harness

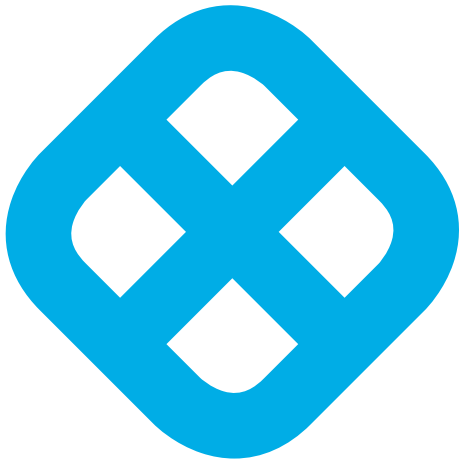
Continuous Delivery Insights 2020

About Harness

Based in San Francisco, Harness is the industry's first Continuous Delivery-as-a-Service™ platform designed to provide 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.

Sign-up for your free 14-day trial at **Harness.io**

Table of Contents



Executive Summary

Over the past year, Harness interviewed over 100 corporations across multiple industries with regards to their current Continuous Delivery process, org structure, tooling and KPIs.

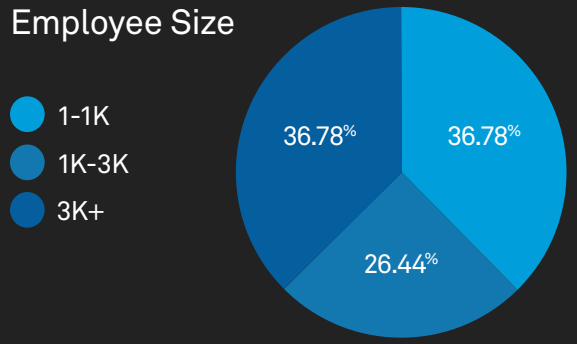
This Continuous Delivery Insights Report will highlight the common challenges faced by corporations adopting continuous delivery throughout the past 12 months.

In this report, we identified the time, effort, cost, and velocity associated with their current Continuous Delivery process.

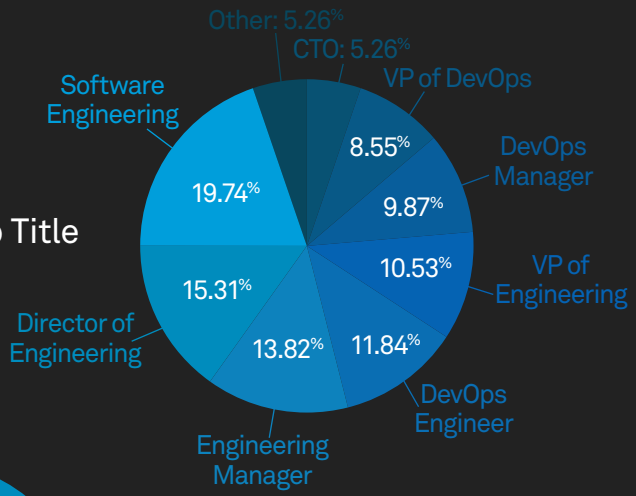
We hope developers and DevOps teams can use these findings to benchmark their own CD process against the organizations we observed.

Demographics

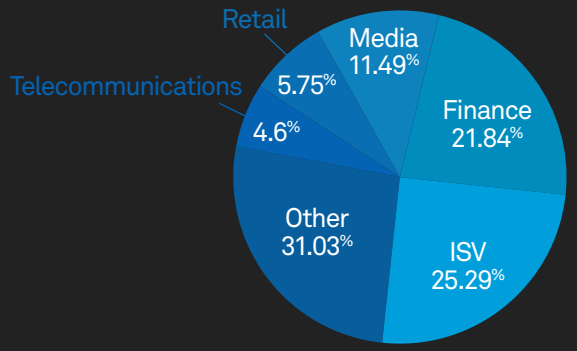
Employee Size



Job Title



Industry



Data Source

The data collected came from interviews during our sales process. These interviews are referred to as Continuous Delivery Capability Assessments (CDCA). The questions asked during the interviews help us create business cases and map deployment maturity. The interview reveals time frames and employee resource metrics, as well as the tools used in the interviewees pipelines. We removed company affiliation from the data, so no data points could be tied back to a specific company.

Analysis Methodology

After the data was anonymized, the data was separated by deployment stage. Tests for variance were used to identify outliers which were then removed by calculating 1st and 3rd quartile limits and then subtracting or adding the interquartile range respectively, normalizing the data.

Key Findings

We observed the following Continuous Delivery performance metrics across the sample. Based on this, the average and median org we observed is considered medium performing.

The customers we interviewed struggled the most with deployment frequency, lead time, and MTTR. Continue reading to further understand the Continuous Delivery challenges faced by these customers.

Deployment frequency represents the number of times a build is deployed to production.

Lead time represents the time it takes a new build to reach production and be verified.

Change Failure Rate represents the percentage of deployments that fail.

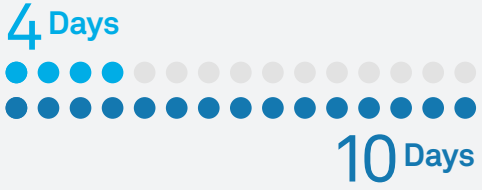
MTTR represents the time it takes to rollback or restore a service following a failed deployment.

Production deployment effort represents the total time spent by every person working to deploy a new artifact into production.

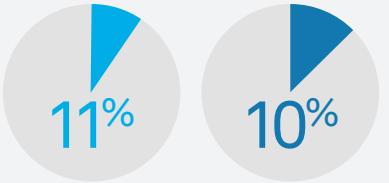
Production deployment cost represents the cost of total time spent by every person working to deploy a new artifact into production based on $56.81 per hour (avg. cost of $100,000 per FTE/year).

Annual Cost of Deployment represents the total number of production deployments per year for each customer divided by the total effort spent throughout their deployment pipeline to get a new artifact successfully into production.

Deployment Frequency



Change Failure Rate



Lead Time

8 Hours | 5.5 Hours

MTTR

60 Minutes | 30 Minutes

Production Deployment Effort

25 Hours | 10 Hours

Production Deployment Cost

$1450 | $535

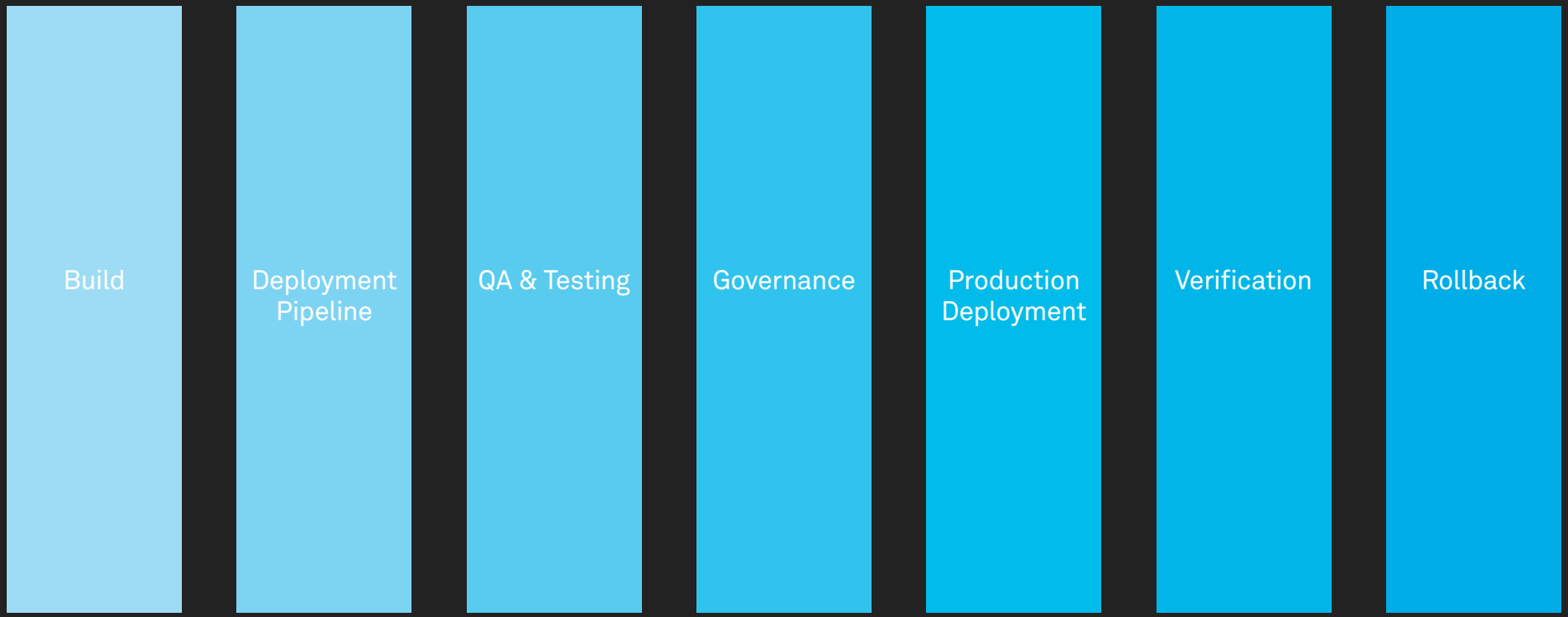
Annual Cost of Deployment

$109K Per App

What is Continuous Delivery?

Continuous delivery enables software changes of all types to reach production environments in a safe, quick, and sustainable way. The goal is to make deployments, in whatever architecture, predictable and routine such that developers can perform deployments on demand.

Here are the phases of a CD process:



Build

The build process takes you from source code to a packaged deployable artifact Think of this as Continuous Integration.

Artifacts are stored and managed in Artifact Repositories.

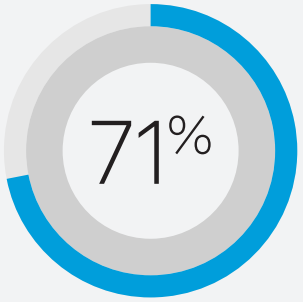
Key Observations:

- Nearly all customers have a mature build and Continuous Integration process that takes minutes to run and complete.
- SonaType Nexus and JFrog Artifactory are the most common repositories to manage artifacts.

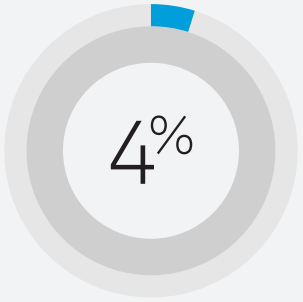
Key Challenges:

- Minimal reported challenges.

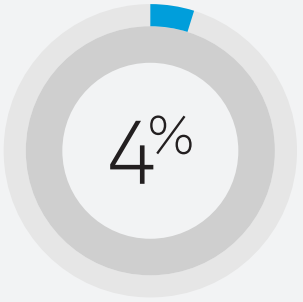
Common Build Tools:



Jenkins

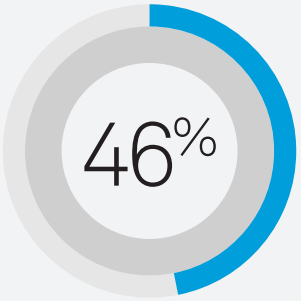


Bamboo

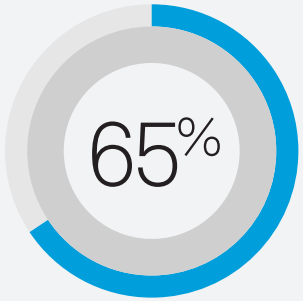


Circle CI

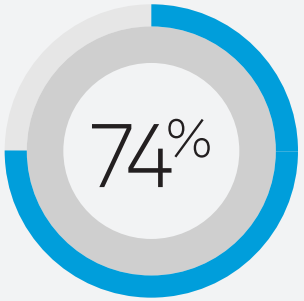
Common Artifact Repositories:



SonaType Nexus

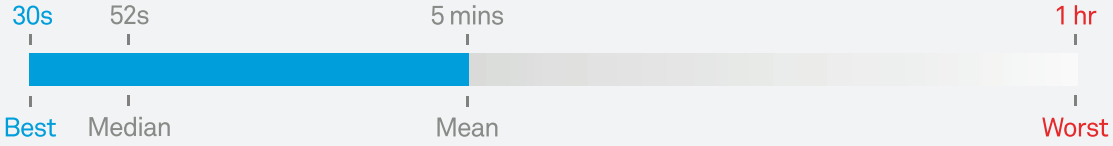


JFrog Artifactory



Native Cloud Container Registry

Exerted Effort Per Build:



Note: Customers may use one or more tools for this process

Deployment Pipeline

A deployment pipeline takes an artifact and is responsible for promoting and verifying it across environments (e.g. Dev, QA, Staging, Production).

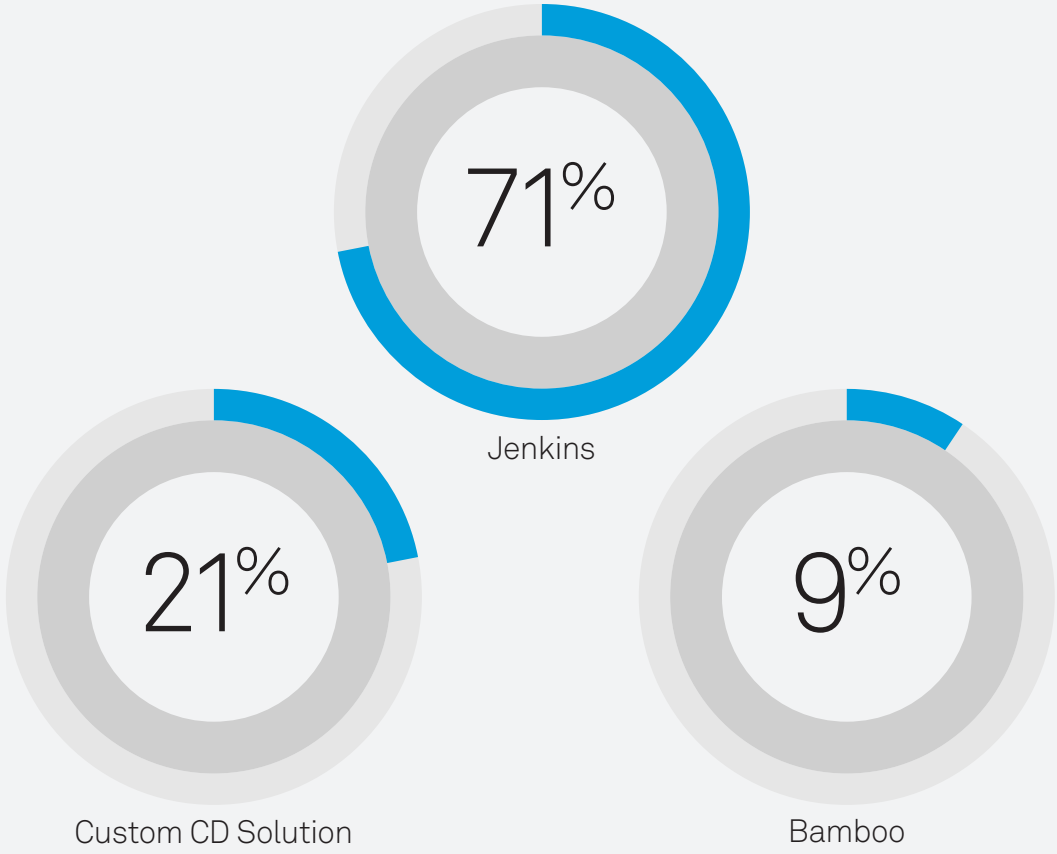
Key Observations:

- Pipelines are typically a set of custom bash or powershell scripts being orchestrated by a build or CI tool.
- Average of 66 work hours needed to create a new pipeline.
- Average of 3 FTEs managing deployment pipelines. Median is 2 FTEs. Maximum was a financial services corporation that had 55 FTEs managing deployment pipelines.

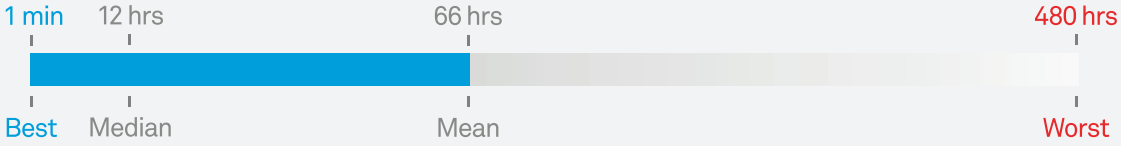
Key Challenges:

- Maintaining scripts (TOIL).
- Difficult to debug/troubleshoot failure.
- Lack of templatization and reuse.
- Managing microservice dependencies.

Common Tools & Solutions:



Exerted Effort to Onboard a New Application (Build a New Pipeline):



Note: Customers may use one or more tools for this process

QA and Testing

The QA and Testing process is responsible for finding defects and regressions in artifacts before they are deployed to customers in production.

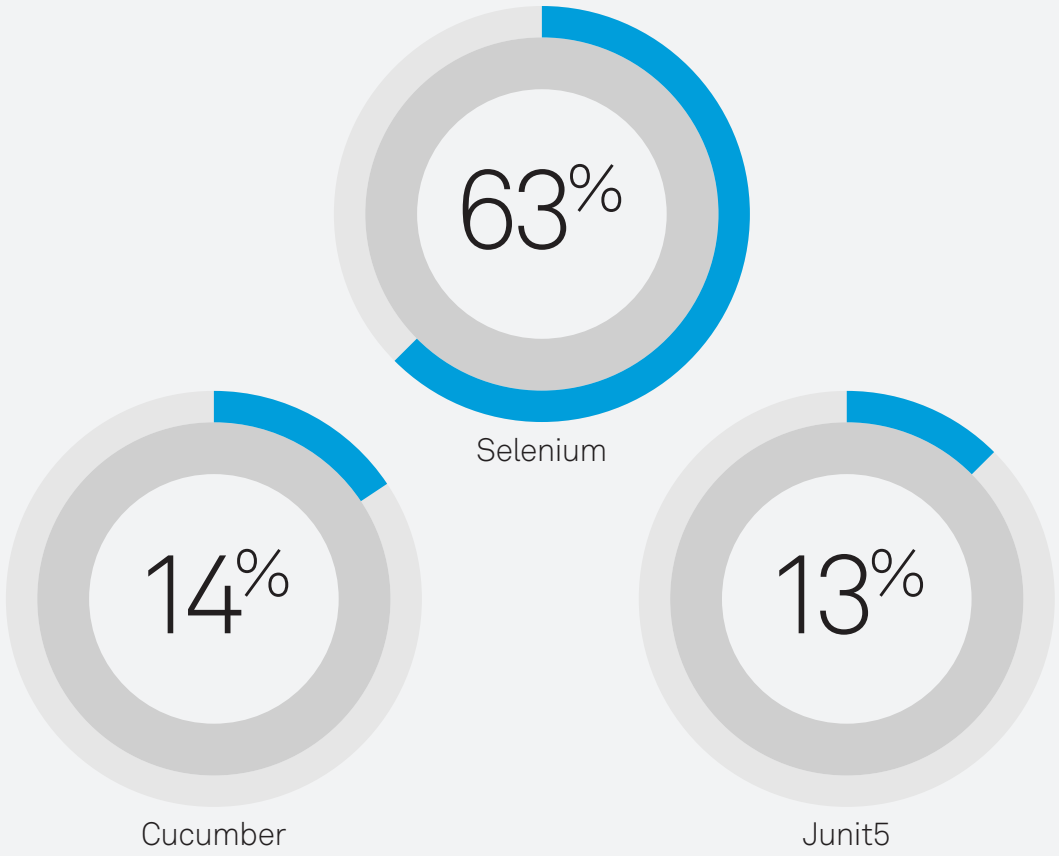
Key Observations:

- Average of 16 work hours for an artifact to move through the QA andTesting phase.
- Despite test automation tools existing most QA remains manual.

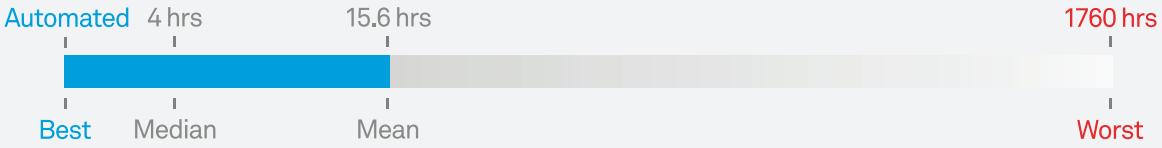
Key Challenges:

- Acceptable test coverage.
- Levels of Automation.
- Creating and maintaining test scripts for highly dynamic cloud-native services.

Common Tools:



Exerted Effort to QA and Test:



Note: Customers may use one or more tools for this process

Governance

Governance is the process of controlling how an artifact is promoted across environments to ensure compliance and security measures are met.

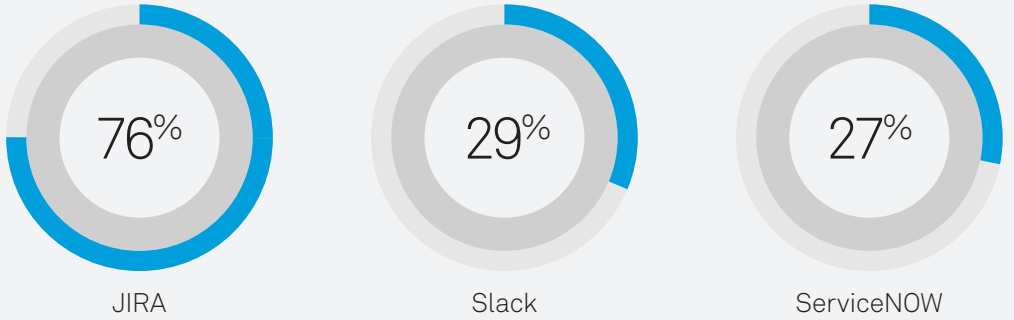
Key Observations:

- On average 3.4 hours is spent in a deployment pipeline waiting for approval.
- The worst deployment pipeline went through a 10-day manual approval process.
- Very few customers are practicing Continuous Deployment (no approvals).
- Atlassian JIRA is typically used for non-prod and ServiceNow for prod envs.

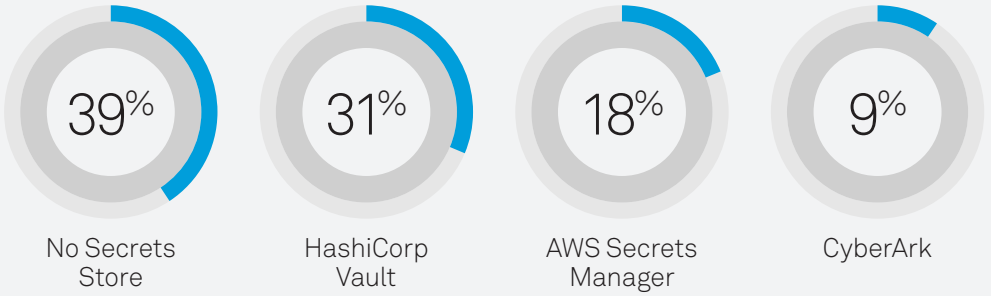
Key Challenges:

- Manual ticketing & approval processes.
- Lack of centralized secret store.
- Lack of role based access control.
- Lack of audit trails.

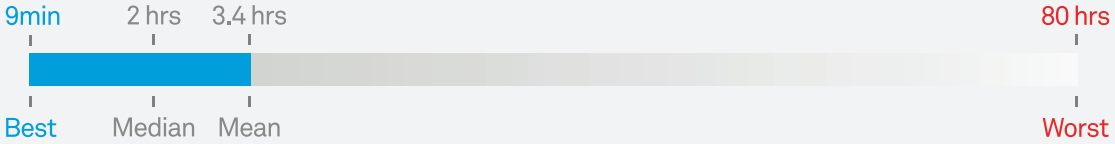
Change Management:



Secrets Management:



Lead Time for Change Approval



Note: Customers may use one or more tools for this process

Production Deployment

Production deployment is the promotion of an artifact to customers or end users in production.

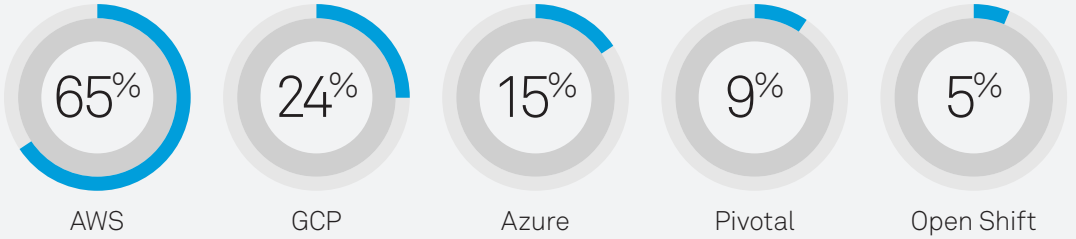
Key Observations:

- Majority of customers still using rolling deployments in production.
- On average it takes 2.2 working hours to deploy a new artifact into production.

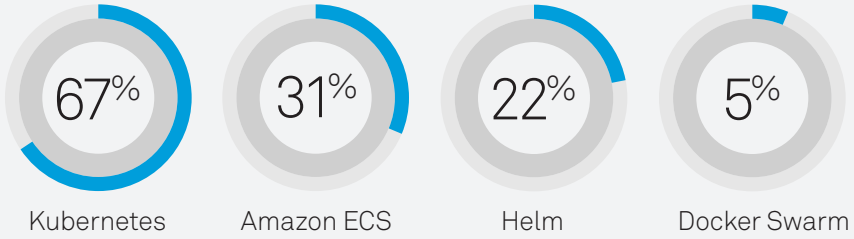
Key Challenges:

- Manual orchestration of scripts.
- Lack of blue/green/canary capabilities.

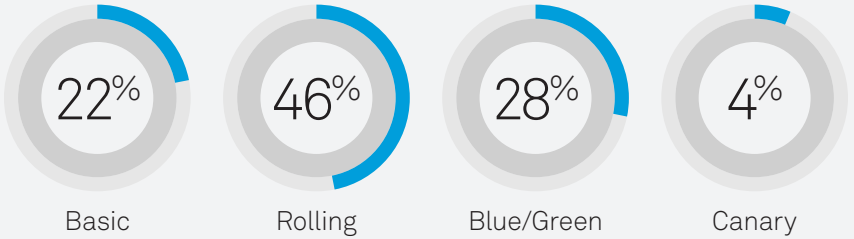
Cloud Providers:



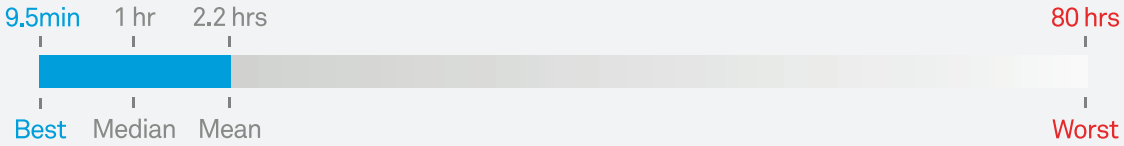
Container Orchestration:



Release Strategies:



Exerted Effort to Deploy:



Note: Customers may use one or more tools for this process

Verification

Verification is the process of ensuring the performance, quality and health of an artifact post-deployment.

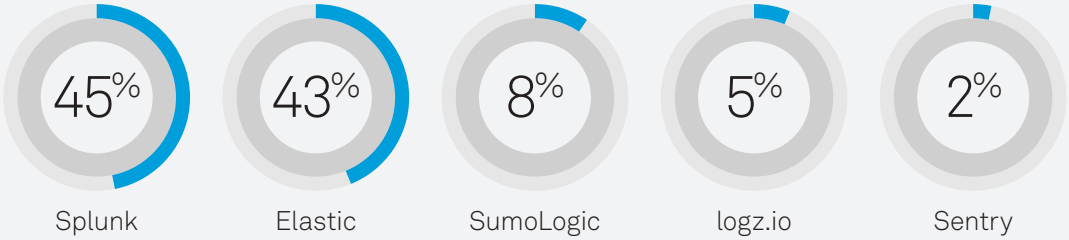
Key Observations:

- Deployment Verification remains a manual task.
- On average it takes 2.2 working hours with 1-2 people engaged in this activity.
- Monitoring tools and data are not integrated into the deployment pipeline.

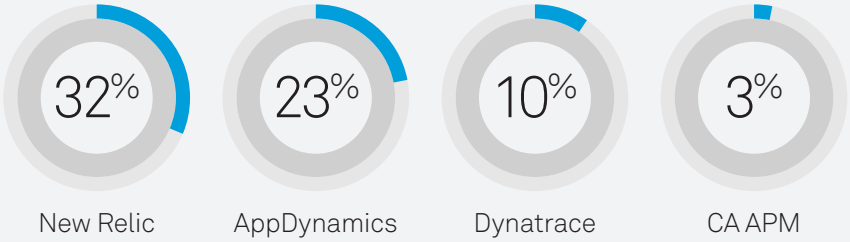
Key Challenges:

- Automating the validation of canary phases/gates and deployments.
- Detecting new errors and exceptions specific to a deployment.
- Setting thresholds of what normal and abnormal looks like in a dynamic microservice architecture.

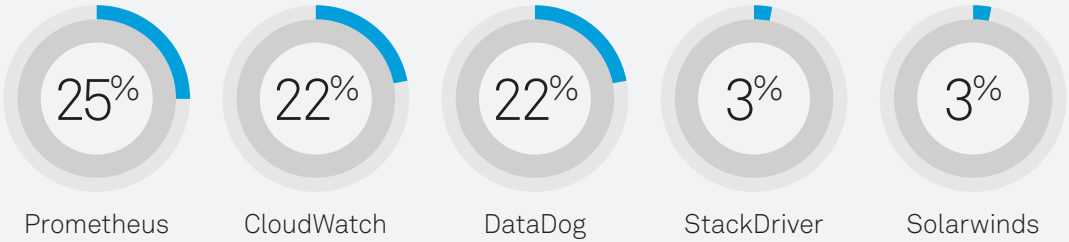
Log Analytics Tools:



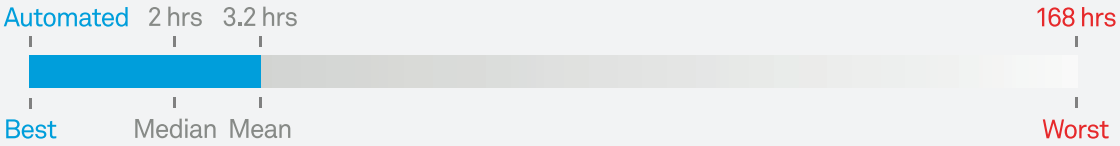
Application Performance Management Tools:



Cloud / Infrastructure Monitoring Tools:



Exerted Effort per Deployment Verification:



Note: Customers may use one or more tools for this process

Rollback

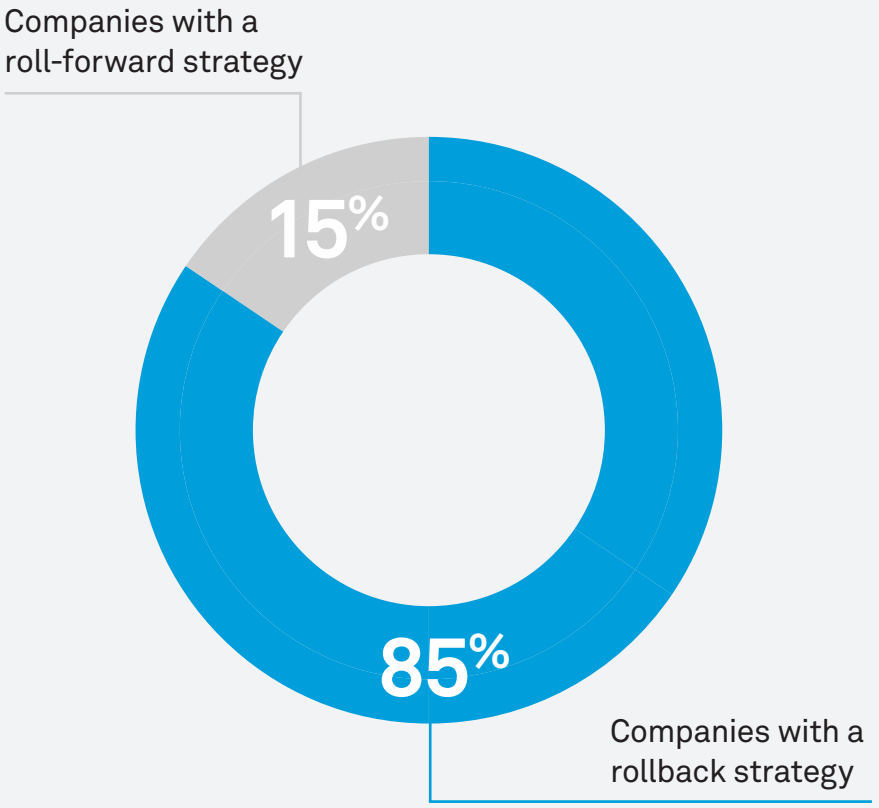
Rollback is the process of deploying the last working version of an artifact in production after experiencing a deployment failure.

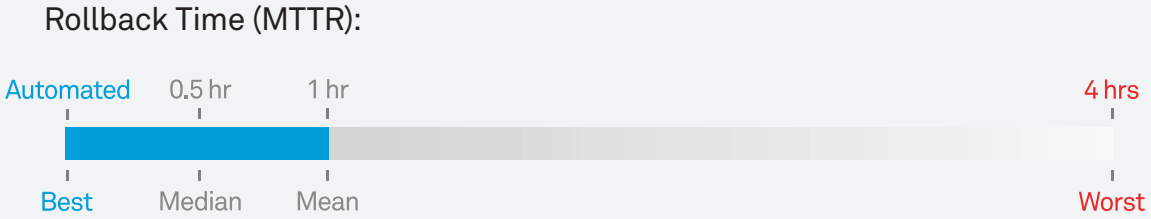
Key Observations:

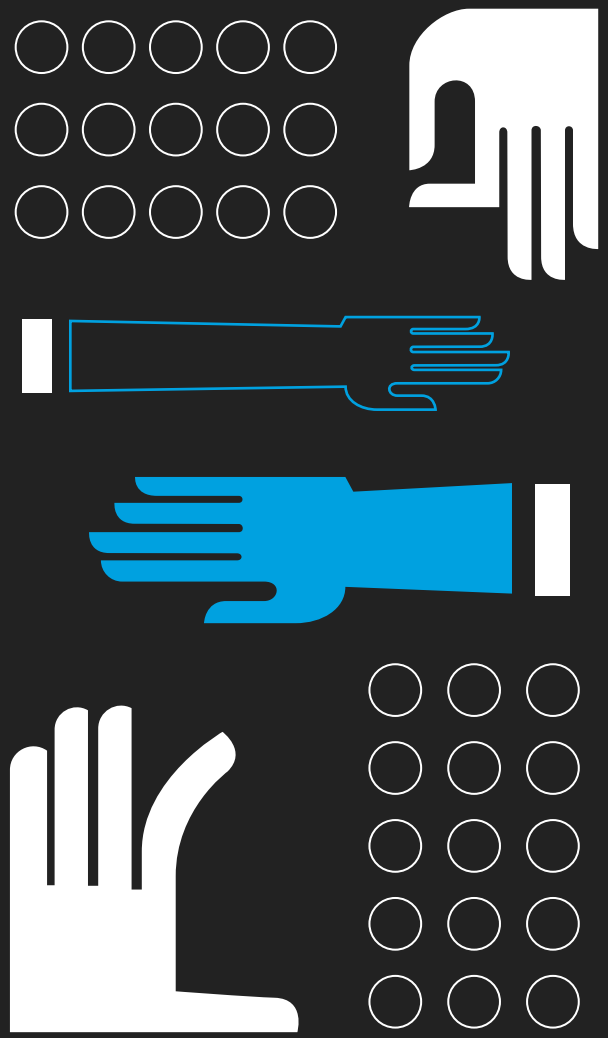
- 85% of companies have a rollback strategy vs 15% of companies that choose to “roll-forward.”
- 11% average change failure rate was observed.
- Average time to rollback (MTTR) was 60 minutes.

Key Challenges:

- Manual rollback using (multiple) scripts.
- Managing state in databases & data stores.
- Local vs. global rollback of microservices dependencies.







Final Thoughts

The average corporation in this report released to production every 10 days. A decade ago this might have been considered agile, today this can be classified as medium to low performing.

In 2020, Elite and High performing teams are deploying on-demand (or daily at the very minimum).

The average corporation exhibited a mature Continuous Integration (CI) process that enabled code to artifact in minutes using tools like Jenkins. The opposite is true with respect to Continuous Delivery (CD) and pushing a new artifact to production. Observed deployment pipelines were expensive to create and maintain, requiring 2-3 dedicated FTEs, and up to 66 hours to onboard a new application. New artifacts on average took 5.5 hours to reach production, and exerted an average effort of 10 hours per deployment. This is why the average cost of deployment per application is costing corporations approximately $109,000 per year.

To increase deployment frequency teams must move to a self-service Continuous Delivery model where developers deploy on-demand, and pipelines are streamlined so new artifacts can be pushed to production in under an hour.

To achieve this deployment pipelines need to become smarter and less reliant on people or scripts. In addition, a modern pipeline needs to orchestrate and automate QA tests, governance, deployment, verification and rollback. Without a truly ‘hands off’ approach Continuous Delivery will remain a pipe dream for most corporations.

Appendix

Acknowledgments

Researched, Analyzed, and written by Dan Lamm

Edited by Steve Burton, Ravi Lachhman

Designed by Aoife Gibbs