PROGRESSIVE DELIVERY IN KUBERNETES

Carlos Sanchez / csanchez.org / @csanchez

Viktor Farcic / technologyconversations.com / @vfarcic

Cloud Engineer @ Adobe

Author of Jenkins Kubernetes plugin

Long time OSS contributor at Apache Maven, Eclipse, Puppet,…

Principal Software Delivery Strategist @ CloudBees

Author of The DevOps Toolkit Series

and Test-Driven Java Development

PROGRESSIVE DELIVERY

Progressive Delivery, a History.... Condensed

By Adam Zimman - August 6, 2018

RedMonk

the developer-focused industry analyst firm

JAMES GOVERNOR'S MONKCHIPS

Towards Progressive Delivery

By James Governor | @monkchips | August 6, 2018

Progressive Delivery is a term that includes deployment strategies that try to avoid the pitfalls of all-or-nothing deployment strategies

New versions being deployed do not replace existing versions but run in parallel for an amount of time receiving live production traffic, and are evaluated in terms of correctness and performance before the rollout is considered successful.

Continuous Delivery is hard

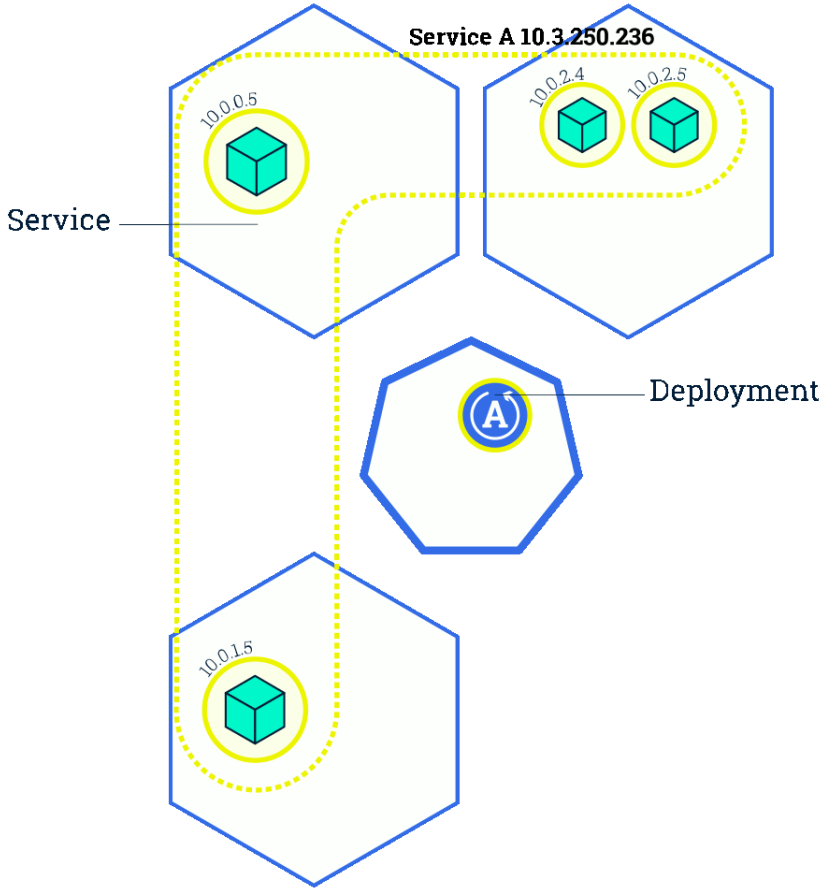
Progressive Delivery makes Continuous Delivery easier to adopt

reduces the risk associated with Continuous Delivery

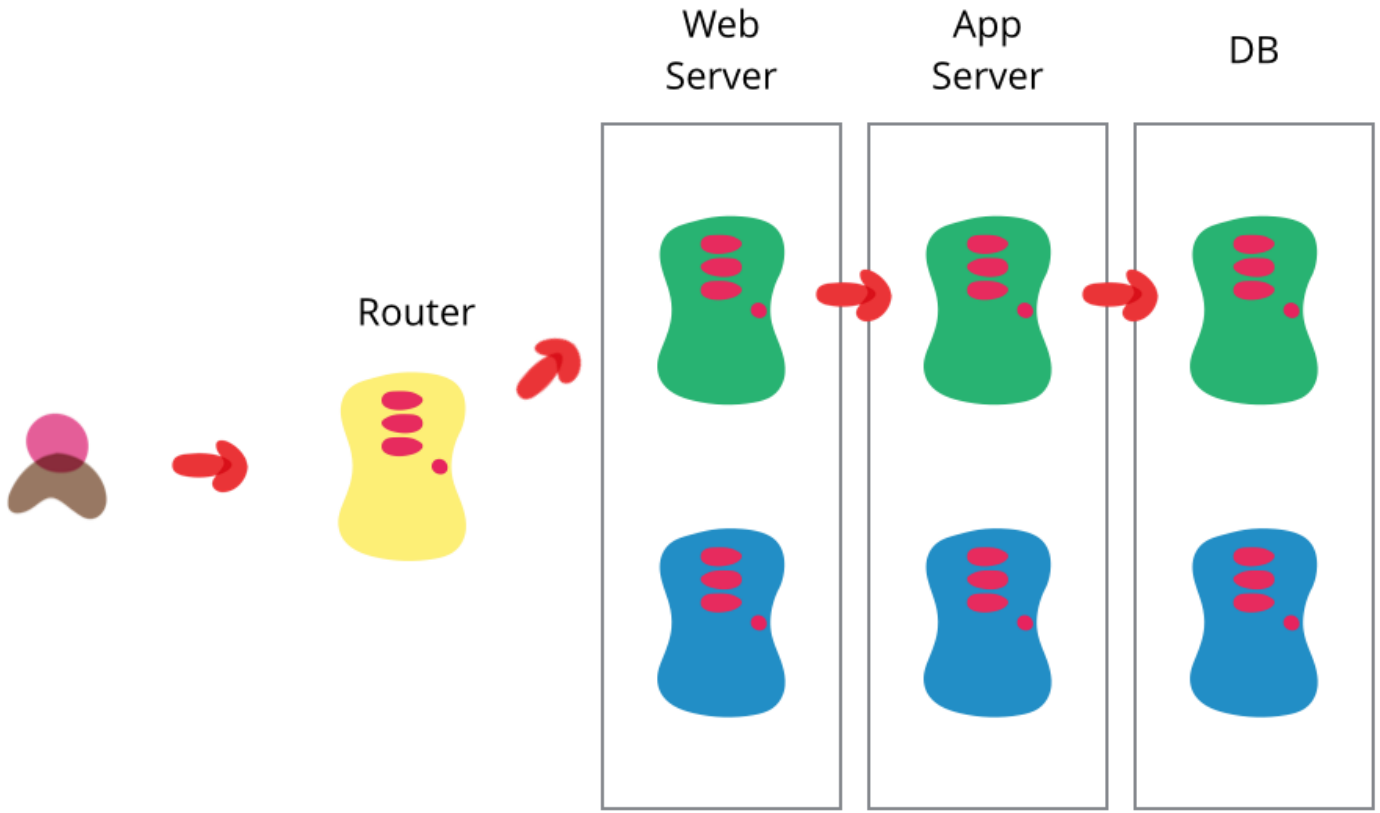
- Avoiding downtime
- Limit the blast radius
- Shorter time from idea to production

PROGRESSIVE DELIVERY TECHNIQUES

ROLLING UPDATES

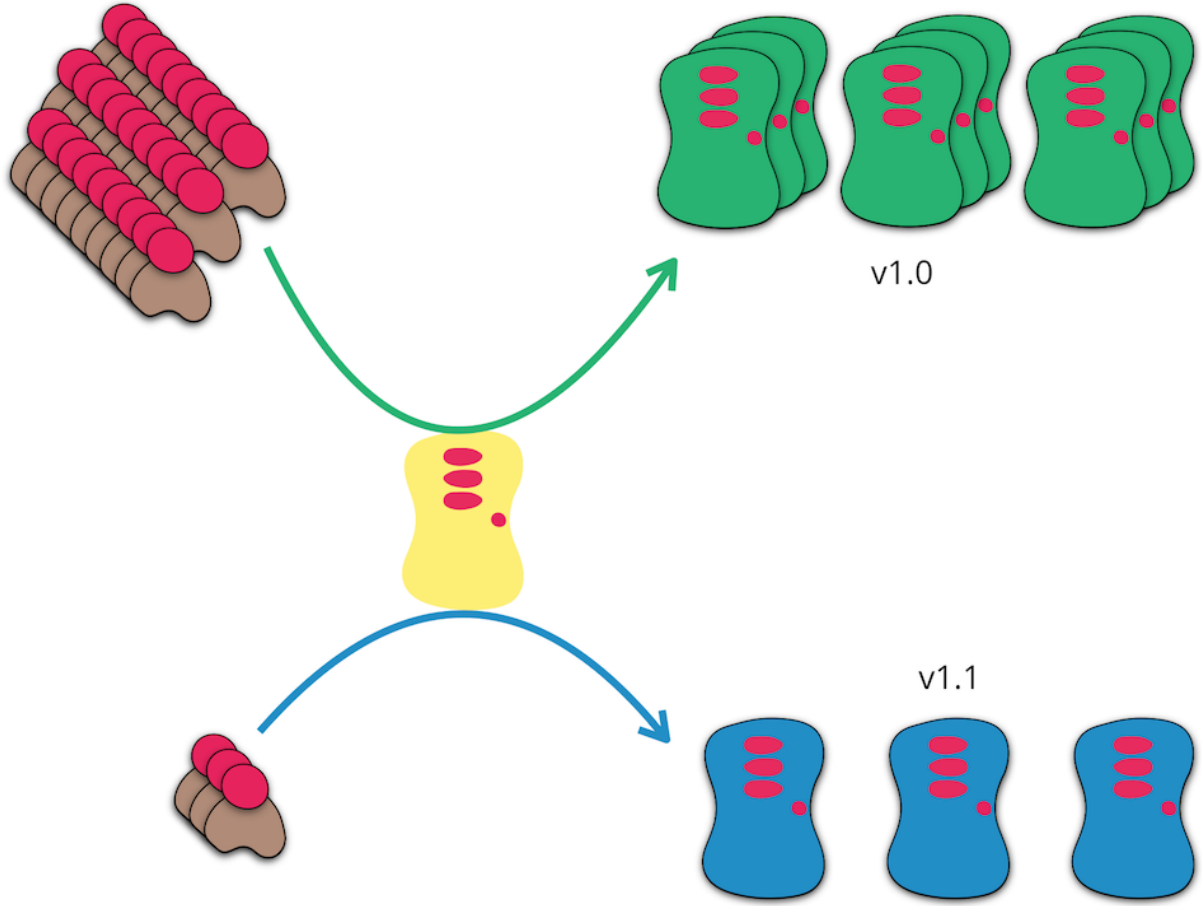


BLUE-GREEN DEPLOYMENT



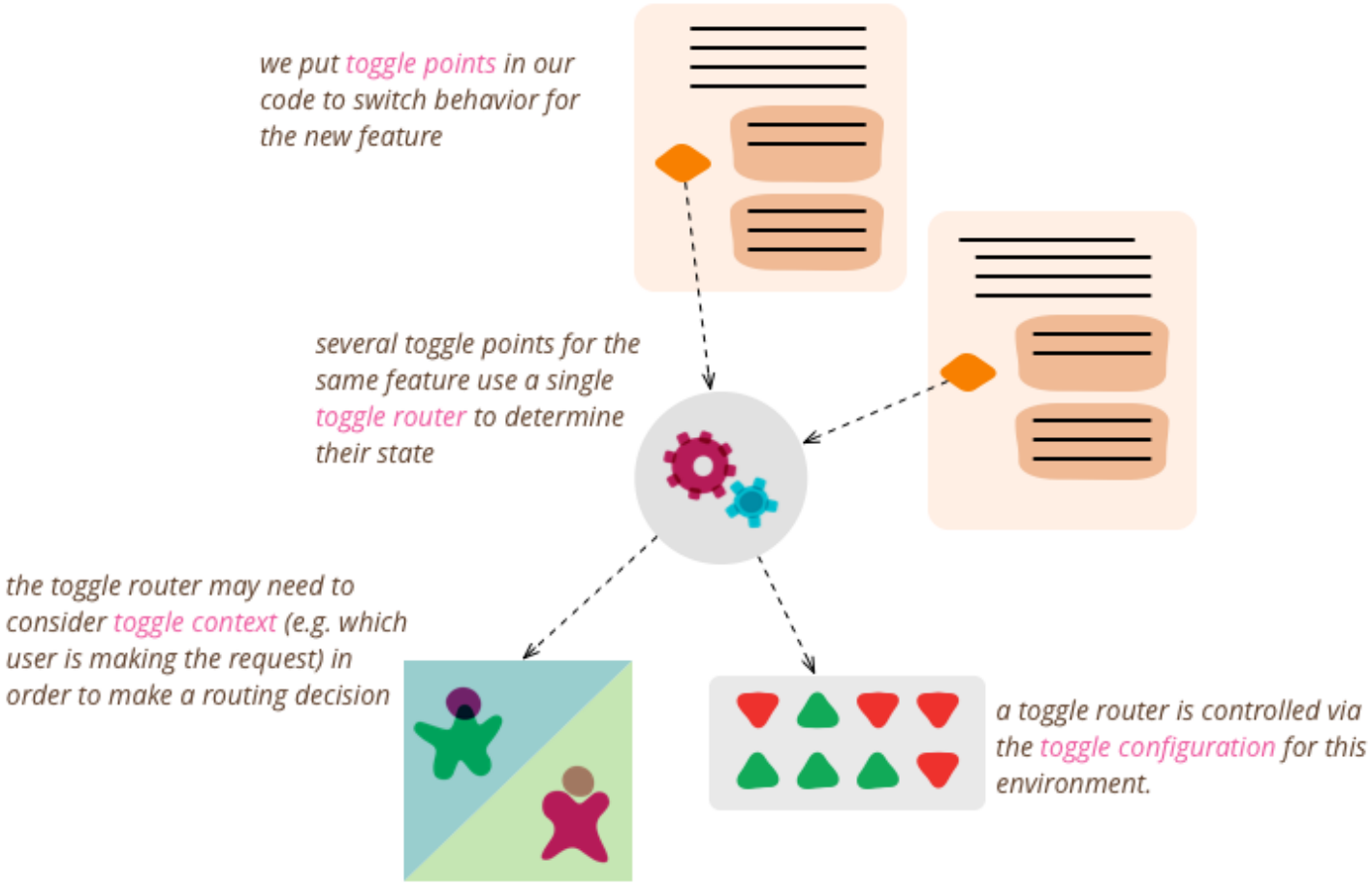
medium.com/continuous-deployment/continuous-deployment-strategies-32e2f7badd2

CANARY DEPLOYMENT



medium.com/continuous-deployment/continuous-deployment-strategies-32e2f7badd2

FEATURE FLAGS



Martin Fowler martinfowler.com/articles/feature-toggles.html


fex

MONITORING IS THE NEW TESTING

Know when users are experiencing issues **in production**

React to the issues **automatically**

Progressive Delivery requires a good amount of metrics

@DEVOPS_BORAT

DevOps Borat

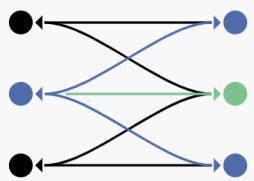
To make error is human. To propagate error to all server in automatic way is #devops.

If you haven't automatically destroyed something by mistake, you are not automating enough

PROGRESSIVE DELIVERY

Istio

Connect, secure, control, and observe services.

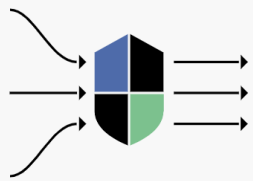


Connect

Intelligently control the flow of traffic and API calls between services, conduct a range of tests, and upgrade gradually with red/black deployments.

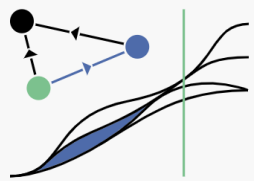
Secure

Automatically secure your services through managed authentication, authorization, and encryption of communication between services.



Control

Apply policies and ensure that they're enforced, and that resources are fairly distributed among consumers.



Observe

See what's happening with rich automatic tracing, monitoring, and logging of all your services.

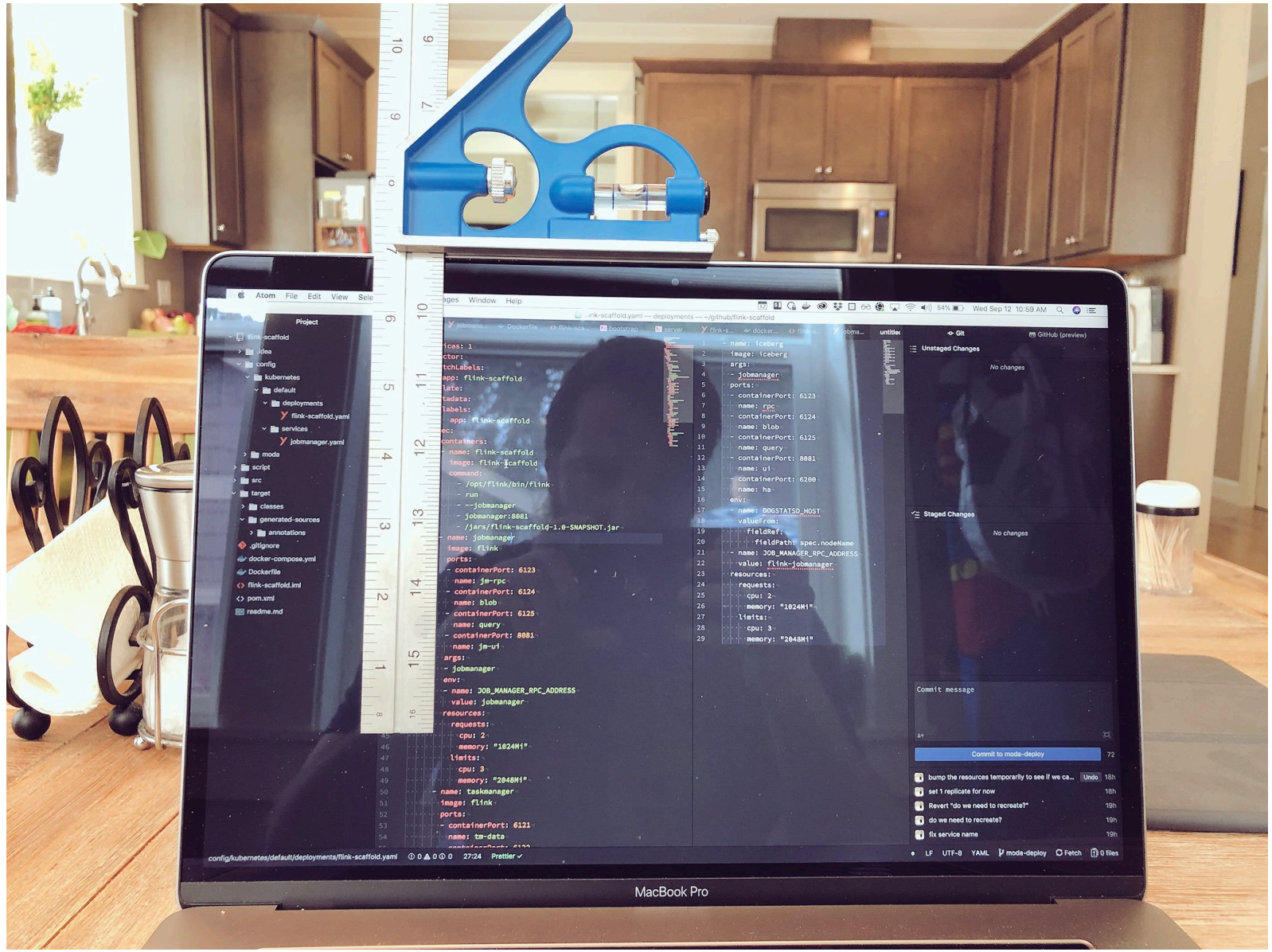
PROMETHEUS

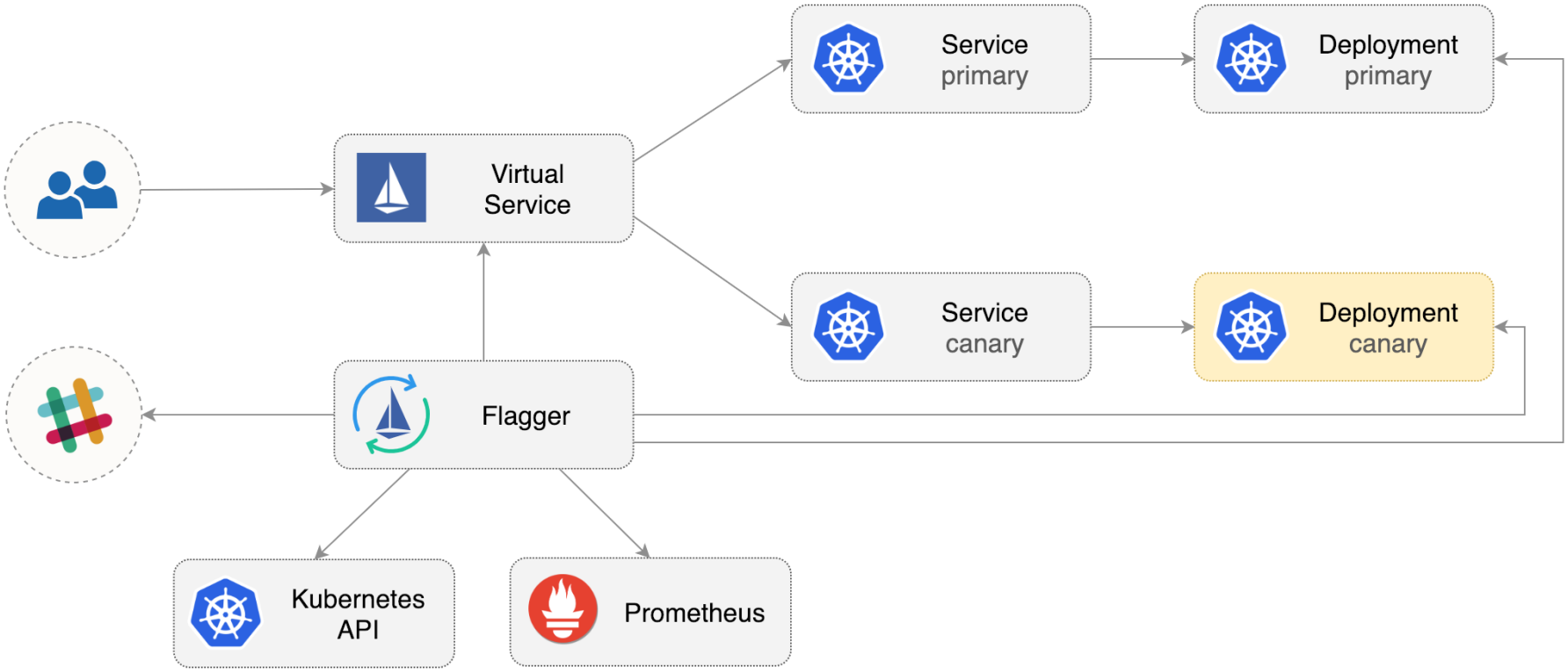
A systems monitoring and alerting toolkit

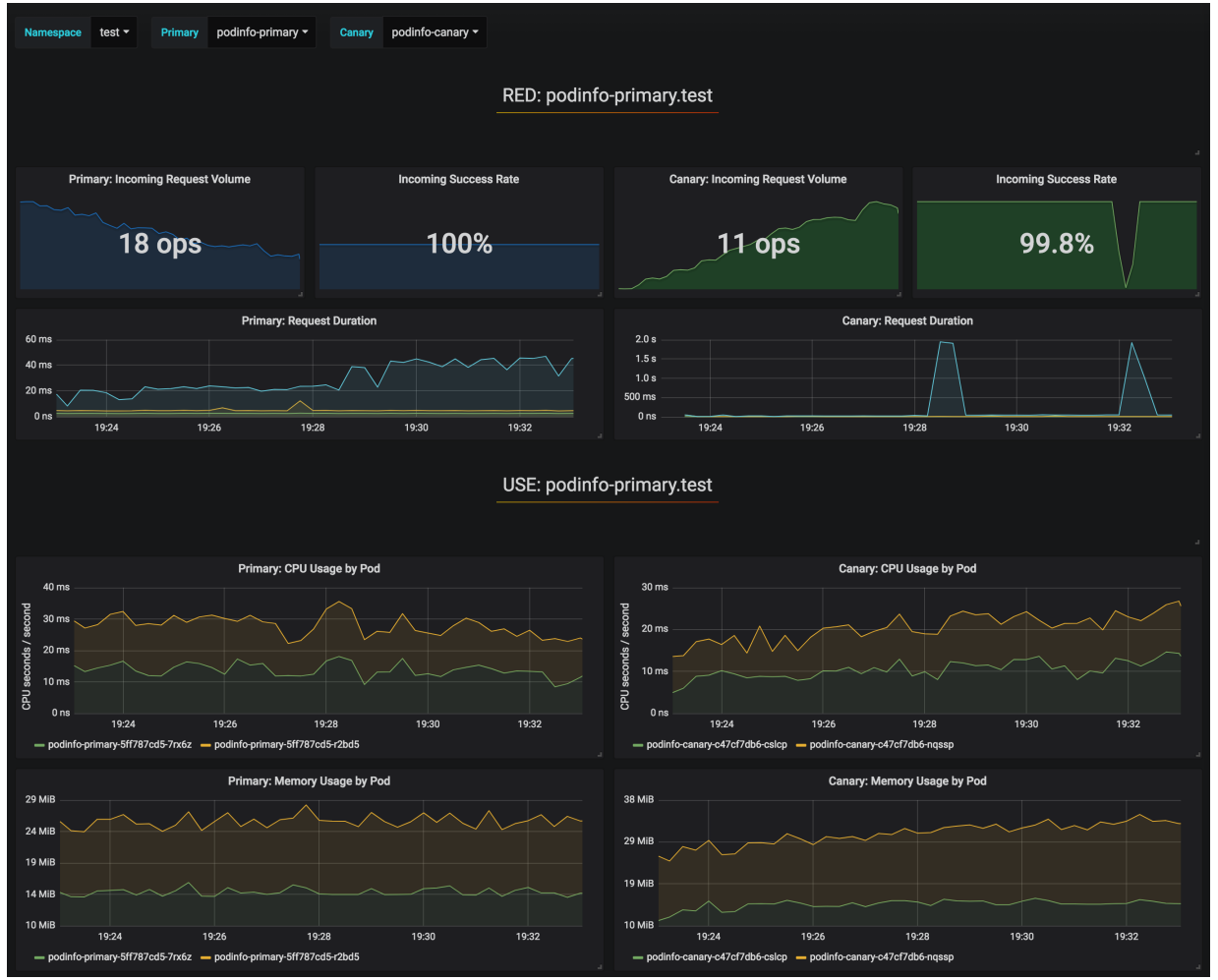
FLAGGER

flagger.app

automates the promotion of canary deployments by using Istio's traffic shifting and Prometheus metrics to analyse the application's behaviour during a controlled rollout


Project
flink-scaffold
.idea
config
kubernetes
default
deployments
flink-scaffold.yaml
services
jobmanager.yaml
moda
script
src
target
classes
generated-sources
annotations
.gitignore
docker-compose.yml
Dockerfile
flink-scaffold.iml
pom.xml
readme.md
Git
GitHub (preview)
Unstaged Changes
No changes
Staged Changes
No changes
Commit message
Commit to moda-deploy
bump the resources temporarily to see if we ca...
Undo
set 1 replicate for now
Revert "do we need to recreate?"
do we need to recreate?
fix service name
MacBook Pro


Service
primary
Deployment
primary
Virtual
Service
Service
canary
Deployment
canary
Flagger
Kubernetes
API
Prometheus


Namespace
test
Primary
podinfo-primary
Canary
podinfo-canary
RED: podinfo-primary.test
Primary: Incoming Request Volume
18 ops
Incoming Success Rate
100%
Canary: Incoming Request Volume
11 ops
Incoming Success Rate
99.8%
Primary: Request Duration
60 ms
40 ms
20 ms
0 ns
19:24
19:26
19:28
19:30
19:32
Canary: Request Duration
2.0 s
1.5 s
1.0 s
500 ms
0 ns
19:24
19:26
19:28
19:30
19:32
USE: podinfo-primary.test
Primary: CPU Usage by Pod
CPU seconds / second
40 ms
30 ms
20 ms
10 ms
0 ns
19:24
19:26
19:28
19:30
19:32
podinfo-primary-5ff787cd5-7rx6z
podinfo-primary-5ff787cd5-r2bd5
Canary: CPU Usage by Pod
CPU seconds / second
30 ms
20 ms
10 ms
0 ns
19:24
19:26
19:28
19:30
19:32
podinfo-canary-c47cf7db6-cslcp
podinfo-canary-c47cf7db6-nqssp
Primary: Memory Usage by Pod
29 MiB
24 MiB
19 MiB
14 MiB
10 MiB
19:24
19:26
19:28
19:30
19:32
podinfo-primary-5ff787cd5-7rx6z
podinfo-primary-5ff787cd5-r2bd5
Canary: Memory Usage by Pod
38 MiB
29 MiB
19 MiB
10 MiB
19:24
19:26
19:28
19:30
19:32
podinfo-canary-c47cf7db6-cslcp
podinfo-canary-c47cf7db6-nqssp

flagger APP 3:30 PM

podinfo.test

New revision detected, starting canary analysis.

Target

Deployment/podinfo.test

Traffic routing

Weight step: 5 max: 50

Failed checks threshold

10

Progress deadline

60s

podinfo.test

Canary analysis completed successfully, promotion finished.

flagger APP 12:12 PM

podinfo.test

Progress deadline exceeded deployment does not have minimum availability for more than 60s

flagger APP 12:18 PM

podinfo.test

Failed checks threshold reached 10

csanchez.org


csanchez


carlossg

technologyconversations.com


vfarcic


vfarcic