Before We Begin

- Mac Folks - Install Brew
  - /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
- Fork and clone repo
  - https://github.com/ithaka/cicd-demo
  - cd cicd-demo
- Install kubectl and kubernetes-helm
  - brew install kubectl kubernetes-helm
  - # other oses : https://kubernetes.io/docs/tasks/tools/install-kubectl/
  - # https://docs.helm.sh/using_helm/#installing-helm

Before We Begin

- minikube and virtualbox
    - brew cask install minikube virtualbox
    - other oses: https://github.com/kubernetes/minikube/releases
    - https://www.virtualbox.org/wiki/Downloads
    - if you already had minikube installed with other settings - minikube stop && minikube delete && rm -rf ~/.minikube
- minikube disk size
    - minikube config set disk-size 40000MB
- Start the cluster
    - minikube start

CI/CD Pipeline Workshop

Ryan Harrington
ryan.harrington@ithaka.org

Tom Green
thomas.green@ithaka.org

Who are we?

- Develop and maintain ~20 microservices
- E-commerce, business to business sales models, fulfillment, auth
- Admin UIs for administering these programs
- Variety of languages
    - Angular, Java (Spring), Node, Ruby (Rspec and Cucumber), Python, Bash, etc.
- Deploy to AWS
- Currently investigating moving to kubernetes

What are we doing today?

- Develop a real CI/CD pipeline
- Use freely available software

Intention:

- To give you a start building pipelines and bringing those to your teams

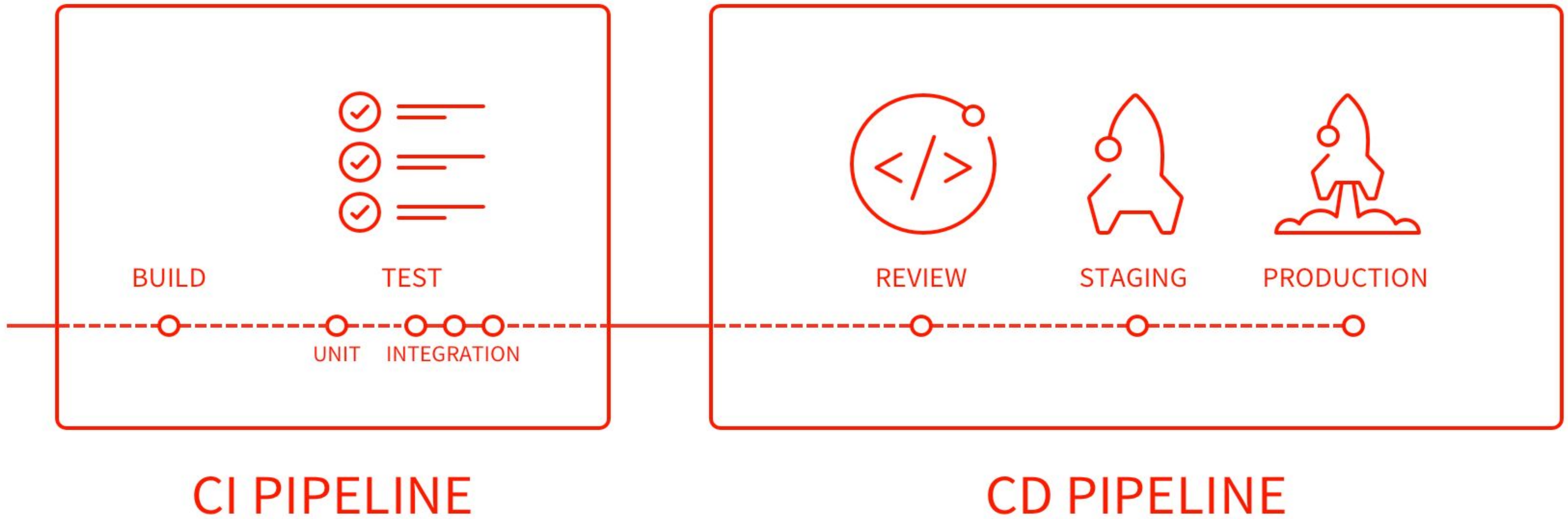
What will not be covered?

- Survey of CI/CD tools
- Everything about CI/CD
- Introduction to Docker and Kubernetes
    - We are expecting you to have had some experience with these technologies and will use some terminology with the understanding that you know what these things are
    - We will try to explain where we can but some previous knowledge is required

What is CI/CD?

- Build, test, and release changes with greater reliability and speed
  - Therefore CI/CD is not strictly about pipelines, or automation.
    - These are the tools, do not conflate this with the process.
  - Many things that facilitate CI/CD will be based on the interactions of people
    - Better and more comprehensive testing
    - A defined workflow
    - Knowledge of the Apps External Dependencies

What is the difference between CI and CD?



Source:
https://docs.gitlab.com/ee/ci/img/cicd_pipeline_infograph.png

Motivation

- **Low risk releases**.
- **Faster time to market**.
- **Higher quality**.
- **Lower costs**.
- **Better products**.
- **Happier teams**.

Source: https://continuousdelivery.com/ - Jez Humble

Shifting Left

In the Typical Dev/Release Process:

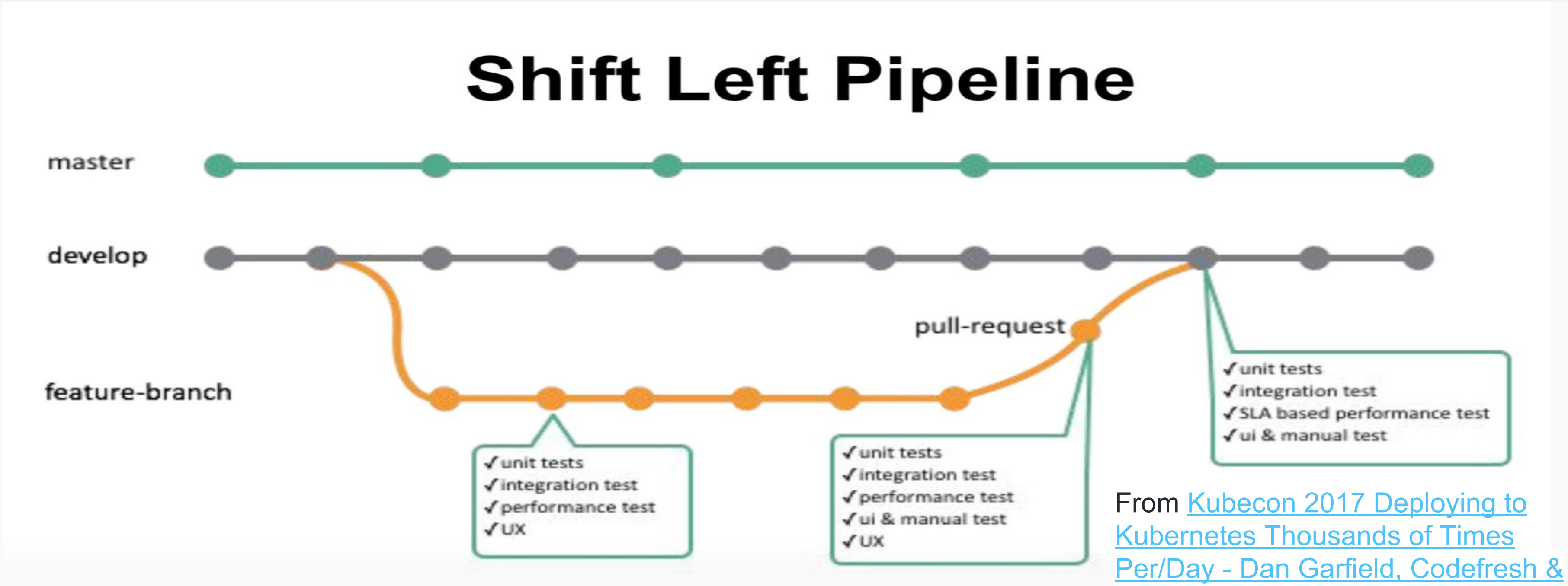
Staging becomes a bottleneck

Solution is to Shift Left so earlier feedback is continuously provided on the feature branch.

“Why give your release candidate to QA to find regressions?”

From Kubecon 2017 Deploying to Kubernetes Thousands of Times Per/Day - Dan Garfield, Codefresh & William Denniss, Google

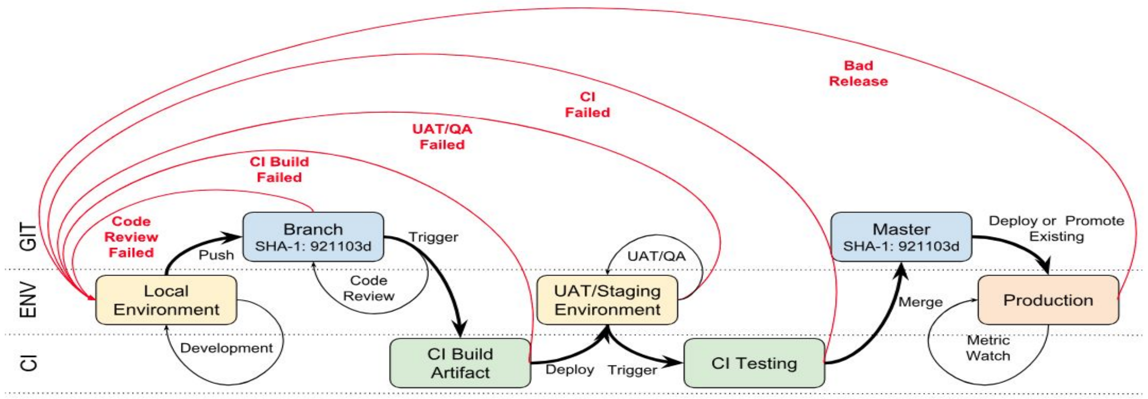
Shifting Left



From [Kubecon 2017 Deploying to Kubernetes Thousands of Times Per/Day - Dan Garfield, Codefresh & William Denniss, Google]

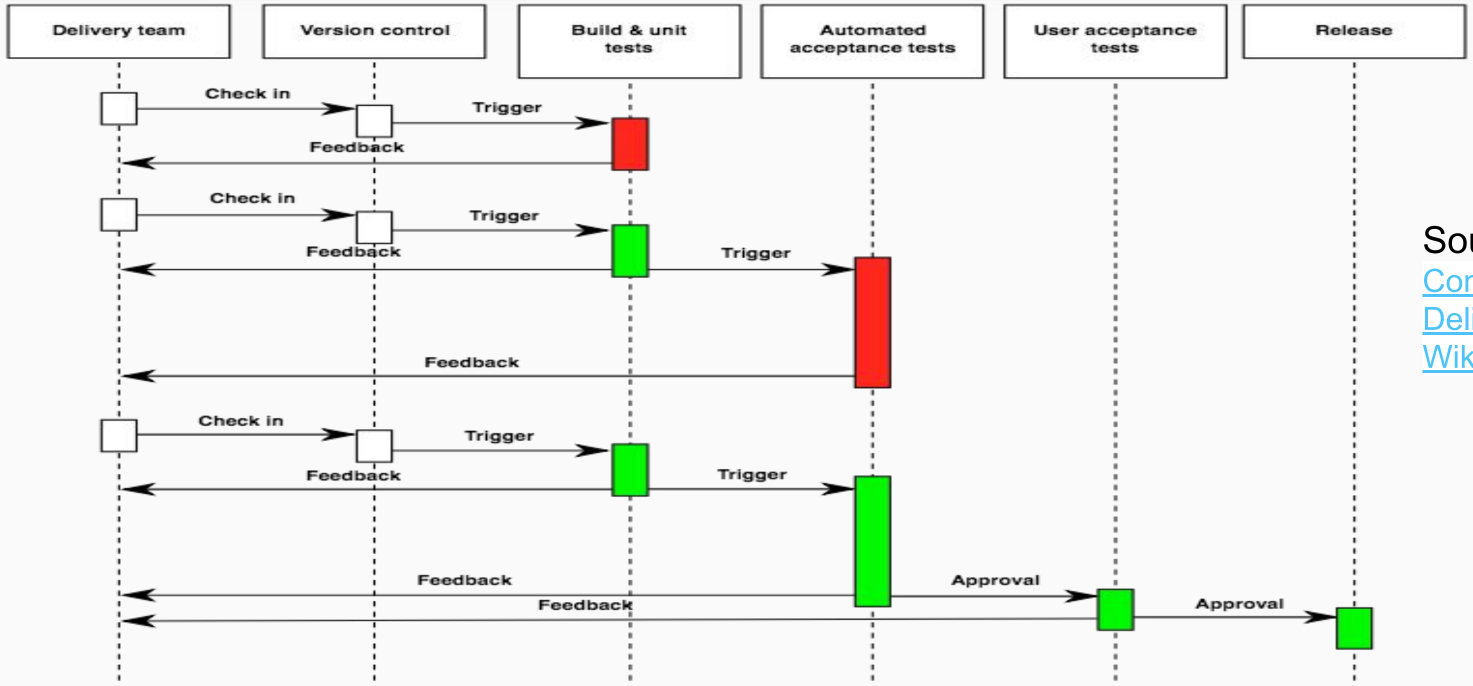
A typical development WF

Workflow Based on Pull Request



Artifact built from commit and tagged with same SHA-1 hash (ex. 921103d)
Identical Immutable artifact is deployed and tested in both staging and prod.
Whether or when commit is on master is a team decision.
Pipeline automation facilitates automatic merging.
Production is able to be rolled back quickly.

As A Process Flow



Source: [Continuous Delivery on Wikipedia]

Let’s Automate a Pipeline

docker

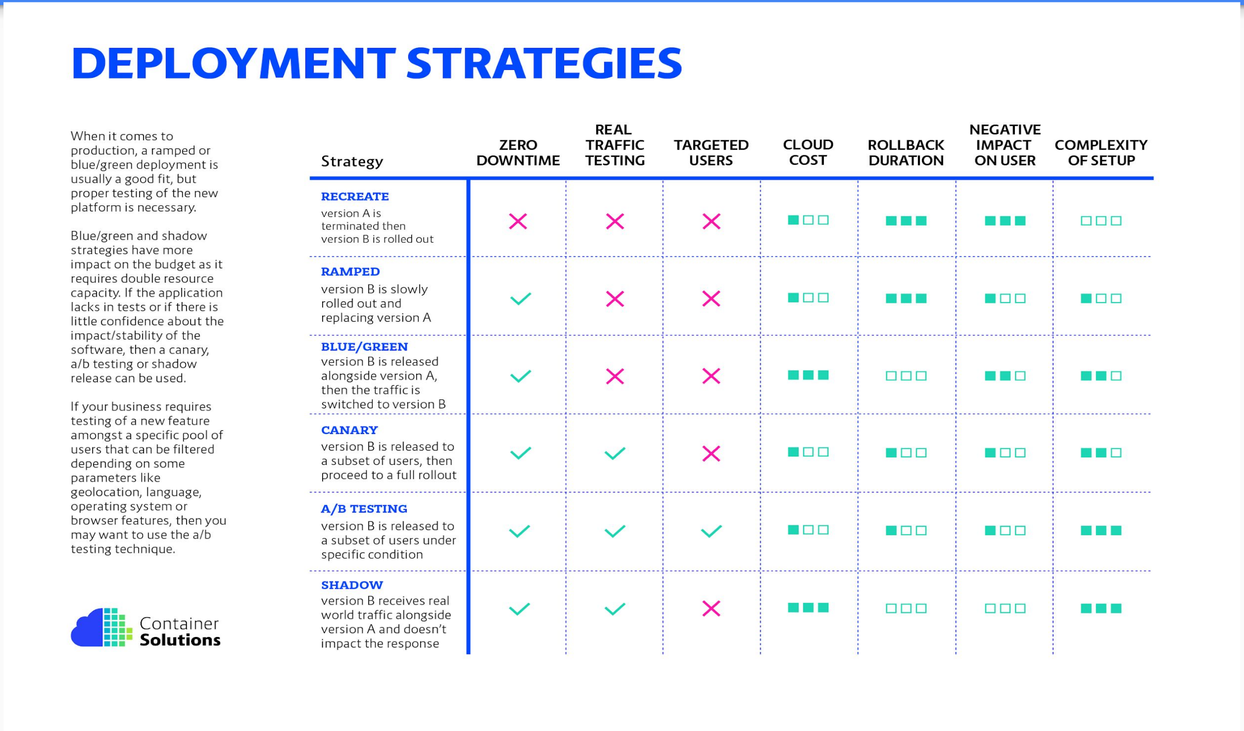
HELM

Jenkins

Demo App

- Web Service written in JS
- Jest Unit Tests
- Integration Tests with Rspec

Summary - Deployment Strategies



DEPLOYMENT STRATEGIES

When it comes to production, a ramped or blue/green deployment is usually a good fit, but proper testing of the new platform is necessary.

Blue/green and shadow strategies have more impact on the budget as it requires double resource capacity. If the application lacks in tests or if there is little confidence about the impact/stability of the software, then a canary, a/b testing or shadow release can be used.

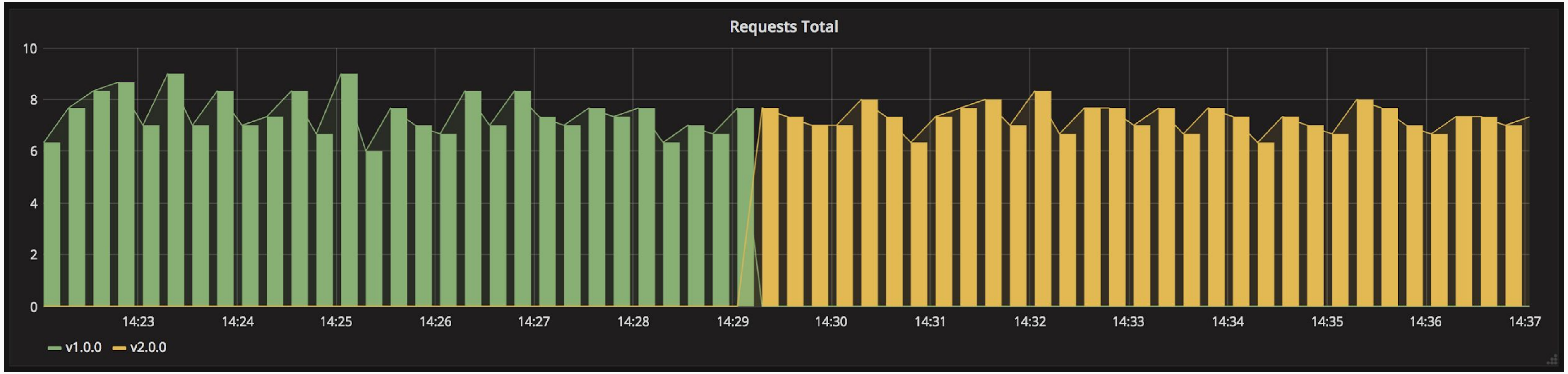
If your business requires testing of a new feature amongst a specific pool of users that can be filtered depending on some parameters like geolocation, language, operating system or browser features, then you may want to use the a/b testing technique.

Strategy	ZERO DOWNTIME	REAL TRAFFIC TESTING	TARGETED USERS	CLOUD COST	ROLLBACK DURATION	NEGATIVE IMPACT ON USER	COMPLEXITY OF SETUP
RECREATE version A is terminated then version B is rolled out	✗	✗	✗	■□□	■■■	■■■	□□□
RAMPED version B is slowly rolled out and replacing version A	✓	✗	✗	■□□	■■■	■□□	■□□
BLUE/GREEN version B is released alongside version A, then the traffic is switched to version B	✓	✗	✗	■■■	□□□	■■□	■■□
CANARY version B is released to a subset of users, then proceed to a full rollout	✓	✓	✗	■□□	■□□	■□□	■■□
A/B TESTING version B is released to a subset of users under specific condition	✓	✓	✓	■□□	■□□	■□□	■■■
SHADOW version B receives real world traffic alongside version A and doesn't impact the response	✓	✓	✗	■■■	□□□	□□□	■■■

Container Solutions

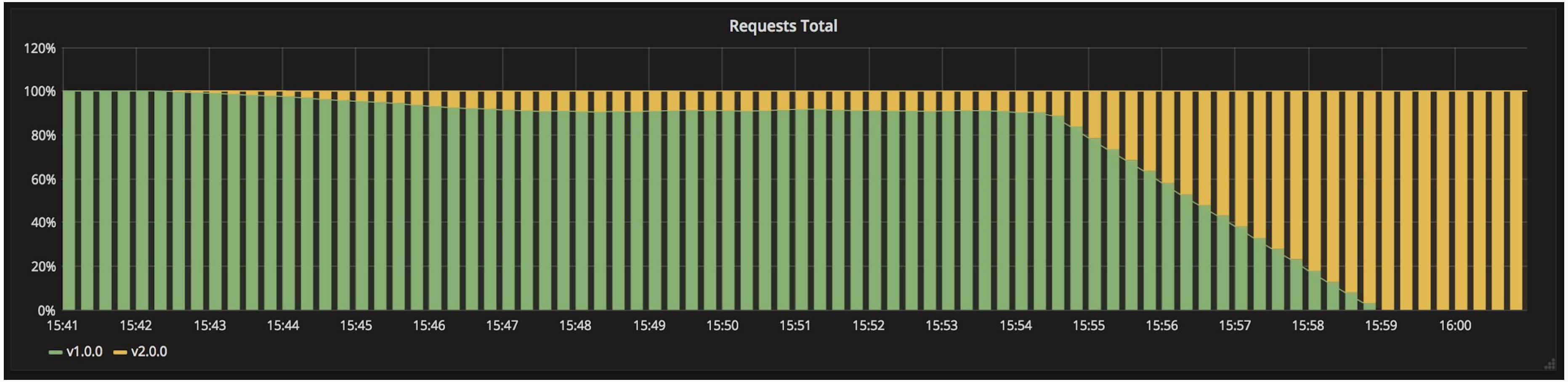
Summary - Deployment Strategies

Blue/Green (green/yellow)



Summary - Deployment Strategies

Canary



Summary

Take Away:

- Get Started - automate locally, you can automate this on your machine using the tools shown and then spread these concepts to the rest of your team

Summary

- Traps
    - Trust your verification
    - Biting off too much
    - Not knowing your Apps Dependencies
    - Not knowing your workflow

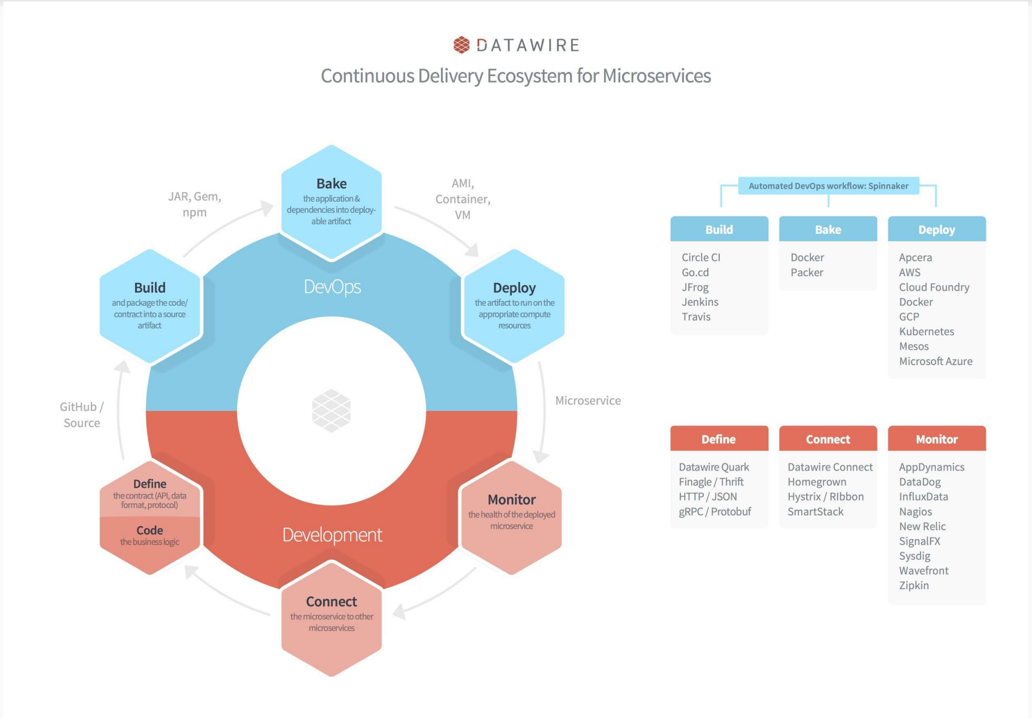
Resources

Alternative Tools

Continuous Delivery:
JenkinsX
GitLab CI/CD

Bake:
Kaniko

Deploy:
WeaveWorks Flux
Skaffold
Helm


DATAWIRE
Continuous Delivery Ecosystem for Microservices
Bake
the application & dependencies into deploy-able artifact
JAR, Gem, npm
AMI, Container, VM
Build
and package the code/ contract into a source artifact
DevOps
Deploy
the artifact to run on the appropriate compute resources
GitHub / Source
Microservice
Define
the contract (API, data format, protocol)
Code
the business logic
Development
Monitor
the health of the deployed microservice
Connect
the microservice to other microservices
Automated DevOps workflow: Spinnaker

Build	Bake	Deploy
Circle CI Go.cd JFrog Jenkins Travis	Docker Packer	Apcera AWS Cloud Foundry Docker GCP Kubernetes Mesos Microsoft Azure

Define	Connect	Monitor
Datawire Quark Finagle / Thrift HTTP / JSON gRPC / Protobuf	Datawire Connect Homegrown Hystrix / RIbbon SmartStack	AppDynamics DataDog InfluxData Nagios New Relic SignalFX Sysdig Wavefront Zipkin

Related Articles:

CICD With Jenkins K8S

K8S Jenkins Azure Helm

Jenkins Pipeline Helm K8S

K8S Azure Jenkins Helm