing, automatic rollbacks, and staggered production deployments lets us minimize the potential impact on production caused by faulty deployments. For example, updates to an AWS product start small; the update is first rolled out to a single server within an AZ and goes through a designated waiting period to verify no issues have arisen. From there, the update is deployed throughout the rest of the AZ, then to other AZs, then to a single Region, and finally, to remaining Regions.

Correction of Error processes

If any issues arise, we leverage incident management mechanisms such as Correction of Error (CoE) processes to help our teams understand the root cause. After an issue is mitigated, we drive company-wide engineering sprints to ensure the issue is fixed across all AWS products, which reduces the chance of a similar event impacting another AWS product in the future. These learnings are documented and become part of the ORR process, which ensures similar issues do not reoccur.

Resilience on the cloud

Now that we've talked about what AWS is doing to build resilience of the cloud, let's talk about how you can make your workloads more resilient in the cloud. Every organization has a different maturity level when it comes to their resilience journey—some are just beginning, while others may be more advanced. The concepts we're going to outline provide a good baseline of things you should consider during your resilience journey, but can be expanded upon depending on your business objectives and available resources to execute your resilience strategy.

Defining and measuring resilience goals

With resilience goals, one size does not fit all. The AWS Well-Architected Framework encourages you to define availability and continuity of operations goals based on an individual workload's criticality to the business versus applying a single goal across all workloads.

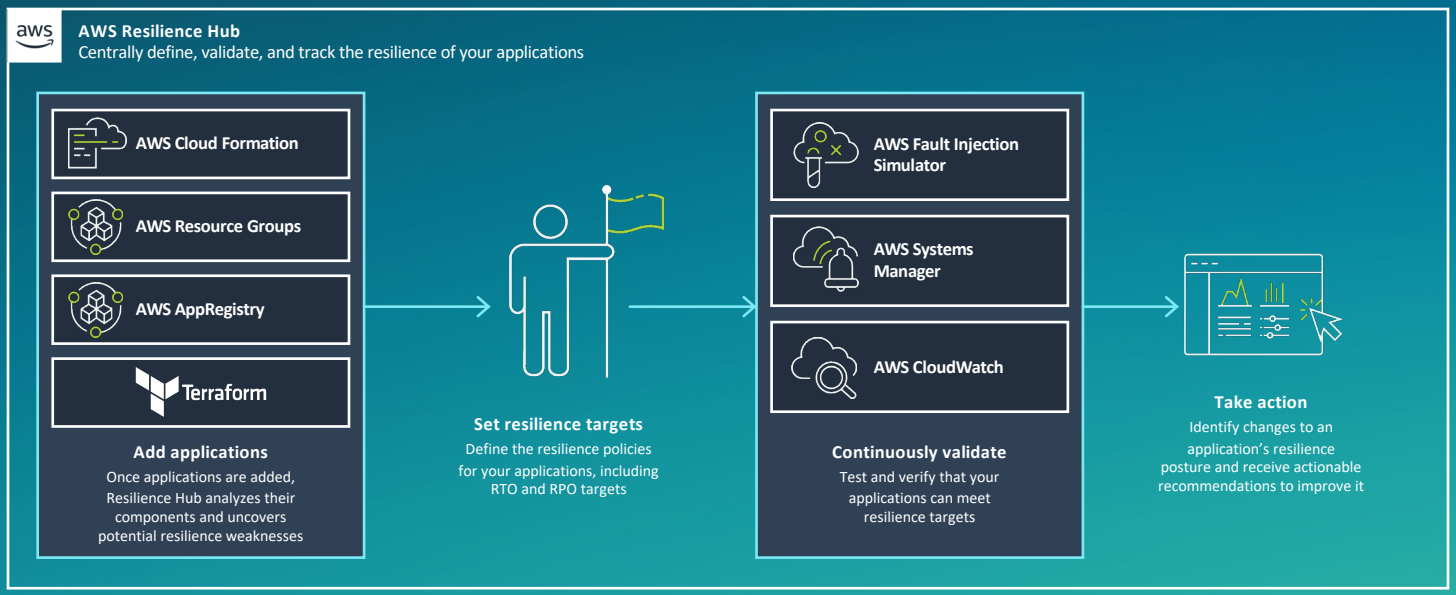
Establishing goals for high availability is only half the equation, though. It's also imperative to create continuity of operations goals, such as recovery time objectives (RTO) and recovery point objectives (RPO). RTO establishes the maximum time acceptable for a workload to recover from a failure, while RPO establishes the maximum acceptable window of time for which data might be lost after an incident.

Once defined, these RTO and RPO targets can be input into AWS Resilience Hub, a central place for you to assess, validate, and measure the resilience posture of your workloads against your defined goals.

AWS Resilience Hub key capabilities:

- Continuously validate and track application resilience to reduce outages
- Evaluate resilience targets
- Identify and resolve issues before they occur in production
- Optimize business continuity while reducing recovery costs

AWS Resilience Hub process flow



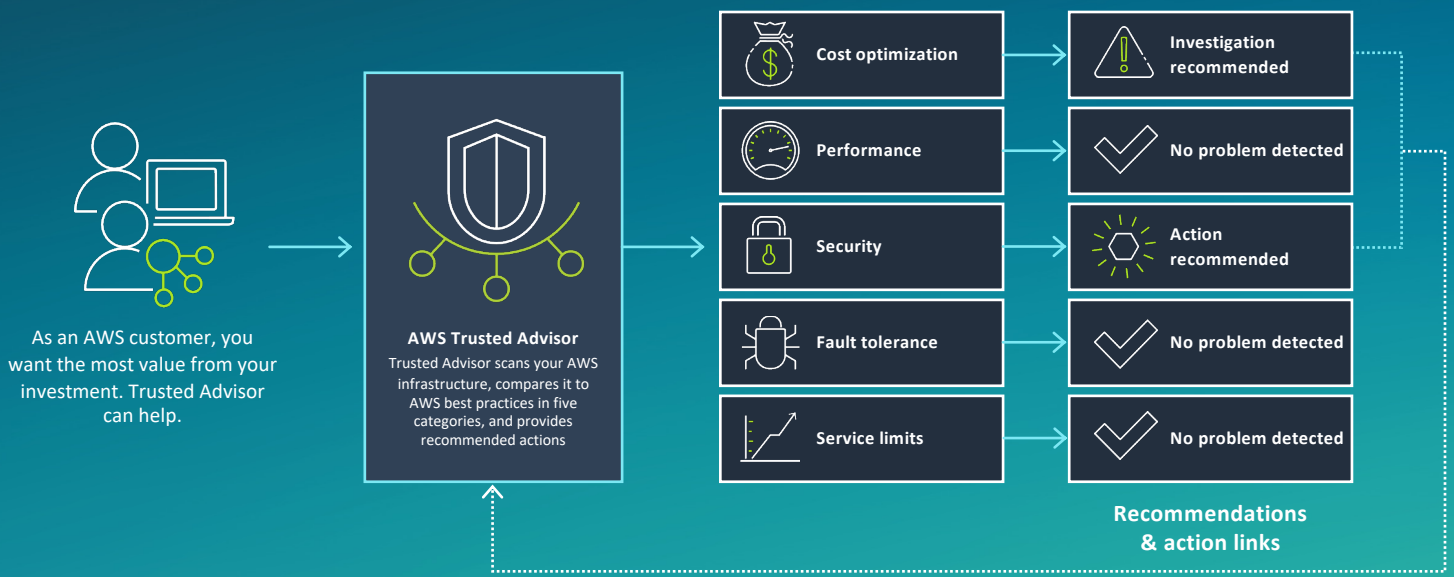
Identifying and mitigating risks

After you've defined your goals, the next step is understanding what your workload's current resilience posture is and if there are any issues that need to be fixed. The **AWS Well-Architected Tool**, based on the AWS Well-Architected Framework, runs your workload against 66 resilience best practices, identifies risks, and generates an action plan for you to remediate those risks.

For example, the AWS Well-Architected Tool recommends that you use a microservice architecture to make components smaller and simpler, which allows you to differentiate the availability needs of different services within your workload. The AWS Well-Architected Tool also categorizes any identified risks into low, medium, and high-risk issues, so you can prioritize remediations accordingly.

While AWS Well-Architected and AWS Resilience Hub help improve resilience of your workloads, **AWS Trusted Advisor** helps improve the resilience of your AWS account and its resources. As illustrated in the graphic below, AWS Trusted Advisor programmatically conducts a series of automated checks against five key areas—Cost Optimization, Performance, Security, Fault Tolerance, and Service Limits—and provides recommendations that enable you to follow known best practices. Within the Fault Tolerance category alone, AWS Trusted Advisor features 23 checks that can help identify risks and improve resilience.

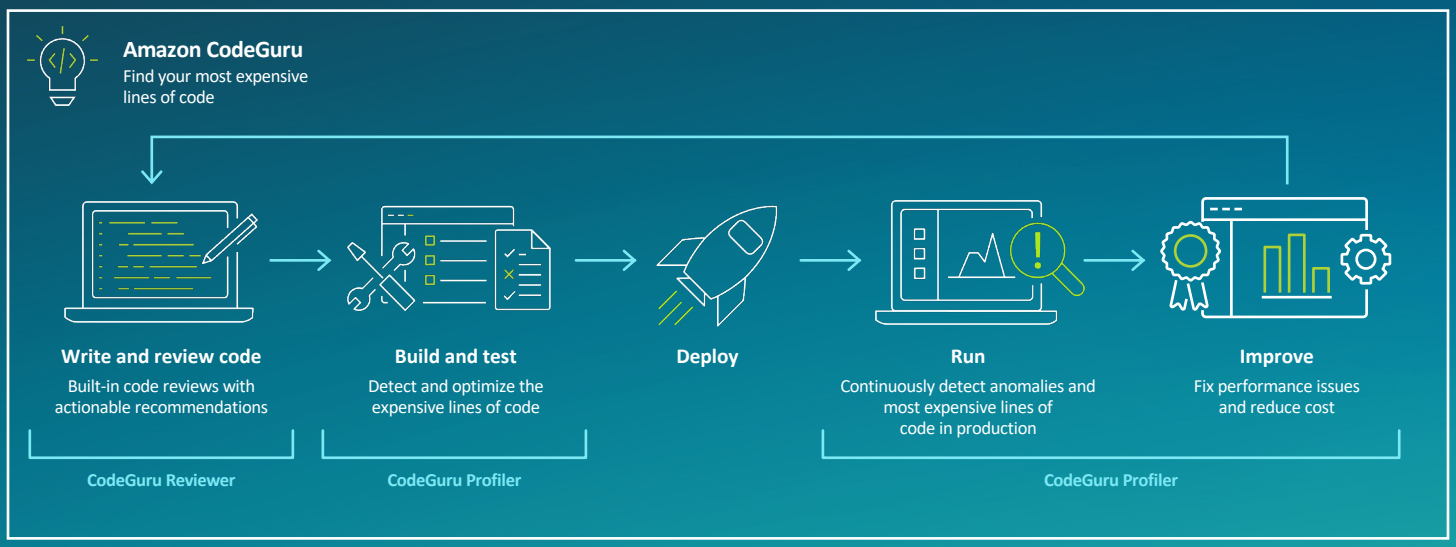
AWS Well-Architected Tool process flow



Continuous code refinement

Part of building a highly available, resilient application is being able to identify and resolve issues in your code before it's deployed. While reviewing your code can be done manually, the success of the review largely depends on the reviewer's expertise and ability to identify potential issues. AWS provides code reviewing tools that not only augment the commentary on code reviews, but also recommend best practices around resilience. Prior to any deployment, it's a good idea to leverage **Amazon CodeGuru Reviewer**, which uses machine learning and automated reasoning to identify critical issues, security vulnerabilities, and hard-to-find bugs during application development. Tools like Amazon CodeGuru Reviewer help you *shift left* on the previously mentioned resilience continuum to stop issues before they start.

Amazon CodeGuru process flow



Continuous Integration/ Continuous Deployment (CI/CD)

Beyond code configuration issues, failures also commonly occur during code deployment. Manual deployment processes introduce unnecessary risk, which is why using configuration management systems that automate as much as possible are key in creating resilient architectures. One way to automate configuration management is through implementation of continuous integration and continuous delivery, a popular DevOps practice commonly known as CI/CD that automates the testing and deployment of code, enables teams to catch potential defects earlier in the process, and automate recovery.

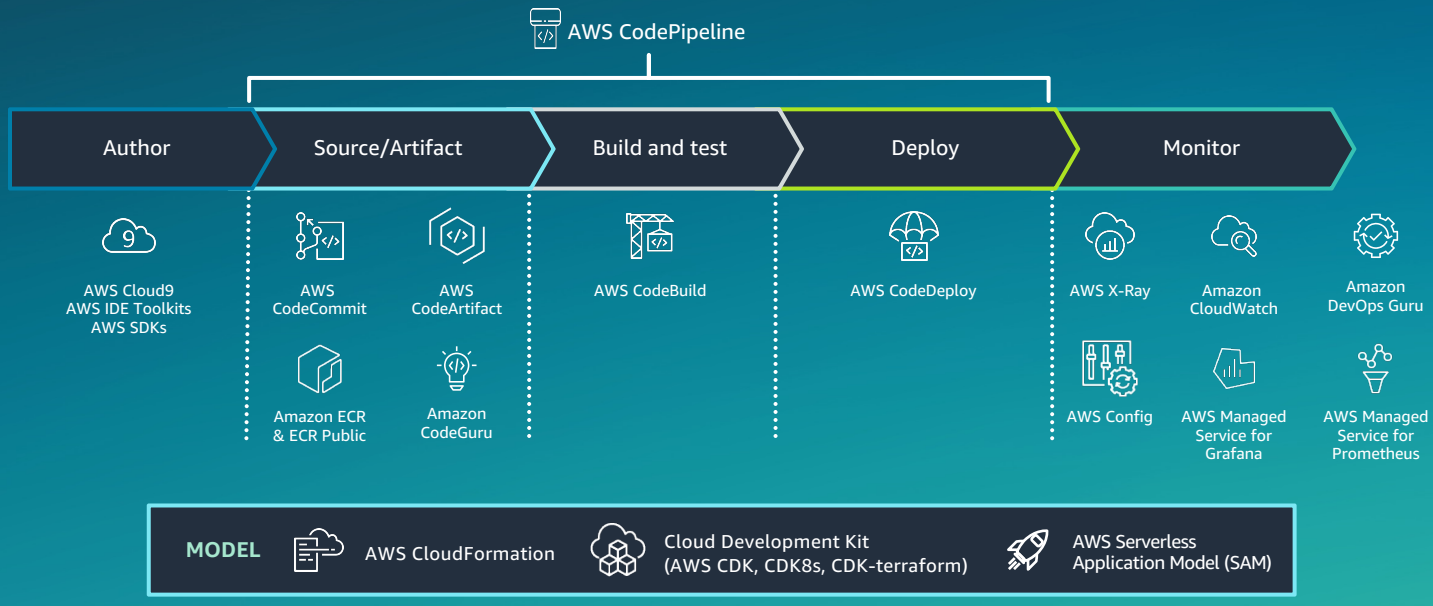
Automate as much as possible

Remove opportunity for manual errors during deployment

AWS developer tools

On AWS, you can build CI/CD pipelines using services within the AWS Developer Tools portfolio, including AWS CodeCommit, AWS CodeBuild, AWS CodePipeline, AWS CodeStar, and AWS CodeDeploy. These services—combined with best practices from the AWS Well-Architected Operational Excellence pillar—can help you address risks in your environment, and limit errors from manual processes.

AWS services mapped to pipeline phase



Continuous testing

Amazon has been purposefully injecting controlled failures into limited environments since the early 2000s to increase resilience and ensure readiness under the most adverse of circumstance. Another popular testing methodology is the concept of resilience **game days**, which simulate a failure or event to test the responses of your systems, processes, and teams.

On a game day, your team actually takes the actions they would during a failure event, which helps build muscle memory for the best way to respond. Game days can be carried out with replicas of your production environment, which enables you to safely test in an environment that resembles reality. More guidance on running game days can be found as part of the AWS Well-Architected Reliability pillar.

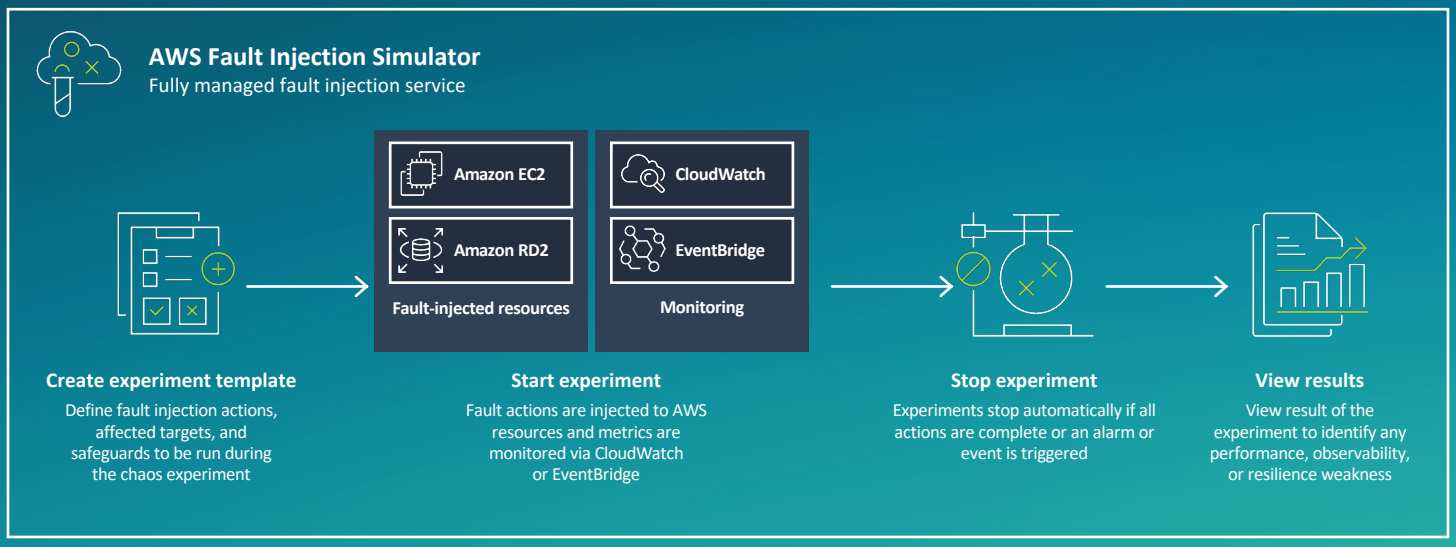
Expect the unexpected

Simulate real-world failures to see how your teams and systems react

AWS Fault Injection Simulator (AWS FIS)

AWS FIS is a fully managed service that simulates real-world failures, such as network errors or having too many open connections to a database. Fault injection experiments help teams create the real-world conditions needed to uncover hidden bugs, monitoring blind spots, and performance bottlenecks that are difficult to find in distributed systems.

AWS FIS process flow



aws marketplace

Continuous Observability

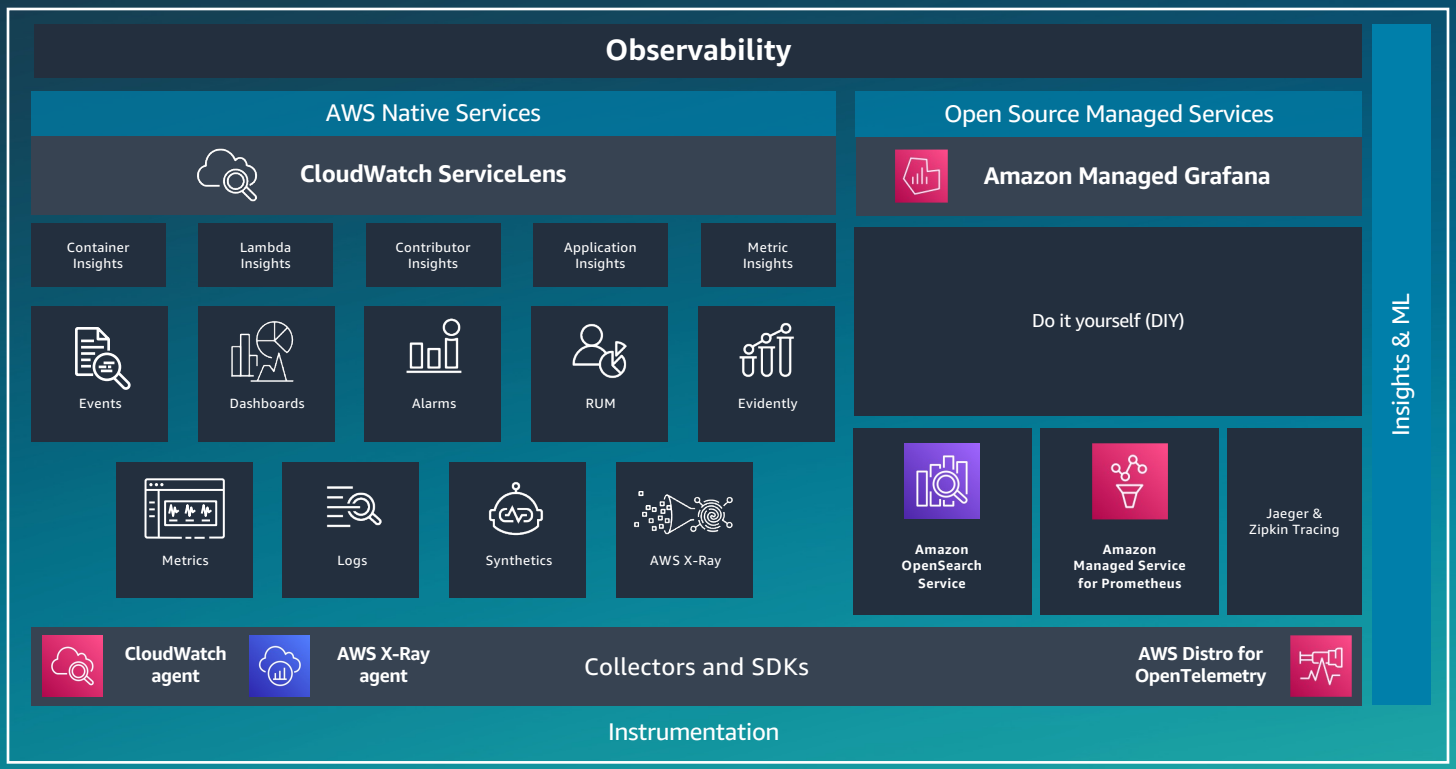
Implementing Observability practices is imperative for resilient architectures; if you can't measure it, you can't improve it.

With AWS availability monitoring and management services such as AWS CloudTrail, Amazon CloudWatch, Amazon DevOps Guru, and AWS X-Ray, you can run assessments, configure alarms and synthetic user requests, detect behaviors that deviate from normal operating patterns, get status updates, and troubleshoot issues. AWS Resilience Hub and AWS CloudTrail also maintain an audit trail log of testing, failure, and recovery events to help meet regulatory and compliance requirements. You can also use the AWS Health dashboard to understand when AWS events affect your workload.

If you can't see it, you can't fix it

Monitor key business metrics using Observability practices

AWS observability services



Recovering quickly

Failures are inevitable, but preparation can help minimize their impact.

To failover gracefully, the underlying data that supports your application must be resilient, and proactively implementing data resilience strategies such as replication, redundancy, and backups, can help. Services like Amazon Simple Storage Service (Amazon S3), Amazon Aurora Global Database, and Amazon DynamoDB provide multi-Region support for data replication.

Amazon S3 provides 99.999999999 percent durability, and services like Amazon RDS provide redundancy and durability of data by storing them across multiple AZs.

Key tenets for cloud-native ops

In the previous sections you learned about the tools and strategies available to developers that help ensure their cloud-native application development practice is resilient and ready to evolve with ever-changing challenges and customer needs. We explored how:

Distributed systems and the need for fast release times presents challenges

The shared responsibility model for resilience works

The AWS Well-Architected Framework helps you evaluate the health of your workloads

The service ownership model incentivizes teams to continuously improve their operations

Resilience refers to the ability for workloads to respond and quickly recover from failures

To better identify and mitigate risk

aws marketplace

Tools for building cloud-native

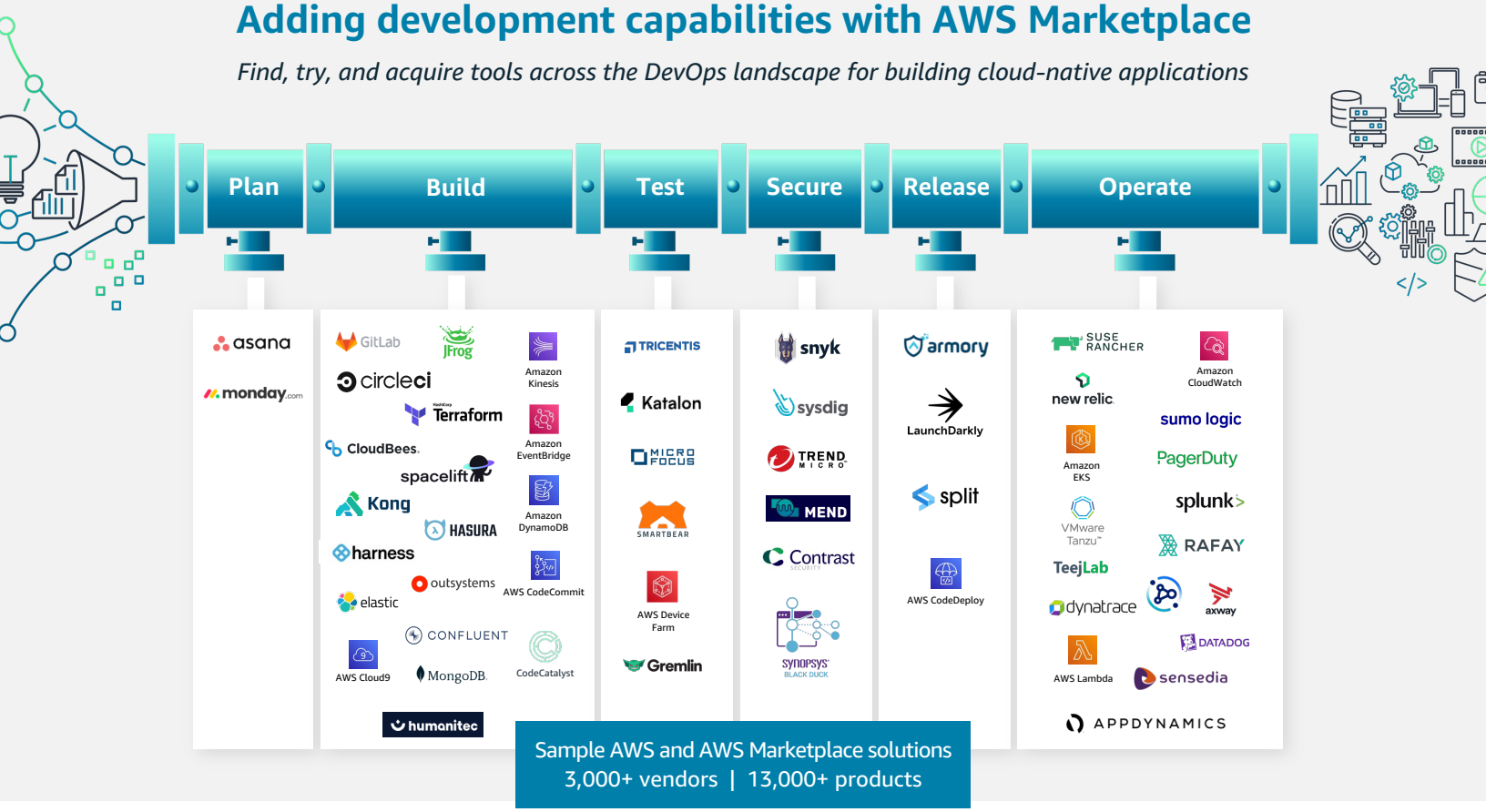
In this section, we'll look at some of the best-fit tools you could use to achieve the tenets discussed in the previous section. At AWS, we've long been believers in enabling builders to use the right tool for the job—and when you build with AWS, you're provided with choice. You can build using the native services AWS provides or use AWS Marketplace to acquire third-party software offered by AWS Partners to take away the heavy lifting and allow your development teams to focus on delivering value to customers.

At this stage in your cloud-native journey you'll need several components that:

- Reduce overhead of managing and maintaining a platform that can provide self-service environments
- Provide full Observability of your cloud operations
- Enable you to react effectively to incidents and offer a way to reduce the time and effort it takes to recover

Adding development capabilities with AWS Marketplace

Find, try, and acquire tools across the DevOps landscape for building cloud-native applications



Sample AWS and AWS Marketplace solutions
3,000+ vendors | 13,000+ products

AWS Marketplace is a cloud marketplace that makes it easy to find, try, and acquire the tools you need to build cloud-native. More than 13,000 products from over 3,000 Independent Software Vendors are listed in AWS Marketplace—many of which you can try for free and, if you decide to use, will be billed through your AWS account.

Complexity of Kubernetes evident as enterprises scale

For most businesses, the biggest source of competitive advantage is the pace of innovation. It's no secret that the speed at which an organization can improve their products and services, capturing the needs of constantly evolving markets, is critical to success. To this end, release cycles are accelerating toward a daily cadence—or even multiple times a day. And as the average number of clusters has continued to grow—now at over 25—cross-functional platform teams and operations practices struggle to solve for a variety of common challenges.

Why? Top reasons we're seeing modernization slow include:

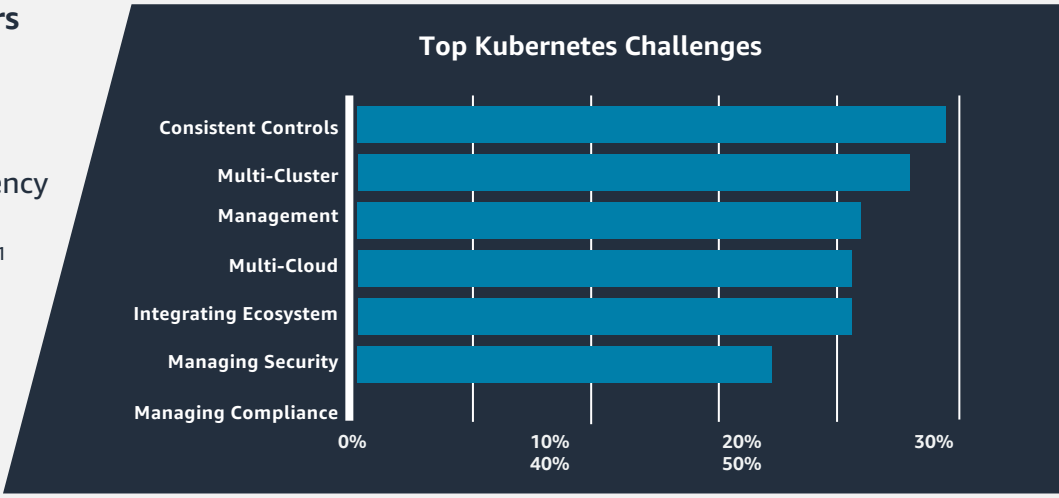
- **Difficulty maintaining consistent controls**
- **Cluster management overhead increasing with the number of clusters**
- **Challenges related to enabling services across data centers and clouds**
- **A complete solution requires integration from a breadth of Kubernetes tools and services**
- **Solving for security and compliance needs**

As operational costs, complexity, and resource needs grow out of control, developing a platform strategy becomes more and more obvious as a way to tackle these challenges.

Complexity factors for containers

Survey respondents say **development release cycles are accelerating,** creating a dependency on automation—however, automation is not keeping pace. [1]

The number of clusters are growing rapidly: "53% of production users are **running 26 or more clusters.**"[2]



[1] CNC`F Survey 2020, https://www.cncf.io/wp-content/uploads/2020/11/CNCF_Survey_Report_2020.pdf
[2] The State of Kubernetes 2020, https://k8s.vmware.com/state-of-kubernetes-2020/
[3] Container Journal, September 22, 2021

DIY Kubernetes requires massive investment

So, let's take a closer look at what teams need to solve for when building an enterprise-grade platform. Just to get a single app running on a single infrastructure stack, there's an incredible amount of heavy lifting. Add the need to move out of the data center to one or multiple public clouds and the problem exacerbates because the same team needs to know the intricacies and nuances of each cloud infrastructure.

Some of the most common DIY challenges include:

Kubernetes skills gap	Most organizations have trouble finding and developing specialized skills to successfully manage and operate Kubernetes.
Managing open-source changes	Keeping up with the newest versions of open-source projects takes away from a team's ability to focus on their core business.
Time and resource investment	It takes months, sometimes years to build and adopt a Kubernetes platform comprised of infrastructure and development workflows.

And once a platform is built, capabilities for automation, security, governance, and visibility are critical to meeting SLAs.

Automation	Most organizations have trouble finding and developing specialized skills to successfully manage and operate Kubernetes.
Security	Keeping up with the newest versions of open-source projects takes away from a team's ability to focus on their core business.
Visibility	It takes months, sometimes years to build and adopt a Kubernetes platform comprised of infrastructure and development workflows.
Governance	Enforce procedures, policies, and rules to help achieve compliance regulations and auditability.

Can your organization invest the time required to research, test, and integrate every component to operationalize a platform? What additional value can your team provide beyond a standard cluster?

aws marketplace

Streamlining Kubernetes operations with Rafay

Adopting Kubernetes and other cloud-native technologies increases agility, accelerates software delivery and supports digital transformation. However, the Kubernetes ecosystem is crowded and complex, shifting time and resources away from generating revenue and differentiation.

Rafay Kubernetes Operations Platform

RAFAY

Try it with AWS Marketplace ›

Watch a demo ›

Start a hands-on lab ›

Rafay's purpose-built platform provides a set of tools and services that enable unified management, operations, and governance across any distribution and any infrastructure—both for clusters and applications.

Many platform teams have provided or are in the process of providing a Shared Service Platform (SSP) to their organization. The SSP approach delivers a single system that manages the fulfillment and operations of all Kubernetes-related services to the rest of the organization. By centralizing the delivery of Kubernetes-related services, enterprises better standardize workflows, increase automation, enable self-service, and optimize application delivery and support for multiple teams across the enterprise.

Key benefits include:

- Standardization of cluster and application configuration
- Centralized and automated cluster and application provisioning
- Unified interface for streamlined cluster lifecycle management
- Efficient and repeatable DevOps workflows
- High level of control, security, policy compliance, and audibility
- Developer, QA, and Ops self-service enablement
- Better collaboration between developers, QA, DevOps, and SRE/Operations
- Faster issue triage and lower support costs, thus lower median time to remediation (MTTR)
- Much lower TCO for Kubernetes management compared to multiple, siloed services

The Rafay platform is made up of six key services:

Multi-cluster management service enables lifecycle management and blueprinting for any Kubernetes distribution, including managed services such as Amazon EKS and runs on any infrastructure, including your data centers, AWS, and remote/edge locations.

Zero-trust access service enables controlled, audited access for developers, SREs, and automation systems to all Kubernetes infrastructure with just-in-time (JIT) service account creation and user-level credentials management

Visibility and monitoring service is key to development, operations, and security/governance teams, helping them visualize and monitor with contextual dashboards

Backup and restore service enables disaster recovery and migrations both for the Kubernetes control plane and application data

Policy management service is built on the Open Policy Agent (OPA) framework helping enable policy management for security and governance

GitOps service enables infrastructure orchestration and application deployment through multi-stage, Git-triggered pipelines

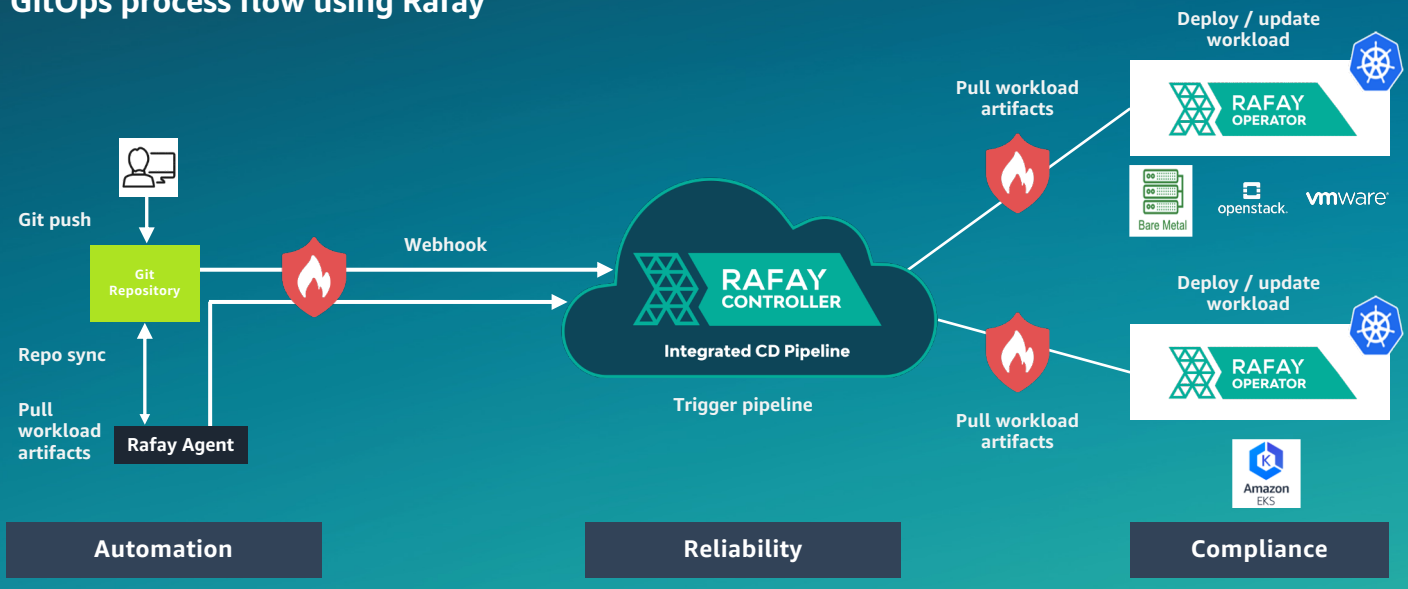
GitOps service by Rafay

Given that 92 percent of organizations use containers in production (according to CNCF Survey Report 2020), the need to rapidly deliver products and features in a manner that minimizes effort and risk is critical to remain competitive in today's market.

GitOps has emerged as a natural evolution of modern software development practices like DevOps, Infrastructure as Code (IaC) and CI/CD principles, specifically for organizations that are building microservices deployed across distributed containers and orchestrated by Kubernetes.

The importance of automating the delivery of containerized applications and infrastructure leveraging a GitOps operating model and workflows is a validated best practice. GitOps uses Git as the system of record for declarative infrastructure and applications, such that all changes are made with pull requests. Enabling developers to use familiar tools to make pull requests is key to accelerating and simplifying both application deployments and operational tasks for Kubernetes.

GitOps process flow using Rafay



aws marketplace

Key benefits include:

- Modernize continuous delivery pipelines by standardizing on Git
- Make it easy to deploy and manage applications and clusters
- Ensure software is always current and up to date with a fully managed SaaS service
- Scale GitOps with manageable multi-stage pipelines
- Simplify pipeline logic with support for both Kubernetes and non-Kubernetes resources
- Multi cluster deployments across data centers, CSPs, and the edge
- Application and cluster templates enabling sophisticated workflows

Rafay for GitOps:

Enables continuous delivery through automated deployment, monitoring, and management via Git helping

Minimizes the complexity of operating Kubernetes clusters with automation and continuous deployment pipelines

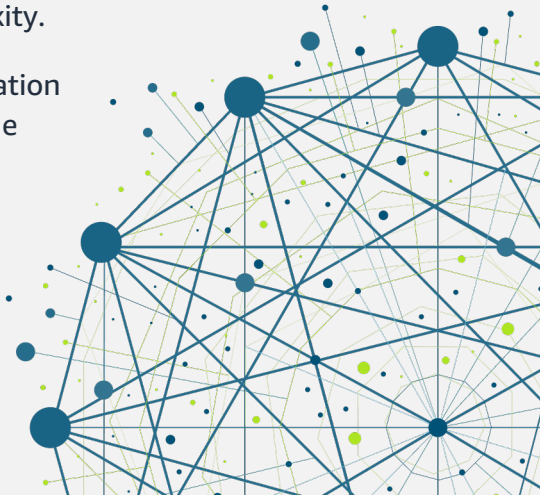
Improves deployment reliability with automated drift detection “and multi-stage pipelines

Helps ensures compliance and guarantee the desired state with built-in workflows, approvals, and governance

Increased complexity creates the need for Observability

Systems are no longer a single server in a single environment. The reality is that with cloud-native, systems are composed of multiple applications and services running together to solve business problems. And the evolution of applications have made them highly distributed, which addresses scalability and resiliency but comes at the cost of added complexity.

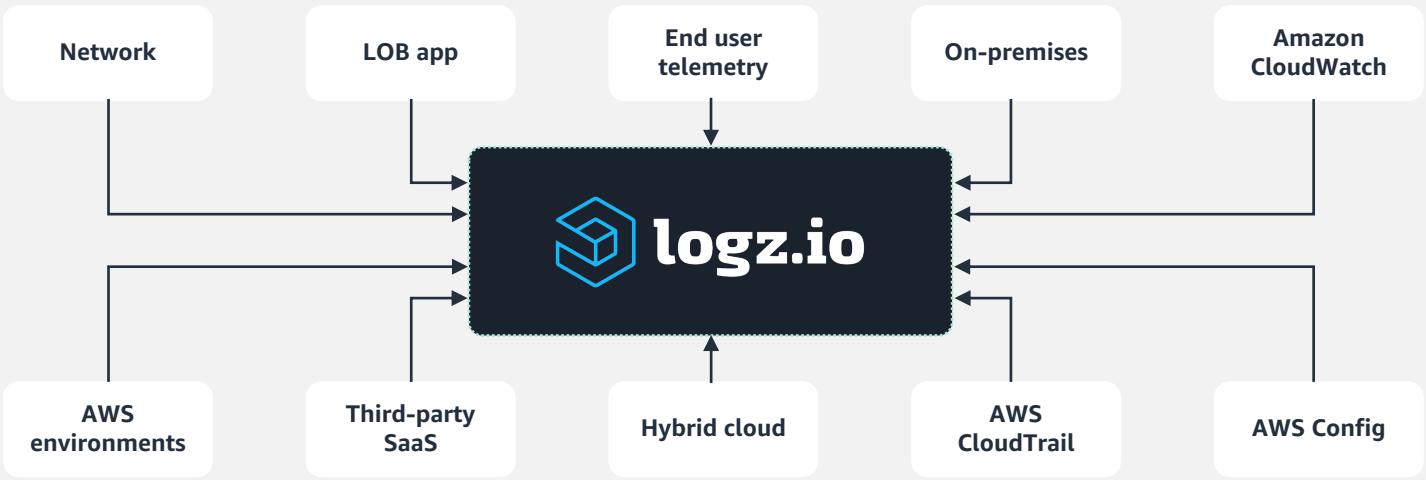
Observability is important in distributed systems because you need information from these different applications to understand a complete picture. Imagine the spherical graphic below is a service, and all the different dots are components that make up the service. If a user reports an issue, how would you pinpoint in this diagram where the issue occurred?



Centralized data collection and visibility

To make this journey more manageable, start with an application or service you want to observe. Armed with different personas and their needs, work backwards from there and determine what data sources will get you the information you need. The diagram below illustrates just a few examples of common data sources you should consider. After you compile a list of the data sources you require, you'll be in a better position to determine which Observability platform will best serve your needs.

Example of centralized data collection solution



aws marketplace

You can choose from multiple options to integrate with any AWS product:

Open-source technologies

Use tools like Fluentd, Prometheus, and OpenTelemetry to collect data from distributed infrastructure and applications.

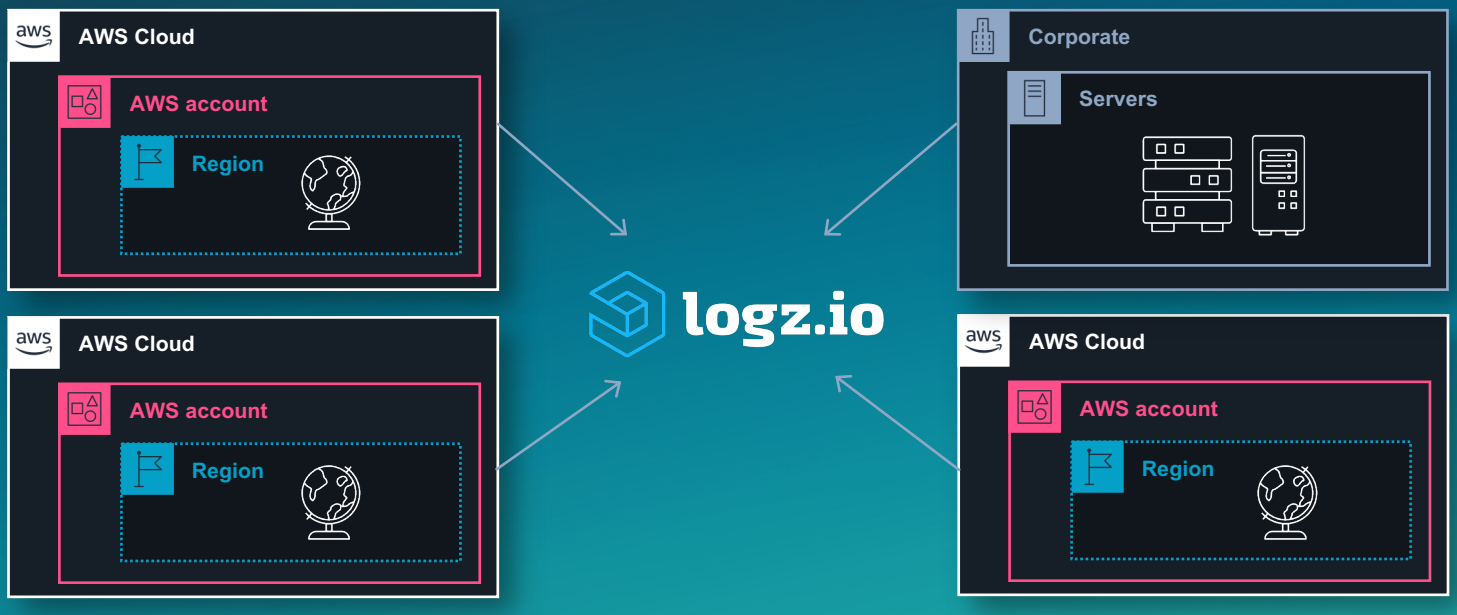
AWS integrations

Collect monitoring data in Amazon CloudWatch or Amazon S3 and send it to Logz.io.

Code libraries

Send data directly from your code with libraries in Javascript, Go, Python, and many other languages.

Multi-account or hybrid Observability example

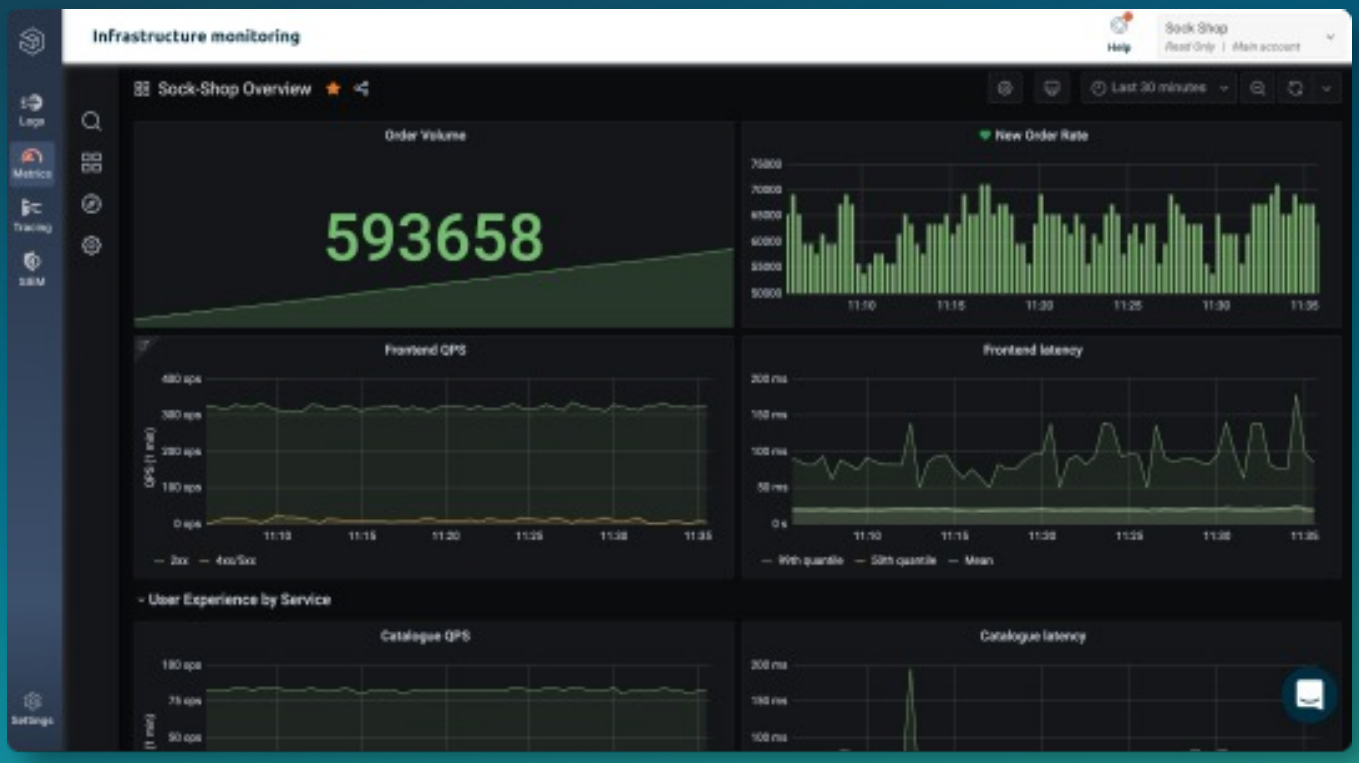


aws marketplace

Scale to accommodate any AWS workload with Logz.io

Logz.io's SaaS platform will seamlessly scale up and down alongside your AWS workloads to ensure constant visibility into the health and performance of your infrastructure and applications. Logz.io can handle any load—in multiple regions—without requiring any effort from your team. It's highly secure and compliant with standards like PCI DSS, GDPR, HIPAA, and SOC 2.

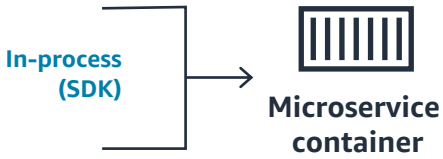
Example of a Logz.io dashboard



Adding Observability to containers

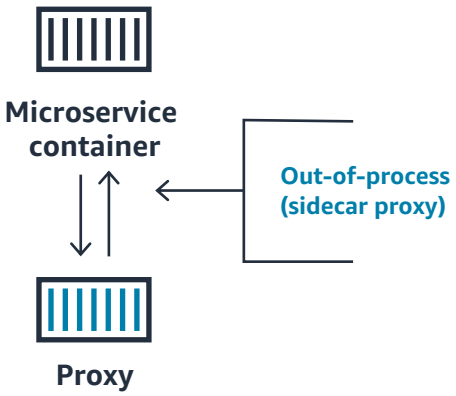
To add Observability to containers, customers are using one of two options: **In-process using an SDK** or **out-of-process using a sidecar proxy**.

Option 1



Option 1 is to use an in-process SDK and add code that is embedded with the application. This method is also referred to as agentless. This option tends to be more heavyweight and requires application code changes and retrofitting. It's also language-dependent, so you need to ensure the language you choose is supported. Plus, the resources }used would have to support this style of instrumentation.

Option 2



Option 2 uses an out-of–process method also known as a sidecar. A sidecar runs as a separate proxy container that manages all the communication outside of the microservice container. The sidecar method allows for decoupling of the operational logic and the SDKs.

A huge benefit of the sidecar pattern is that it provides a straightforward way to implement logging, tracing, and metrics without having to retrofit code or apply an SDK. They are language independent and can be used as a method to capture telemetry data for applications and services where you might not have access to instrument the programming code.

But sidecars can be difficult to scale, which is why we recommend AWS App Mesh for service-level communications. AWS App Mesh is a service mesh that provides a dedicated infrastructure layer for your applications. It allows you to transparently add capabilities like observability, traffic management, and security, without adding them to your own code.

An example application of option 2 would be to use **Rafay** and send all logs, tracing, and metrics to **Logz.io**

aws marketplace

Respond to failure

As we've mentioned several times, not everything will go 100 percent as planned. You can use:

PagerDuty

A SaaS incident response platform for IT departments—along with AWS products to more easily identify and respond to issues.

Incident response

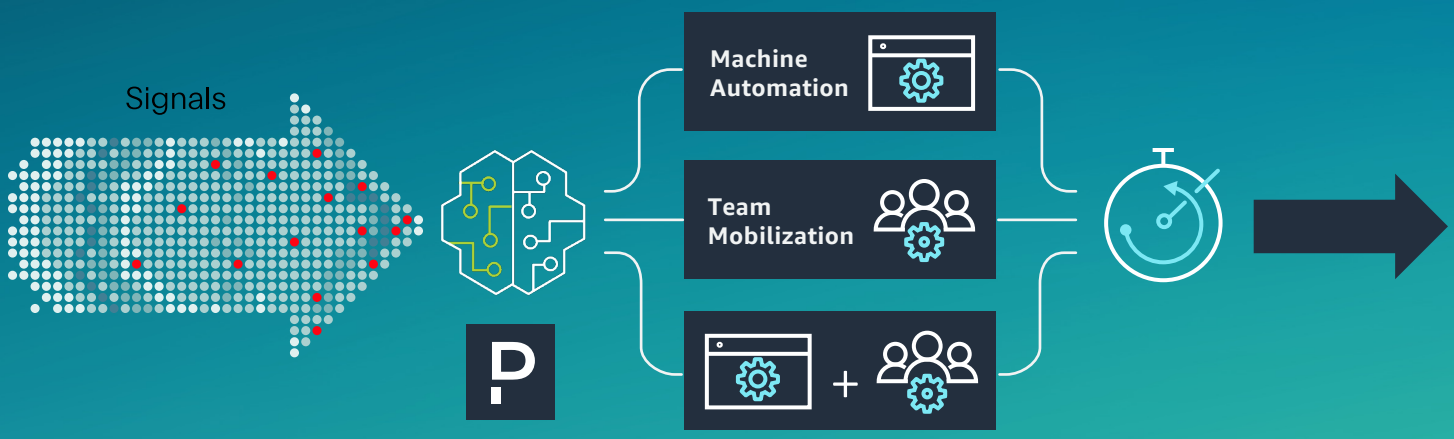
Design the appropriate response for any impact level—mobilize responders, engage stakeholders, and send status updates. Response automation can be executed either with a single tap from any device or automatically for mission-critical services.

Stakeholder communication

Keep business stakeholders like IT management, support, and executives in the know—without interrupting responders—by providing clear business impact information at scale and in real time. Communicate status updates on incident resolution progress to arm stakeholders with the context they need to orchestrate a business response and minimize customer impact.

IT service management (ITSM) integration

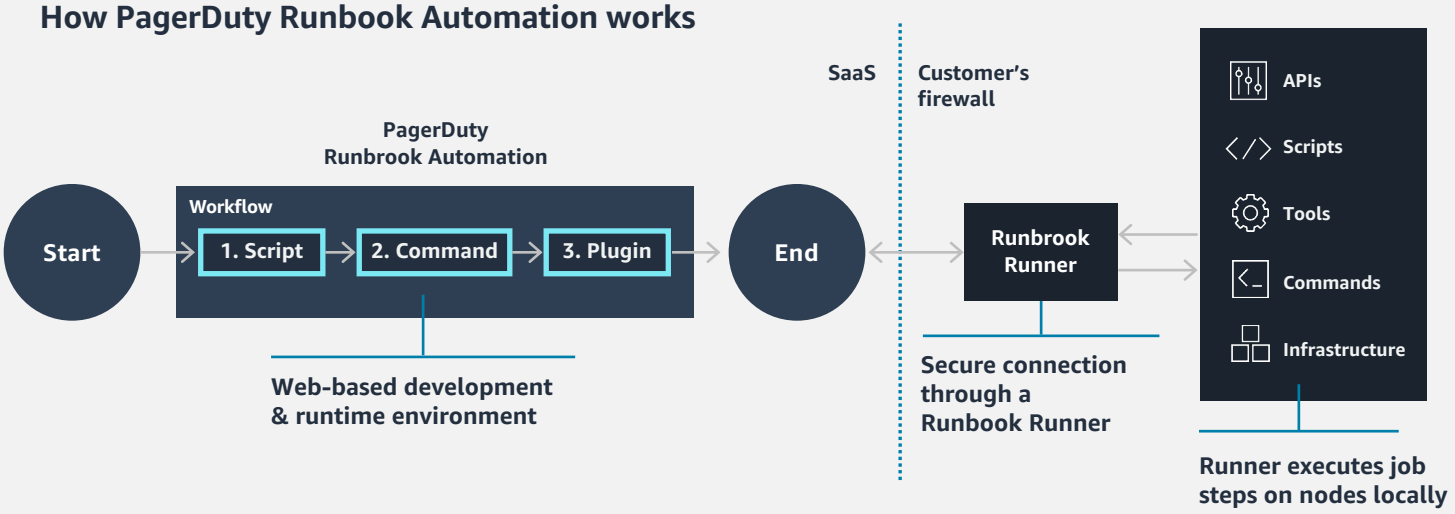
You can integrate with any ITSM or ticketing solution (JIRA, ServiceNow, BMC Helix, and others) to automatically create tickets from PagerDuty incidents and vice versa. Create conference bridges, add responders, track dependencies, run response plays, send status updates, sync status, and view audit history and incident details, and more—all while taking advantage of PagerDuty's response capabilities to modernize your ITIL-based major incident management process.



Integrated runbook automation

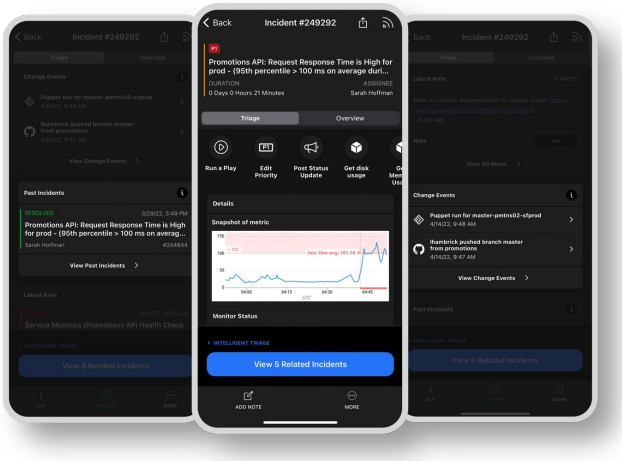
Reduce manual effort, escalations, and response times with PagerDuty Automation Actions. Provide incident responders with tailored diagnostic and remediation automation plays so they can resolve incidents safely and securely in PagerDuty with the click of a button.

How PagerDuty Runbook Automation works



Seamless mobile experience

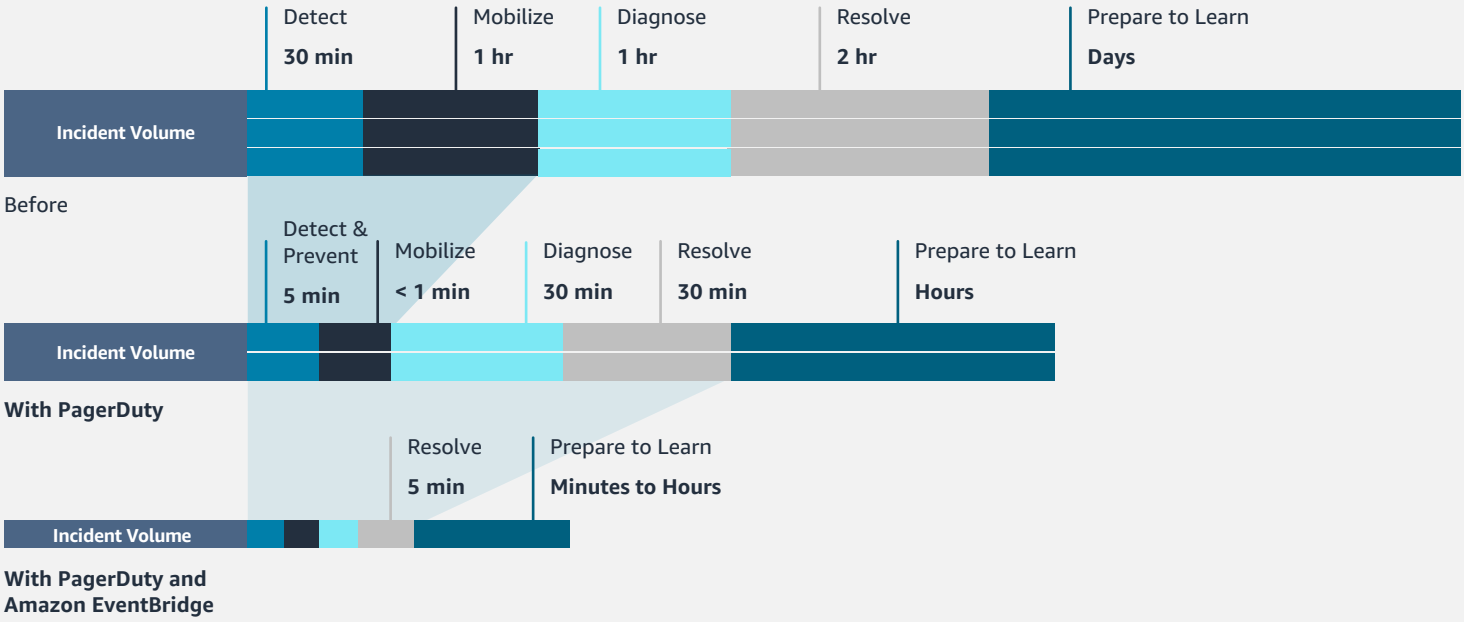
Handle incidents seamlessly from your mobile device. Get all the information you need at a glance with on-call shifts and impacted technical services front and center, with easy navigation to see change events, past incidents, and service dependencies on mobile. You can even run an Automated Action directly from the PagerDuty mobile app, available on iOS and Android.



Improved time to resolution

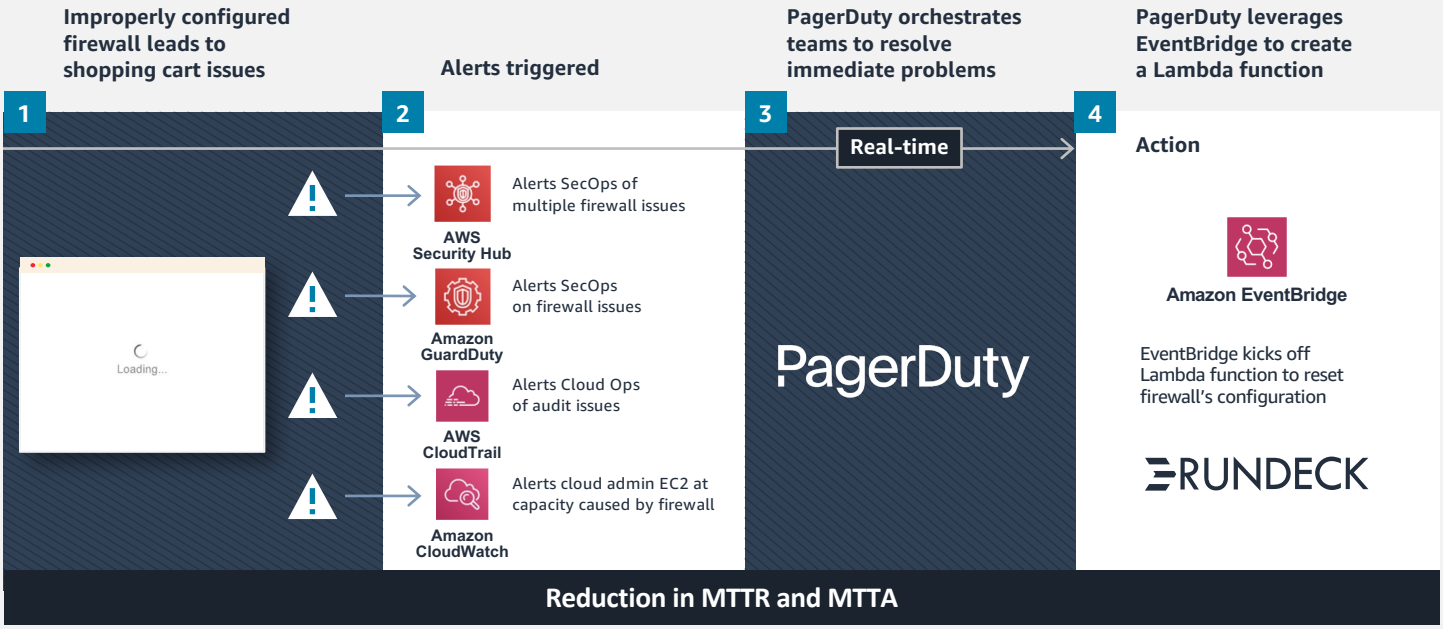
Manual incident response and resolution processes can take multiple hours to address issues, but with PagerDuty, you can achieve significant efficiencies in triaging issues and mobilize the right teams to work on remediation. The automation inherent in PagerDuty drives down resolution time and analytics features such as post-mortems help you understand where to invest in order to prevent future issues, or to drive down future resolution time. With PagerDuty and Amazon EventBridge together you gain the benefits of supercharging those actions.

PagerDuty and Amazon EventBridge work together to speed remediation



PagerDuty lets responders immediately invoke both diagnostic activities (disk space, memory usage) or remediation activities (scale up, restart the service) to deal with emergent behavior. Collecting basic diagnostic information saves time and makes triaging easier.

An example scenario where an improperly configured firewall leads to shopping cart issues



Use cases beyond auto-remediation

- **Self-service automation for cloud ops** allows developers to provision their own AWS environments, but in a standardized model so that dependencies can be preloaded and security settings can be applied before the environment is made available to developers. In addition to ensuring that requirements are met and costs are controlled, developers can start coding right away instead of wasting time performing repetitive tasks or waiting for the platform team do the same.
- **Compliant cloud ops is a necessity.** Every job execution is logged, which provides auditable history more directly aligned with business activity. This makes it easier and more optimal for cloud engineering teams to meet compliance requirements.
- **Multi-account automation.** Runbook Automation makes it possible to safely delegate access for the same automated tasks, so you can easily manage infrastructure in multiple AWS accounts. For example, allowing customer success engineers to easily support customer environments without needing detailed knowledge or direct access to infrastructure resources.

aws marketplace

Continue your journey with AWS Marketplace

Rafay, Logz.io, and PagerDuty can be used together along with AWS to establish a well-engineered approach to cloud-native operational resilience. Gaining the capabilities these tools offer can provide a strong foundation for continuing to advance through your cloud-native journey. You can explore these and other DevOps and cloud-native-focused tools in AWS Marketplace.

To get started, visit: https://aws.amazon.com/marketplace/solutions/devops

AWS Marketplace

Third-party research has found that customers using AWS Marketplace are experiencing an average time savings of 49 percent when needing to find, buy, and deploy a third-party solution. And some of the highest-rated benefits of using AWS Marketplace are identified as:

Time to value

Cloud readiness of the solution

Return on Investment

Part of the reason for this is that AWS Marketplace is supported by a team of solution architects, security experts, product specialists, and other experts to help you connect with the software and resources you need to succeed with your applications running on AWS.

Over 13,000 products from 3,000+ vendors:

APPDYNAMICS

bridgecrew BY PRISMA® CLOUD

Quali

circleci

HashiCorp

JFrog

Buy through AWS Billing using flexible purchasing options:

- Free trial
- Pay-as-you-go
- Hourly | Monthly | Annual | Multi-Year
- Bring your own license (BYOL)
- Seller private offers
- Channel Partner private offers

Deploy with multiple deployment options:

- AWS Control Tower
- AWS Service Catalog
- AWS CloudFormation (Infrastructure as Code)
- Software as a Service (SaaS)
- Amazon Machine Image (AMI)
- Amazon Elastic Container Service (ECS)
- Amazon Elastic Kubernetes Service (EKS)

aws marketplace

Get started today

Visit aws.amazon.com/marketplace to find, try and buy software with flexible pricing and multiple deployment options to support your use case.

https://aws.amazon.com/marketplace/solutions/devops

Authors:

James Bland
Global Tech Lead for DevOps, AWS

Aditya Muppavarapu
Global Segment Leader for DevOps, AWS