Continuous Delivery

harness

Who are these people yapping?

Drew – Tech Guru and

Lifestyle Coach

Ravi – Rick & Morty SME

and Astrologist to the Stars

What are we talking about?

- Continuous Integration vs Continuous Delivery
- Don’t let taxonomy get in the way
- Automate your yes [and no]
- Deploying Workloads on K8s – Best Practices
- Setup a simple Harness deployment on K8s

What is CI or CD?

Back in my day… 2008

JMS

IBM WebSphere

ORACLE WEBLOGIC

JBoss by Red Hat

Still my day… 2020

Continuous Integration

```
@Override
public void apply(Project project) {
    PluginContainer plugins = project.getPlugins();
    plugins.apply(DeployedPlugin.class);
    plugins.apply(JavaLibraryPlugin.class);
    plugins.apply(ConventionsPlugin.class);
    plugins.apply(InternalDependencyManagementPlugin.class);
    StarterMetadata starterMetadata = project.getTasks().create("starterMetadata", StarterMetadata.class);
    ConfigurationContainer configurations = project.getConfigurations();
    Configuration runtimeClasspath = configurations.getByName(JavaPlugin.RUNTIME_CLASSPATH_CONFIGURATION_NAME);
    starterMetadata.setDependencies(runtimeClasspath);
    File destination = new File(project.getBuildDir(), "starter-metadata.properties");
```

Identify the CD

Or

Or

Correct! Doughnuts = Awesome

Doughnut Deployment Pipeline

Idea

New Treat

Build

Gather Raw Material

Prepare New Doughnut Flavor

Days

Manual

Test

Ask co-workers about taste

Blast on Social Media

Hours

Subjective

Deploy

Mins

Yum

You

Verify

Eat / Instagram Doughnut

Mins

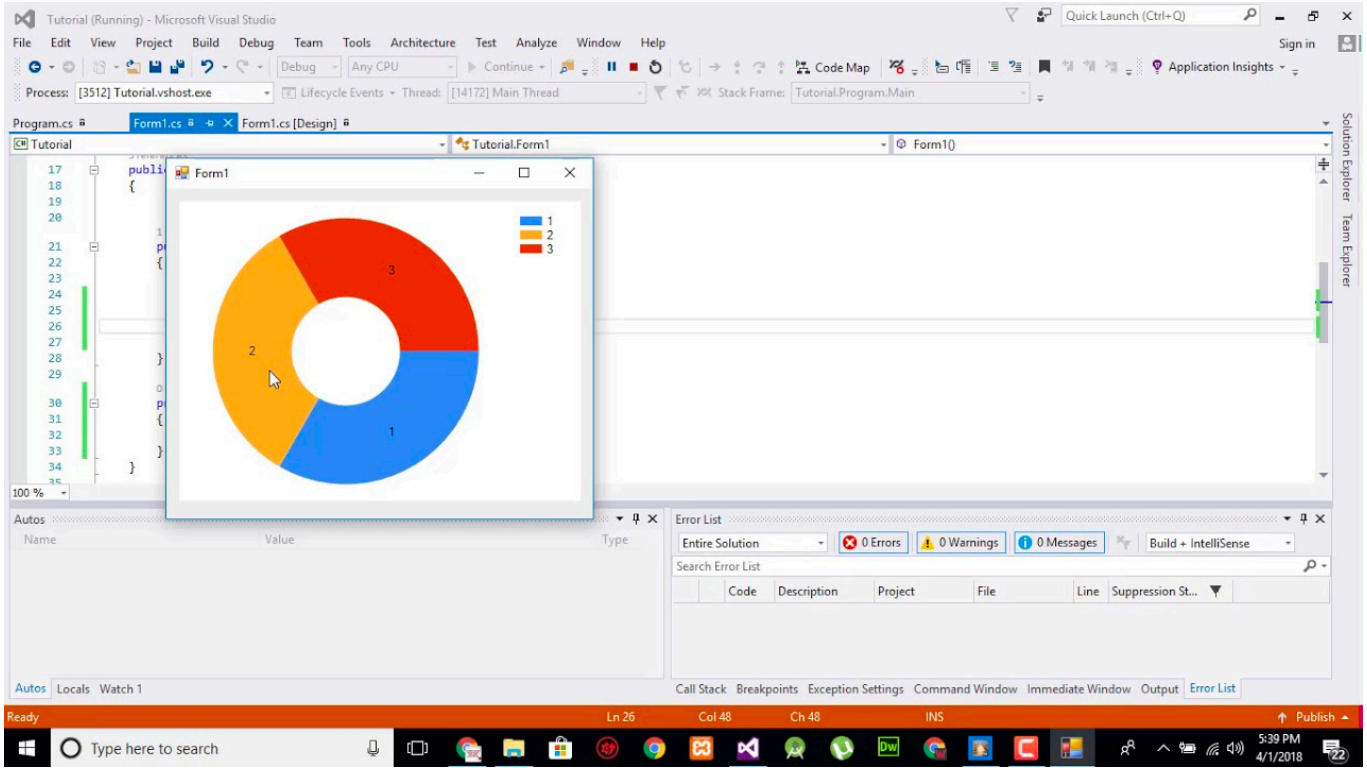
Look Here

Rollback

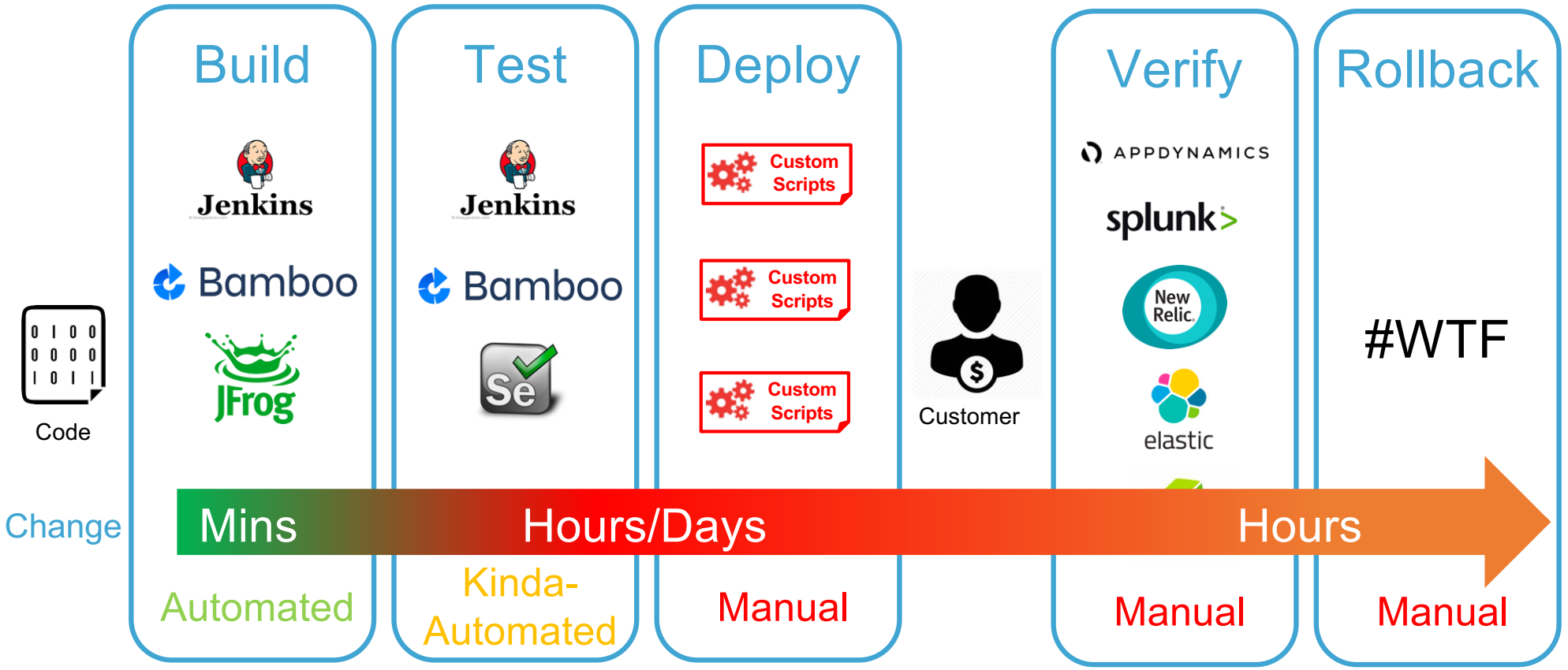
Yelp / Google Map Krispy Kreme

Ouch

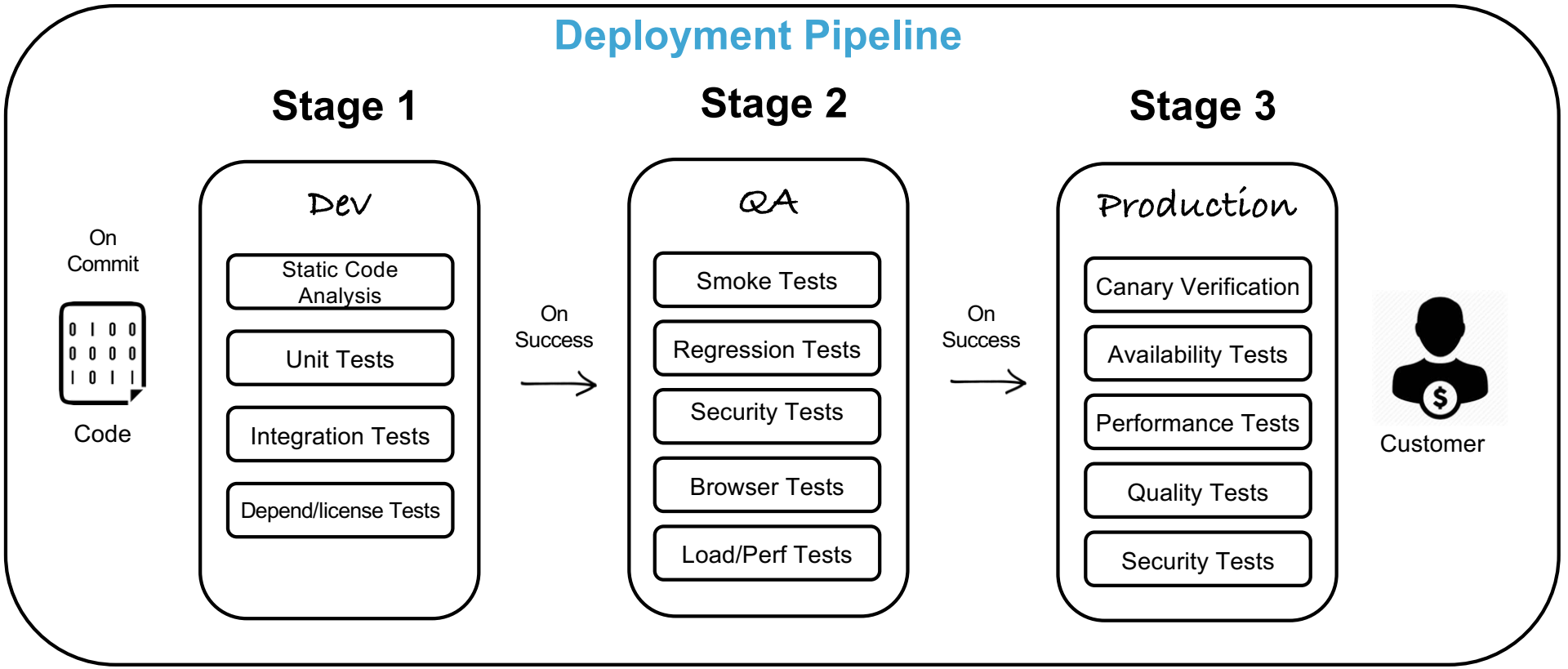
One Problem


Tutorial (Running) - Microsoft Visual Studio
Form1
1
2
3
Autos
Error List
Ready
Type here to search

Software Deployment Pipeline

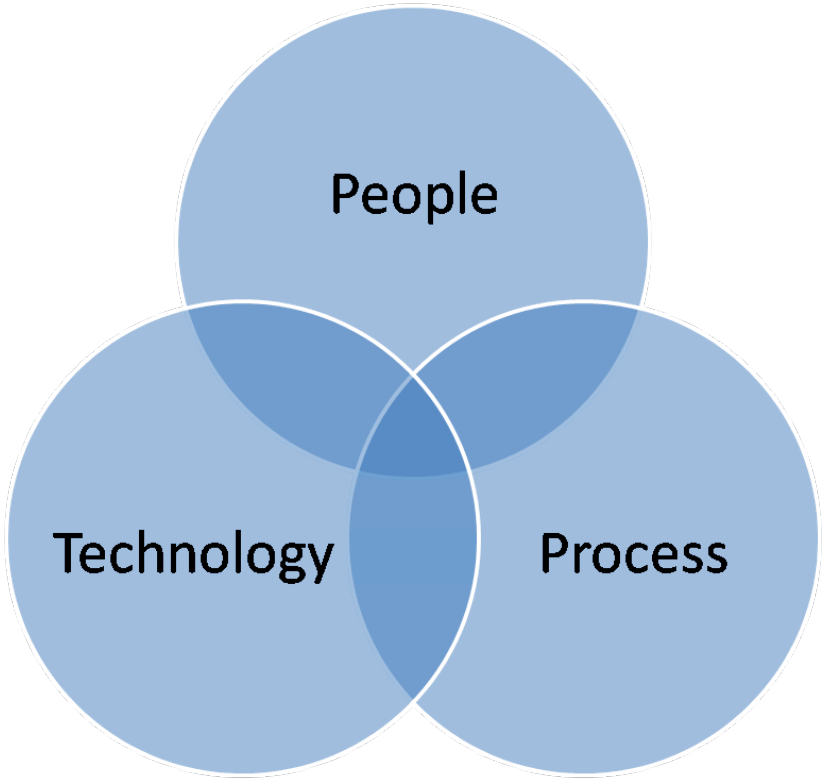


Continuous Delivery = Automate Everything

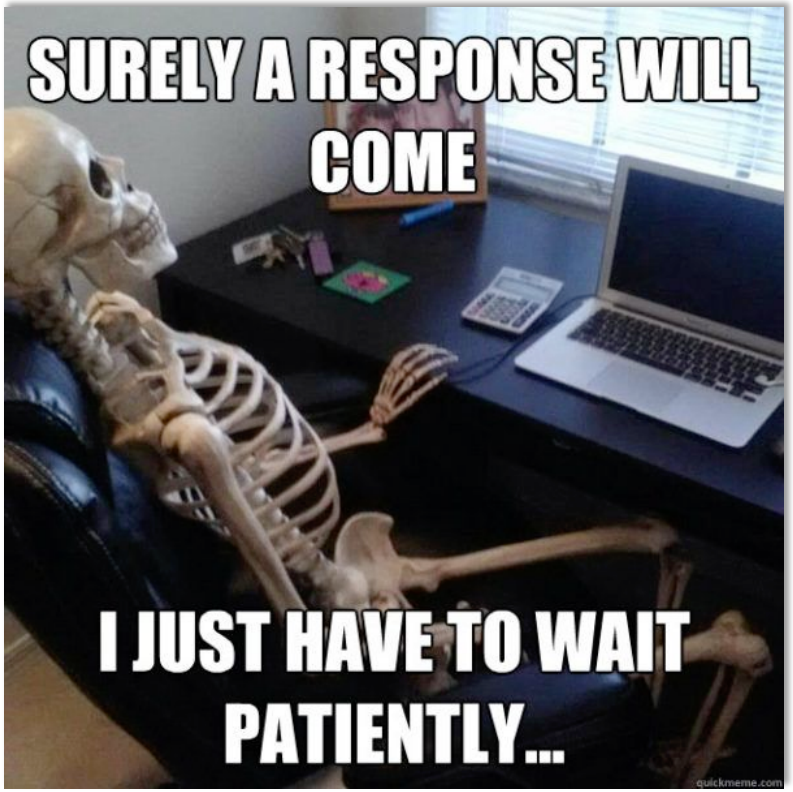


Are you confident?

Confidence Trifecta



Apply for a credit card in 1990

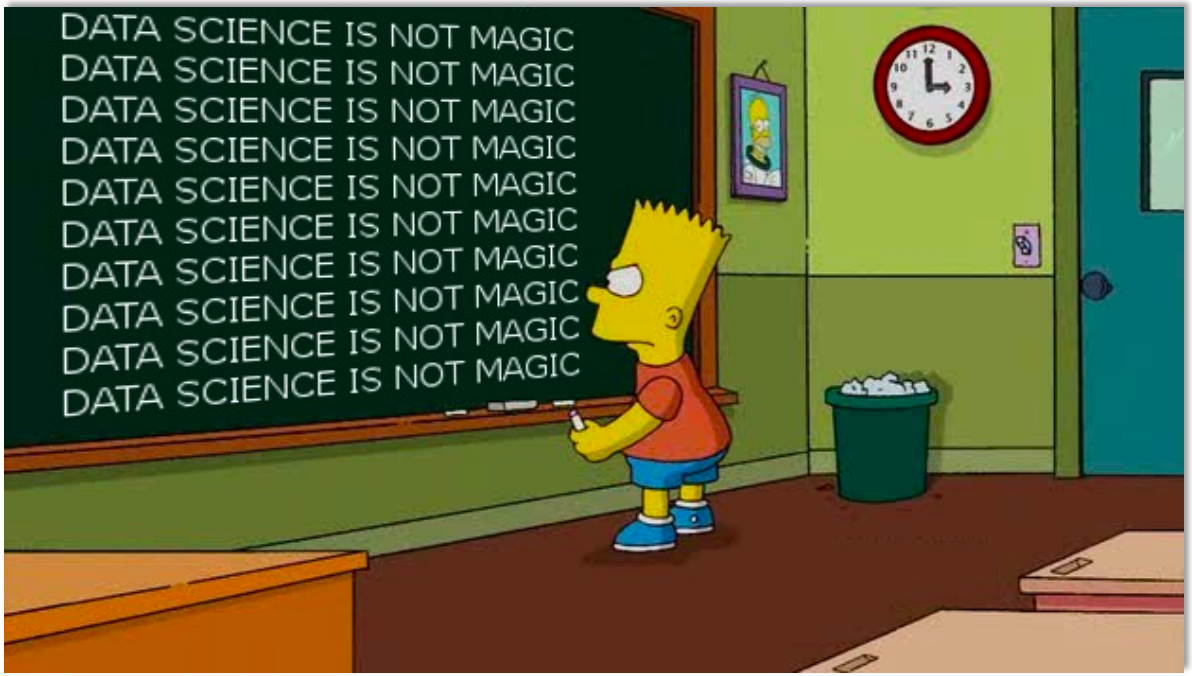


Apply for credit card 2020

the POINTS GUY

Level of automation up to you

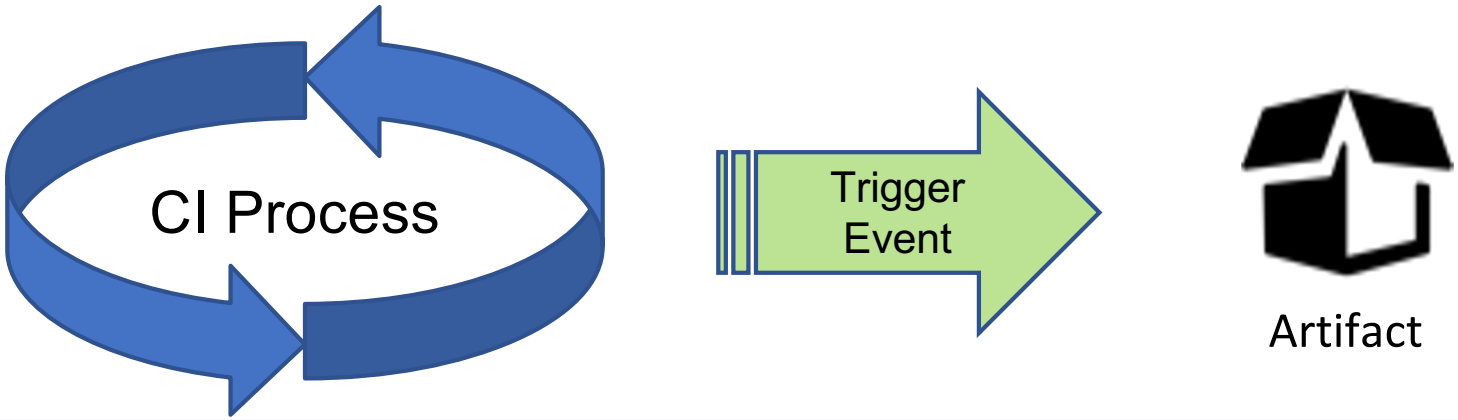
Machines should help



Deployment Best Practices for Kubernetes

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



What Triggers That Trigger?

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

Janette McGamexec
CTO, Second To Last Games, Inc.

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 . . .

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

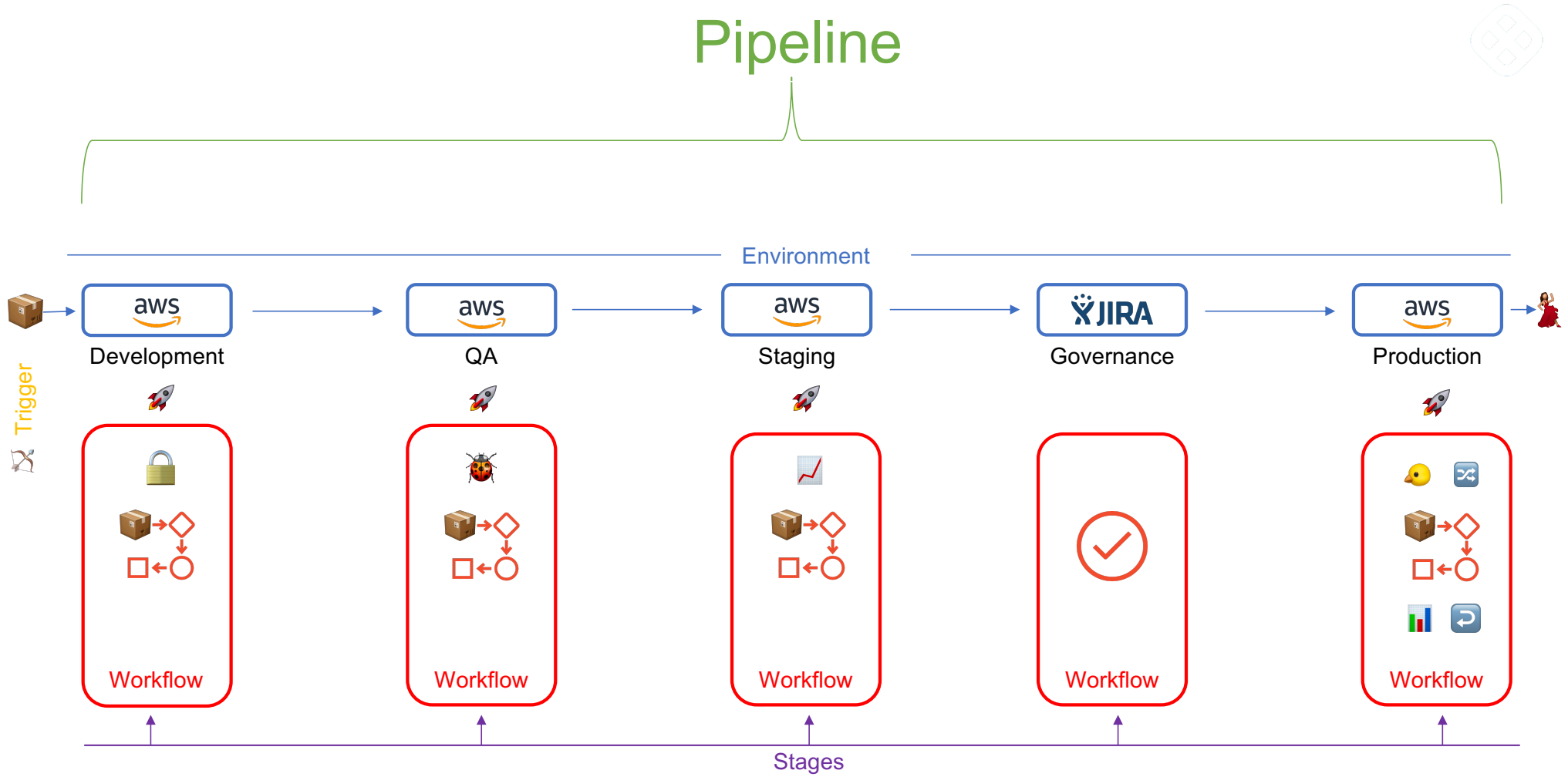
Harness.io SaaS Is a Product of Continuous Deployment

- 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

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.
- See next slide for an example

Pipeline



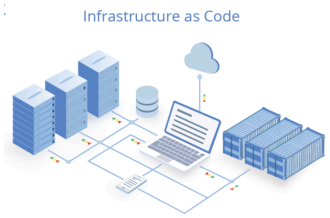
Building Blocks of a CD Pipeline

Typical deployments follow these steps:

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

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



Terraform

CloudFormation

PaaS
Platform as a Service

CF
Pivotal CF

CLOUD
COMPUTING

aws

Google Cloud

Release Strategy

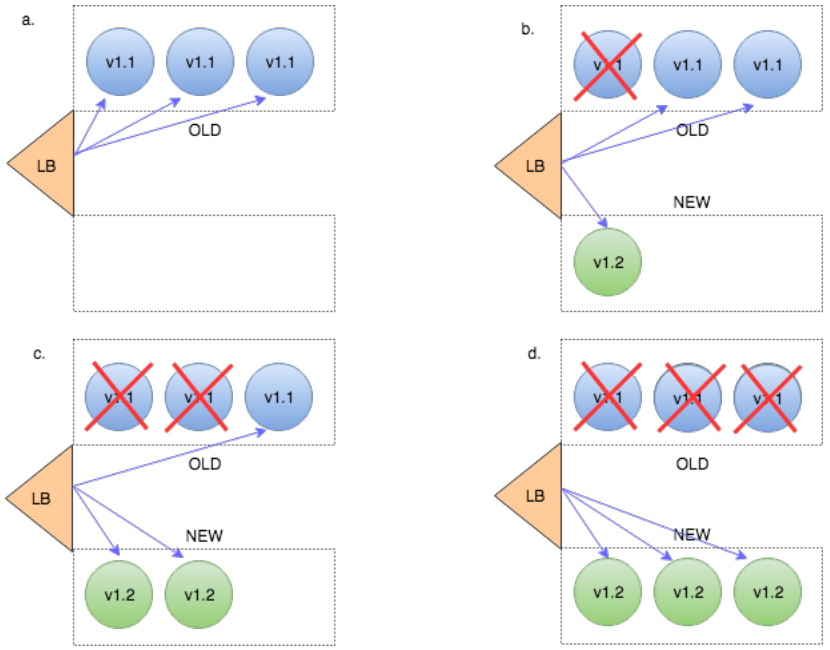
- Who approves your deployment? Sec-Ops? Product Management? Execs? All of the above?
- What tools do they use for this approval?

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

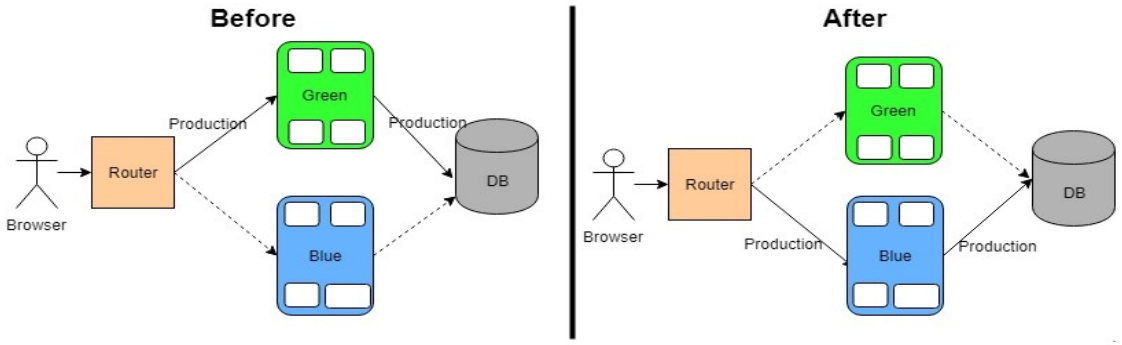
Types of Deployments – Rolling Upgrade

- **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



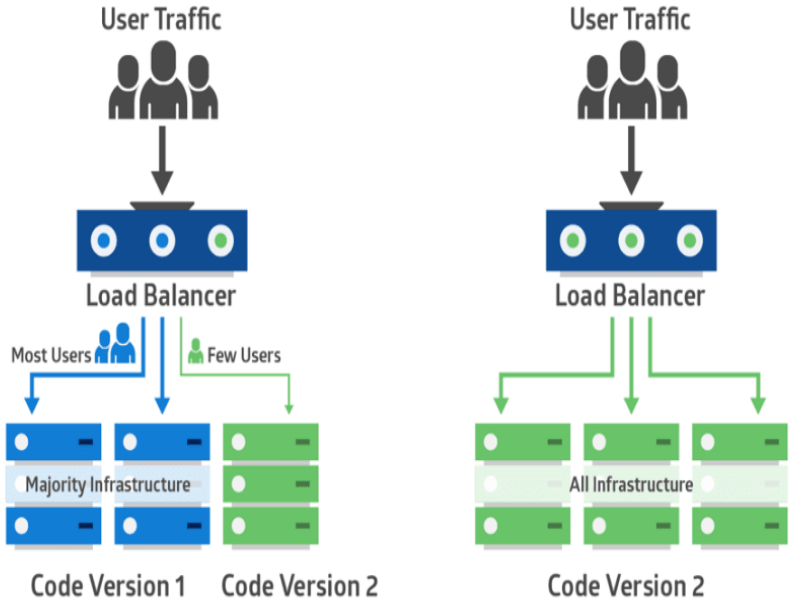
Types of Deployments – Blue / Green

- **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

Modern Deployments Are Complex

- So many different systems, different logos, different logs, metrics, reports, and verifications
- Environment dependent variables, certificates, localizations, and other settings
- Factor in security and compliance requirements on top of this . . .

Modern Deployment Solutions

```
#!/bin/sh
#!/bin/bash
#!/usr/bin/env python3
```

- Scripted Solutions
- Most common CD "solution"

- Open source
- Originally developed by Netflix

- SaaS and on prem
- From the founders of AppDynamics

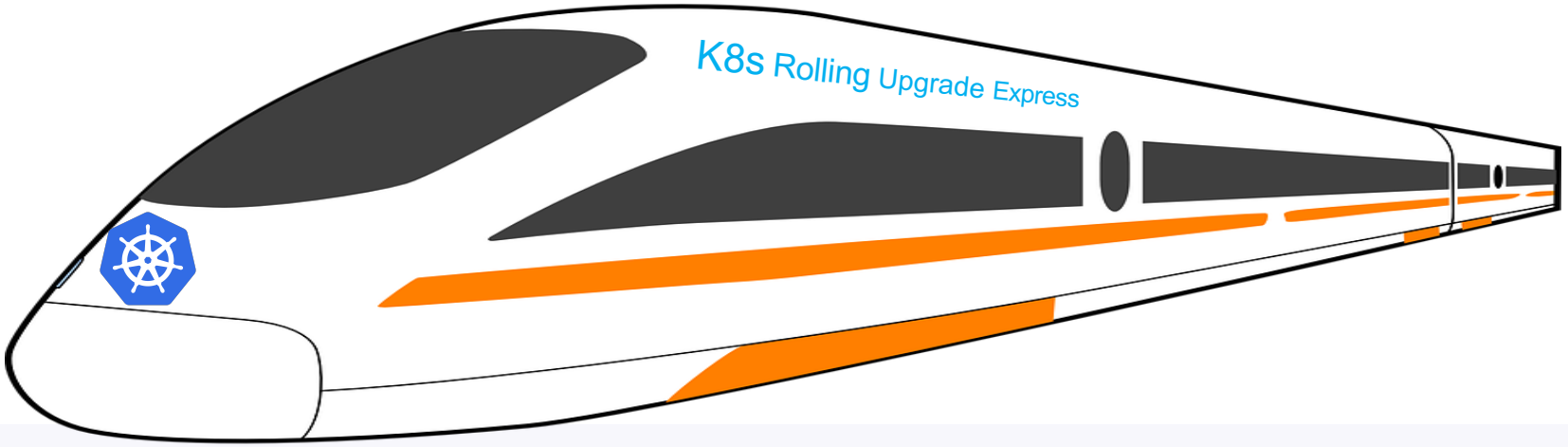
Labeling and Taxonomy Hugely Important to K8s!!

- Before you can dream of continuous delivery with K8s you need to make sure your K8s is up to the task
- Build a clearly defined namespace infrastructure
- Establish clear tagging taxonomies for **all** your workloads
- Namespaces and labels control everything in K8s: RBAC, security, networking, resource management, networking, and even deployments!!!

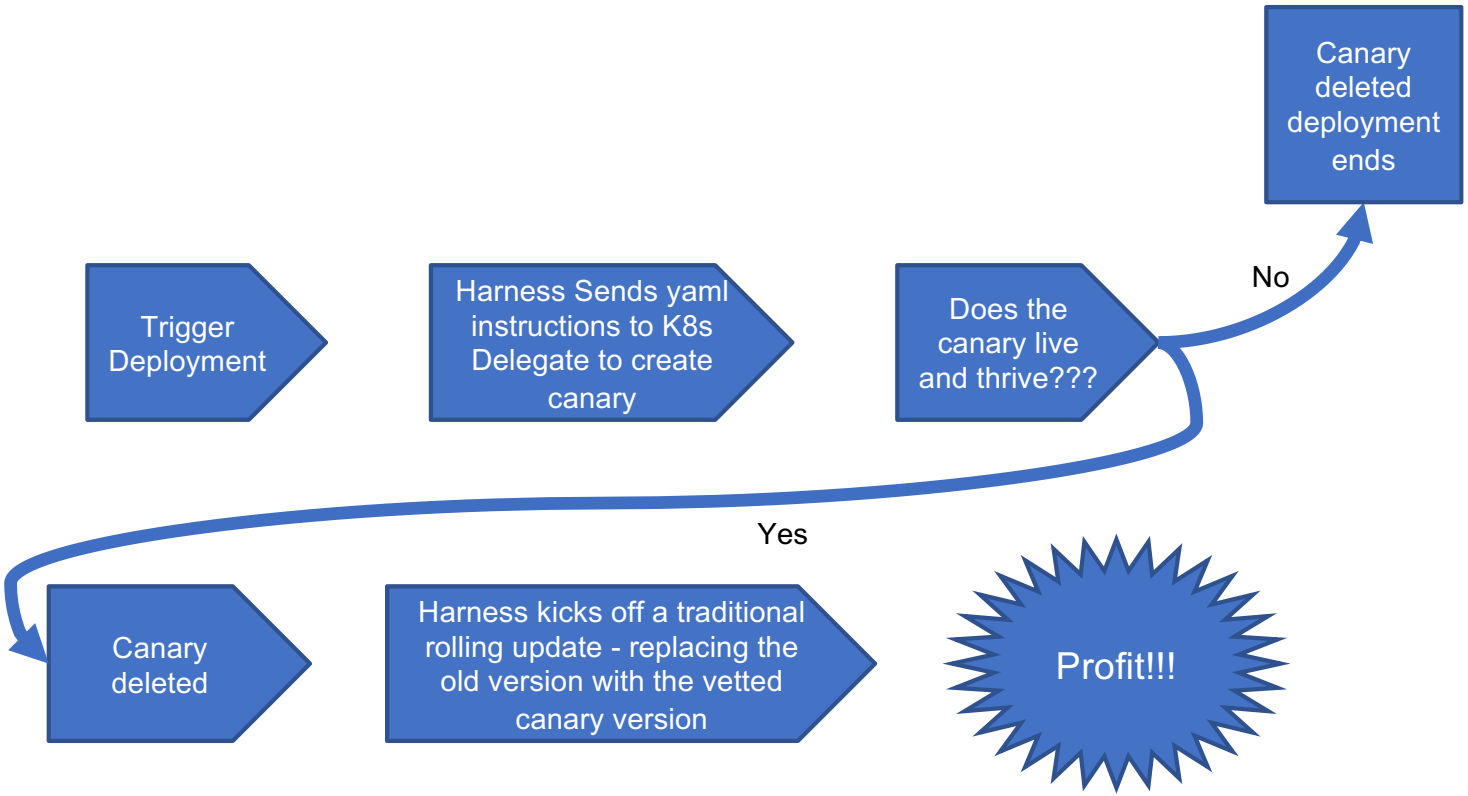
```
namespace = backend
app = backend_proxy
version = 1.7.5.678
team = backend_network
zone = prod
region = EMEA
date_deployed = 1-25-20
dependencies = none
```

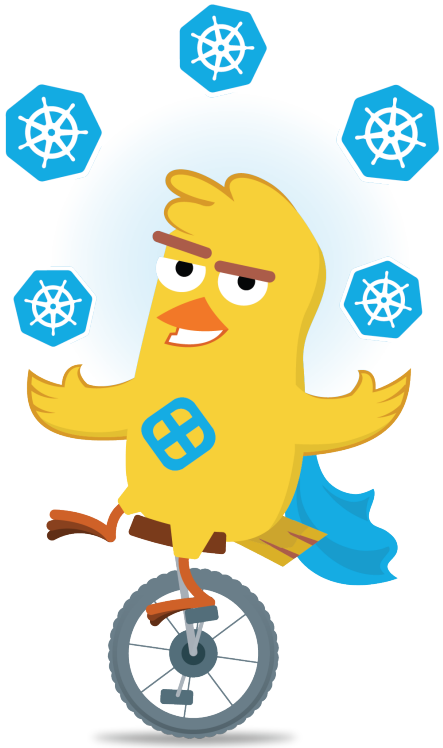
Make Sure Your Deployments Work With K8s' Strengths

- Kubernetes does rolling upgrades really well – like really really well
- It's a rolling upgrade machine – once you kick it off, it's relentless!
- But you better *really* be ready to upgrade once that train leaves the station!



Kubernetes Canary Workflow

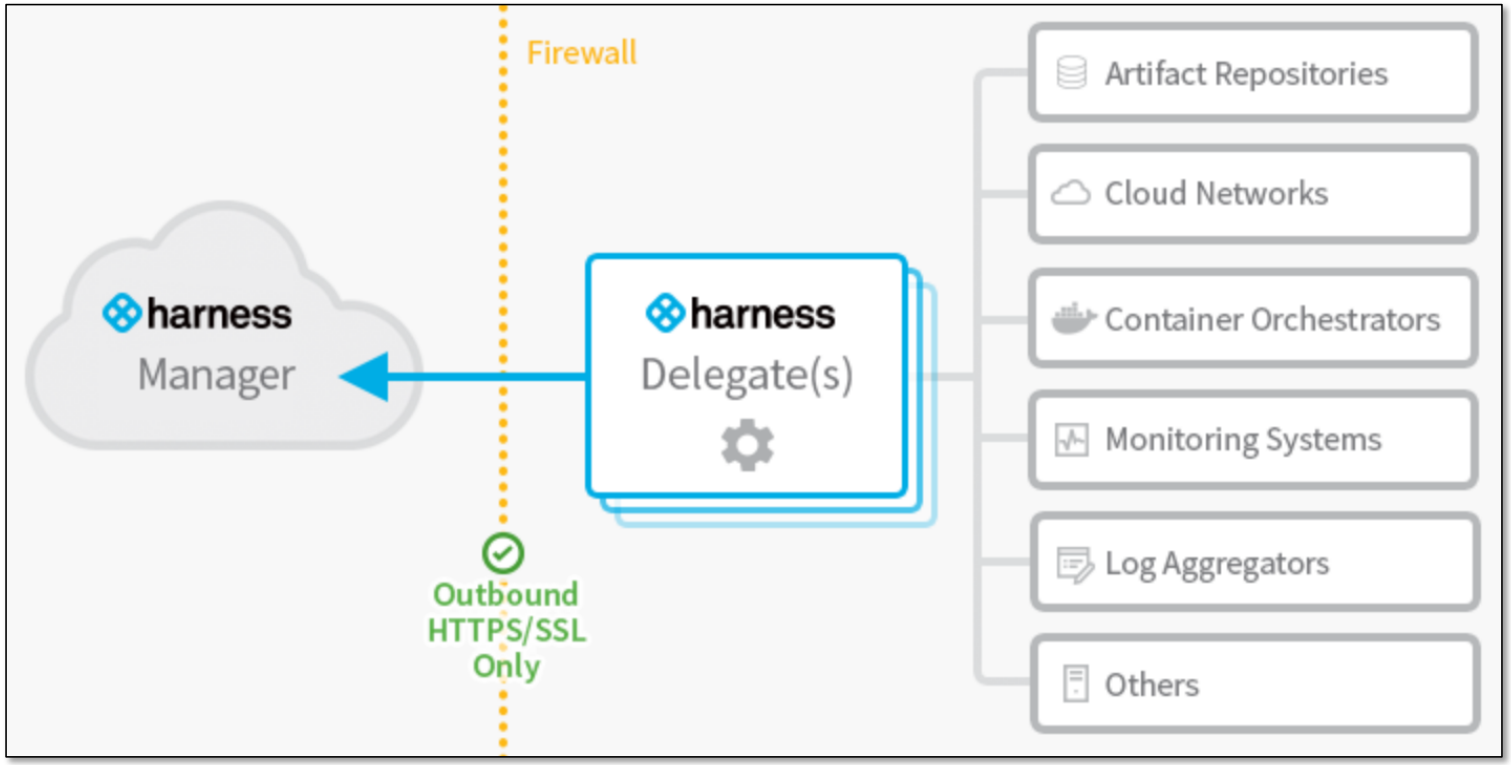




Canary Demo

Let’s Build a Simple K8s Deployment

The Harness.io Platform



The Kubernetes Delegate

- The Delegate yaml files creates the following Kubernetes structures:
  - A namespace named “harness-delegate”
  - A cluster-admin ClusterRoleBinding – this is necessary for the Delegate to spin up deployments in Kubernetes
  - 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

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 Connectors

Setup > Connectors

Artifact Servers

(1) Harness Docker Hub

Verification Providers

(0) None

Source Repo Providers

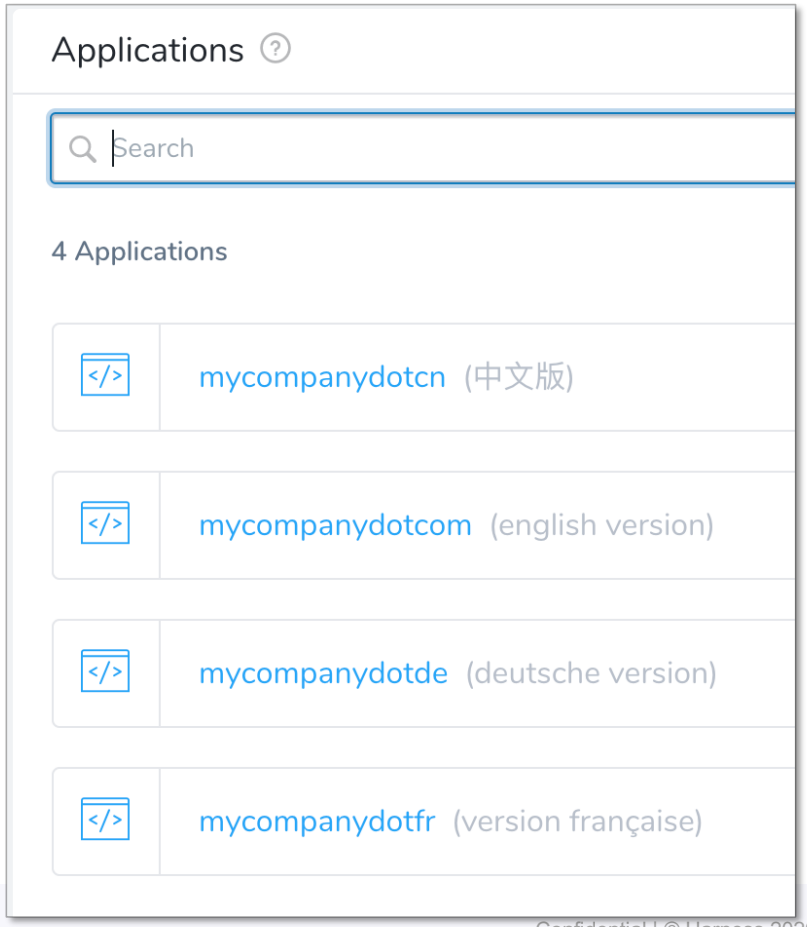
(0) None

Collaboration Providers

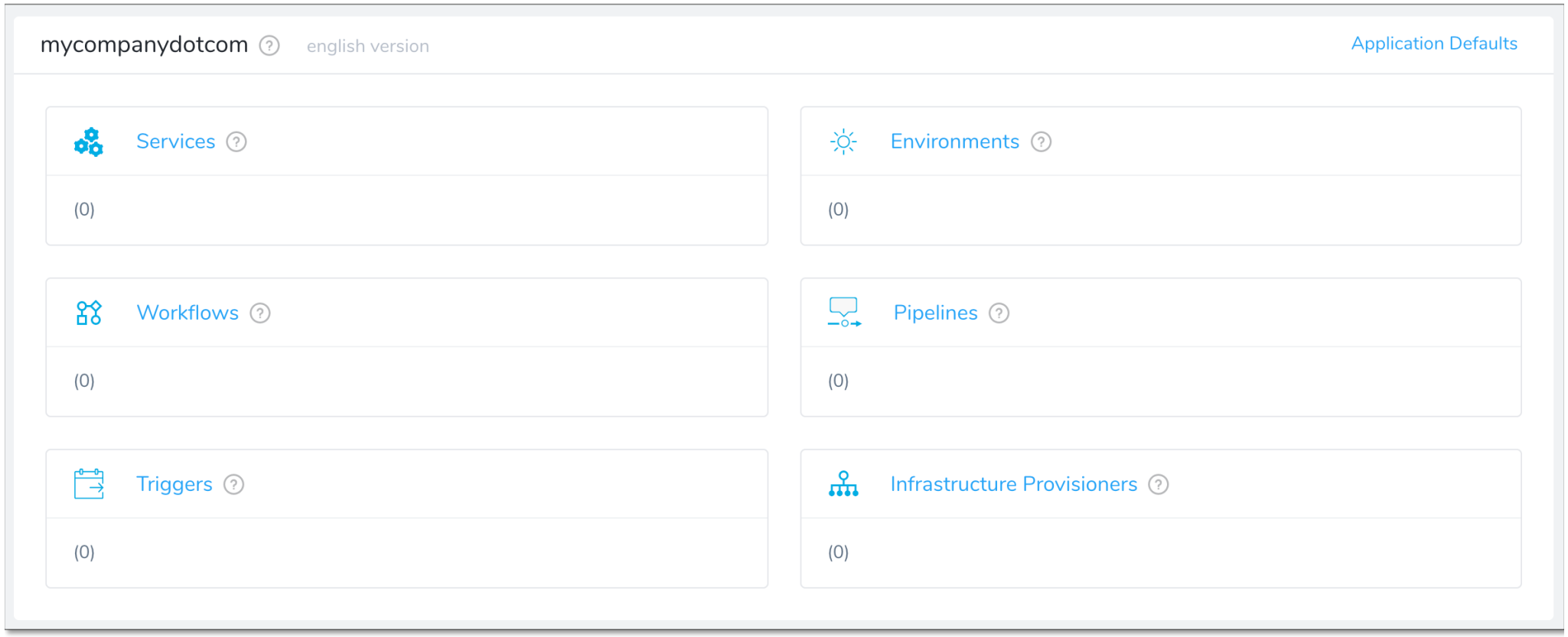
(0) None

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

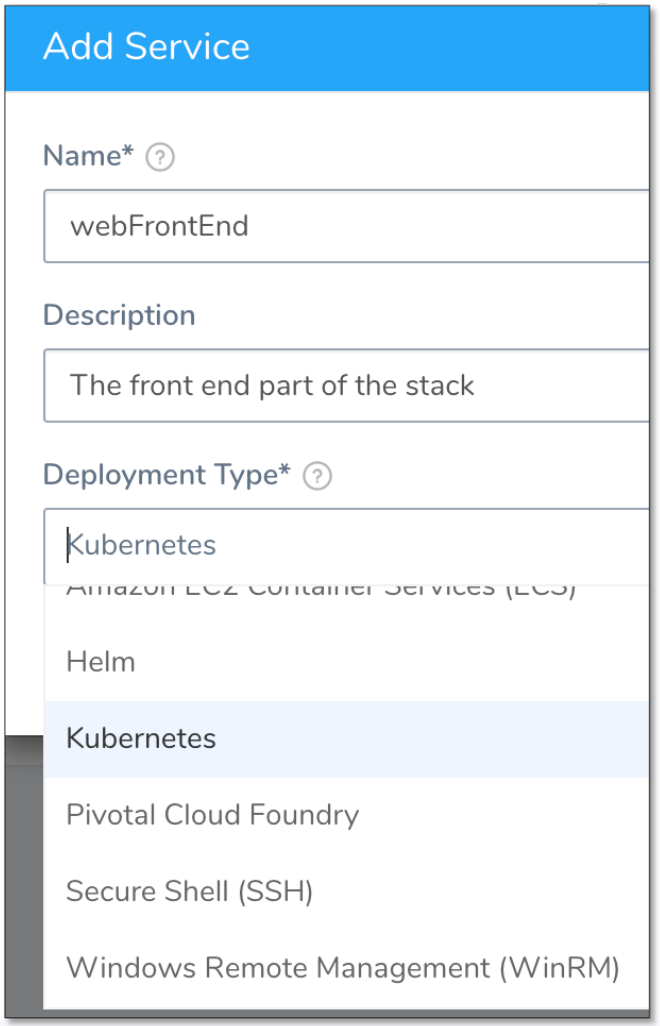


Inside an Application



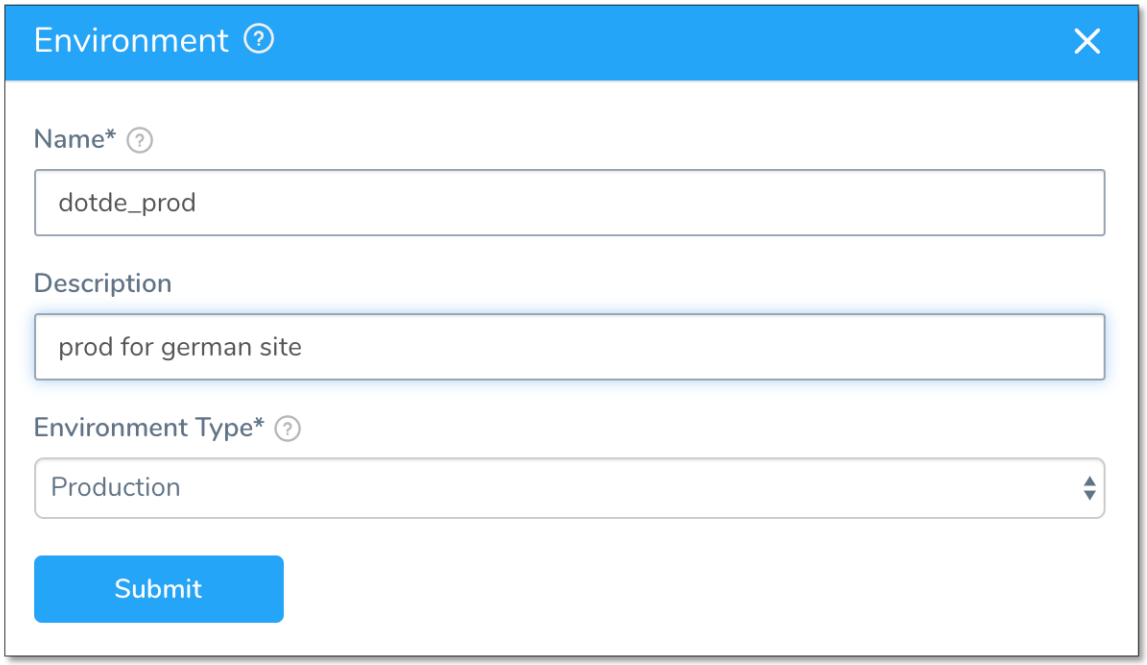
What Are Services

- Services are the individual components / microservices that make up your stack
- In Kubernetes usually represented by a group of yamls to define the service parameters
- In Helm represented by a chart
- See docs for more details https://docs.harness.io/article/i3n6qr8p5i-deployments-overview#deployment_guides

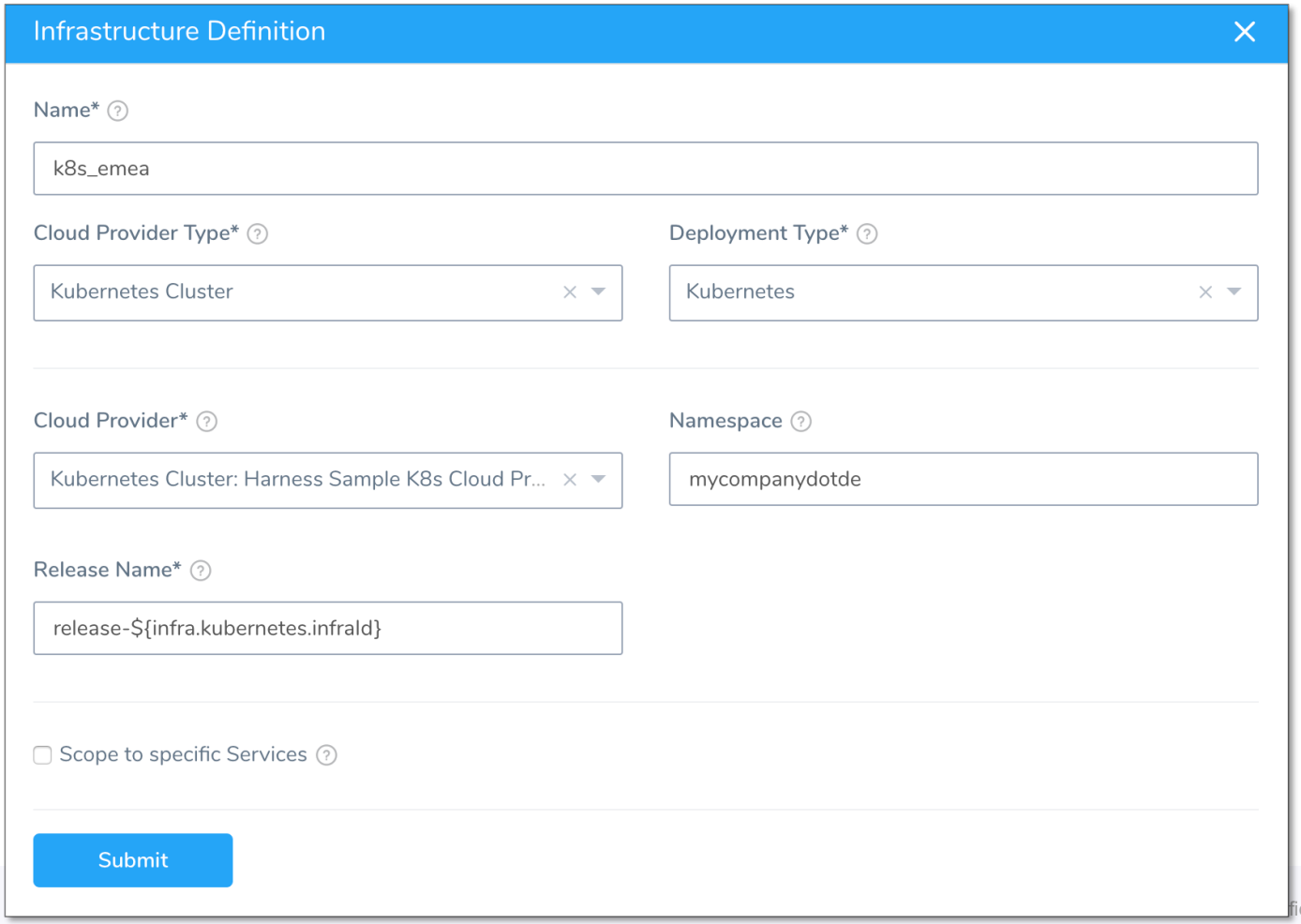


Environments

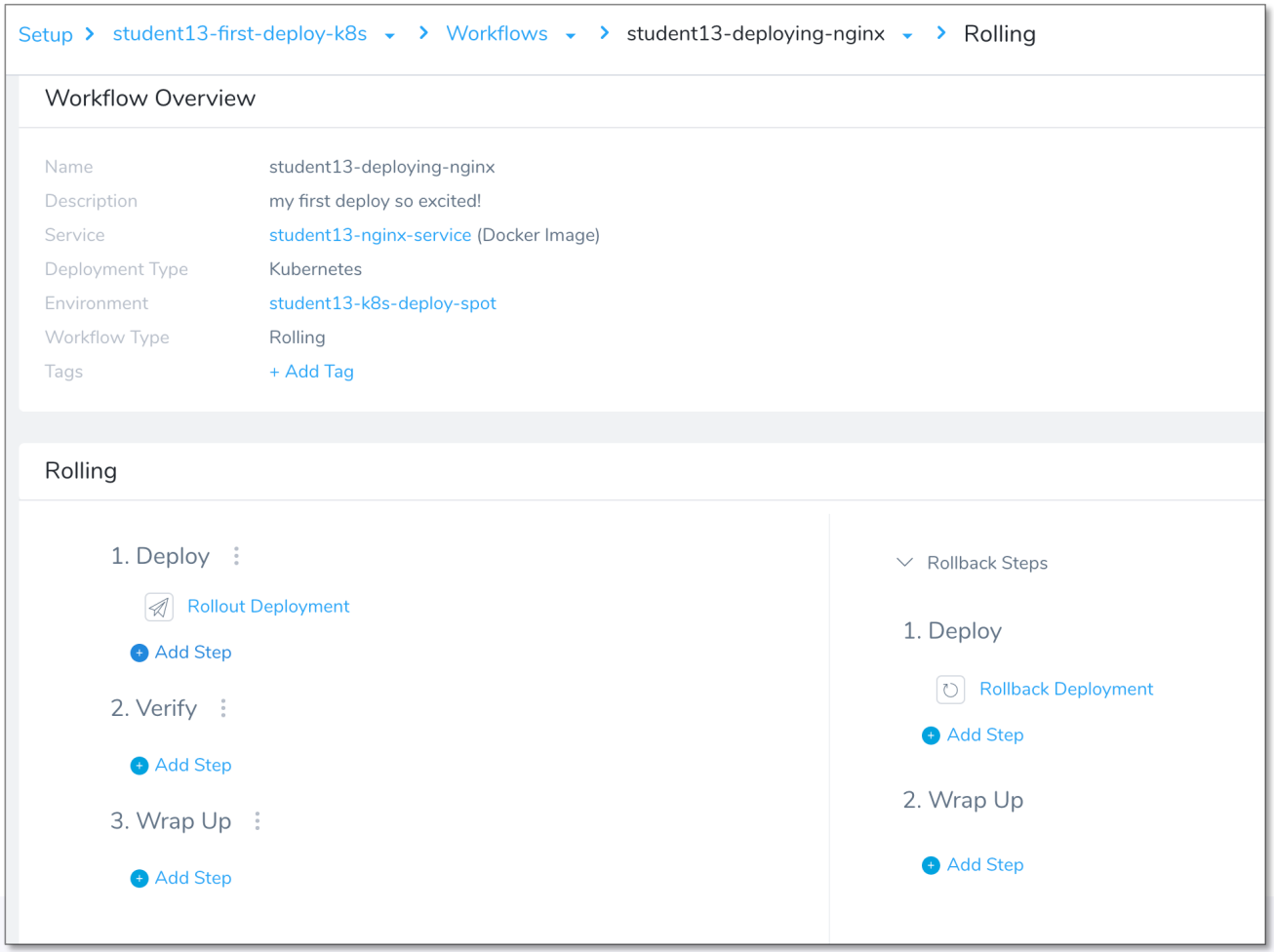
- If Harness Services are what you are deploying, Environments are where you are deploying them to
- For example: Prod, QA, Staging, multi-geo sites for HA and DR, etc.



Environment Infrastructure Definition K8s



Building the Workflow



Do Labs 1-4

Thank you!