PROGRESSIVE DELIVERY IN KUBERNETES

Carlos Sanchez / csanchez.org / @csanchez

Adobe

Cloud Engineer @ Adobe

Author of Jenkins Kubernetes plugin

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

PROGRESSIVE DELIVERY

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.

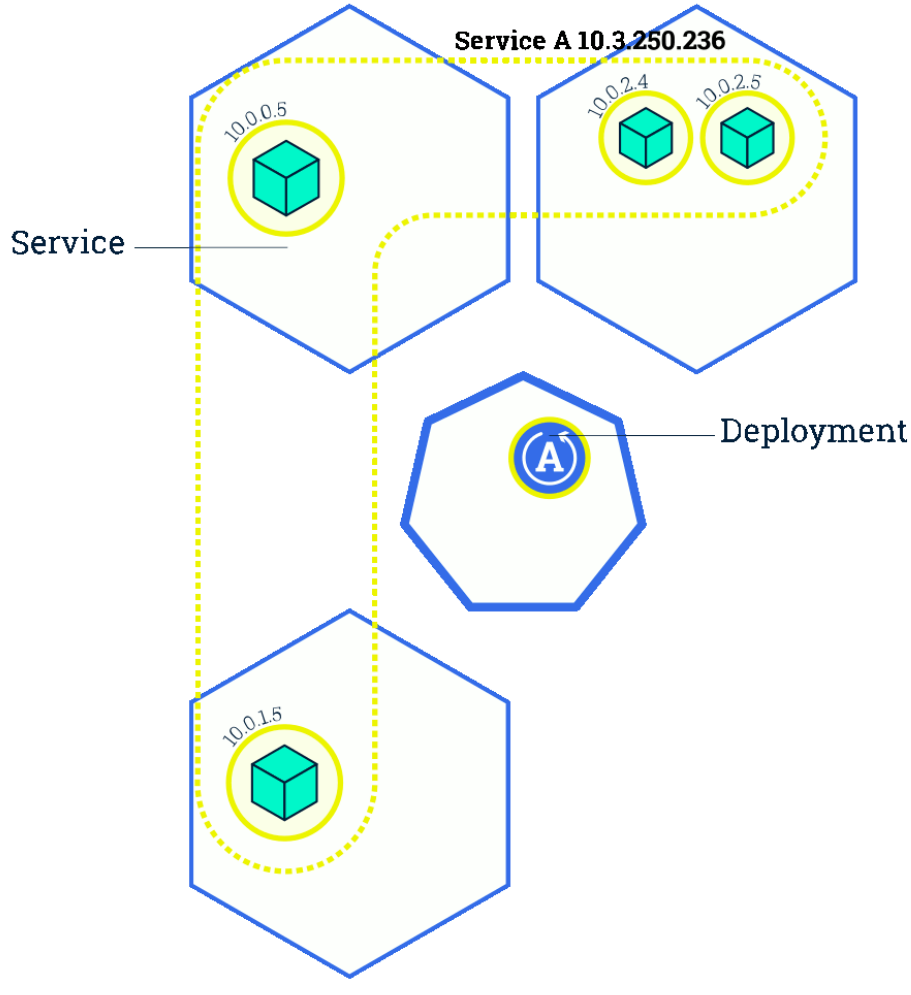
Continuous Delivery is hard

Progressive Delivery makes Continuous Delivery easier to adopt

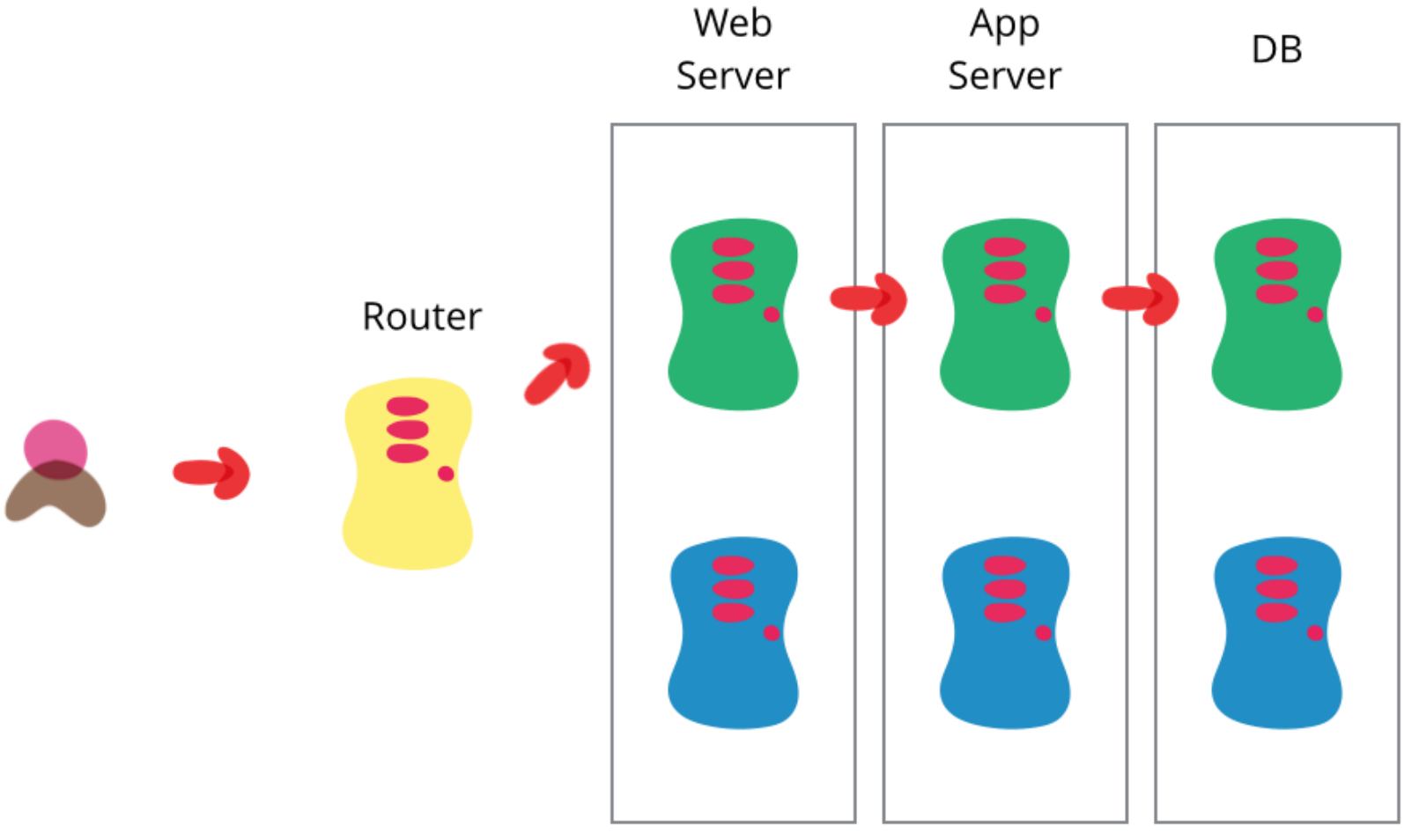
reduces the risk associated with Continuous Delivery

PROGRESSIVE DELIVERY TECHNIQUES

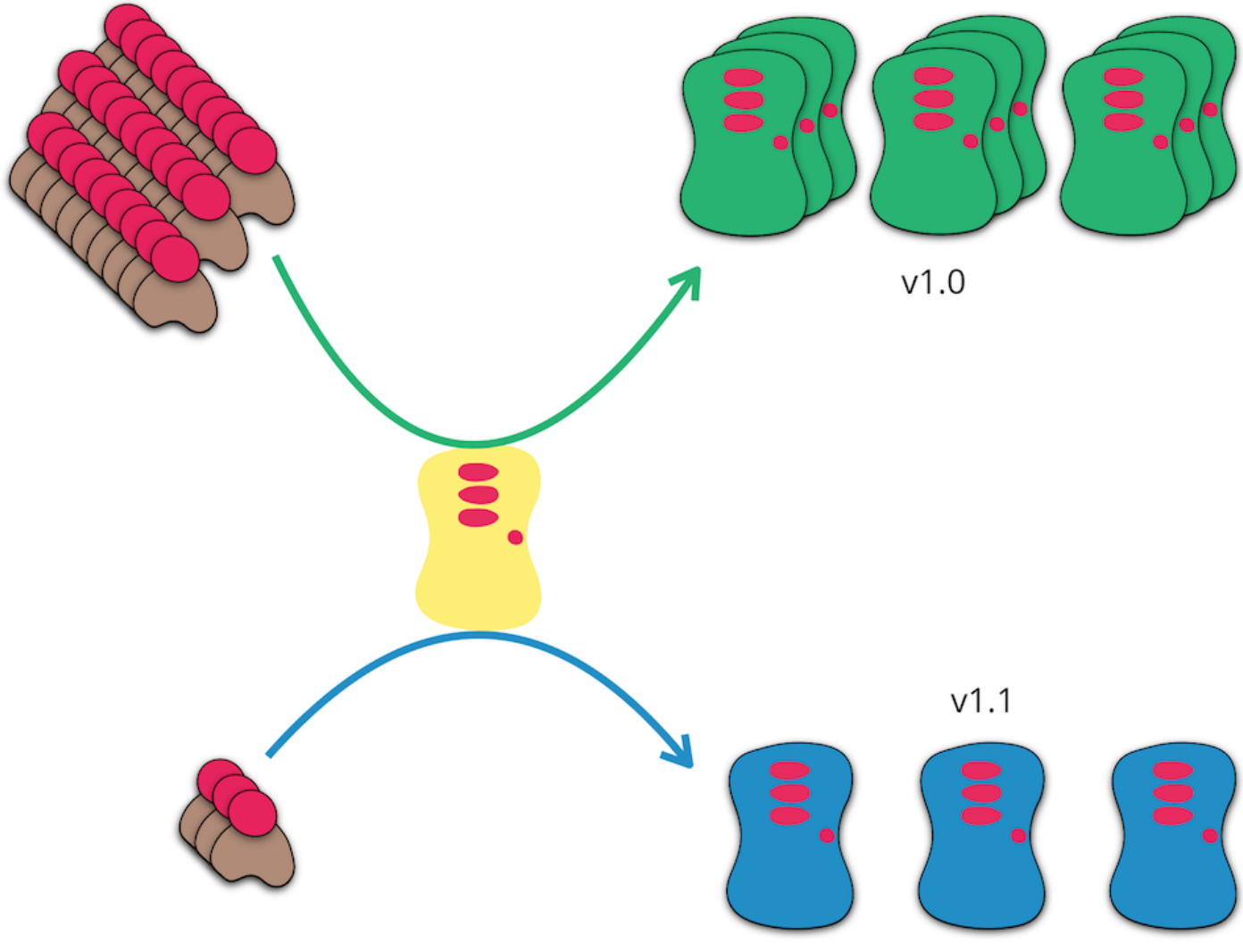
ROLLING UPDATES


Service A 10.3.250.236
10.0.0.5
10.0.2.4
10.0.2.5
Service
Deployment
A
10.0.1.5

BLUE-GREEN DEPLOYMENT



CANARY DEPLOYMENT



MONITORING IS THE NEW TESTING

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

React to the issues **automatically**

JENKINS X


kubernetes


docker


Nexus


CHARTMUSEUM


SKAFFOLD


DRAFT


JENKINSX


prow


HELM

TEKTON

Pipeline engine in Kubernetes

Uses Pods and containers to run the pipeline steps

Implements ChatOps

Handles GitHub webhooks

Package manager for Kubernetes

SKAFFOLD

Build Docker images with multiple backends:

- Docker build
- Kaniko
- Google Cloud Build
- Jib (Maven/Gradle)

Generates Dockerfile and Helm charts for your project

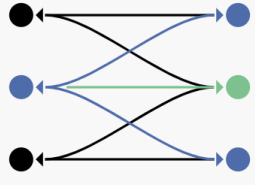
PROGRESSIVE DELIVERY

WITH JENKINS X

jenkins-x.io/developing/progressive-delivery

Istio

Connect, secure, control, and observe services.

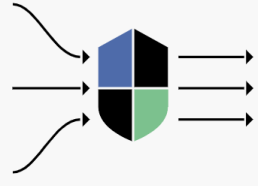


Connect

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

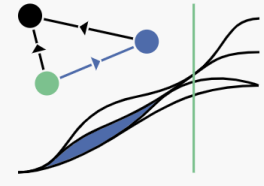
Secure

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

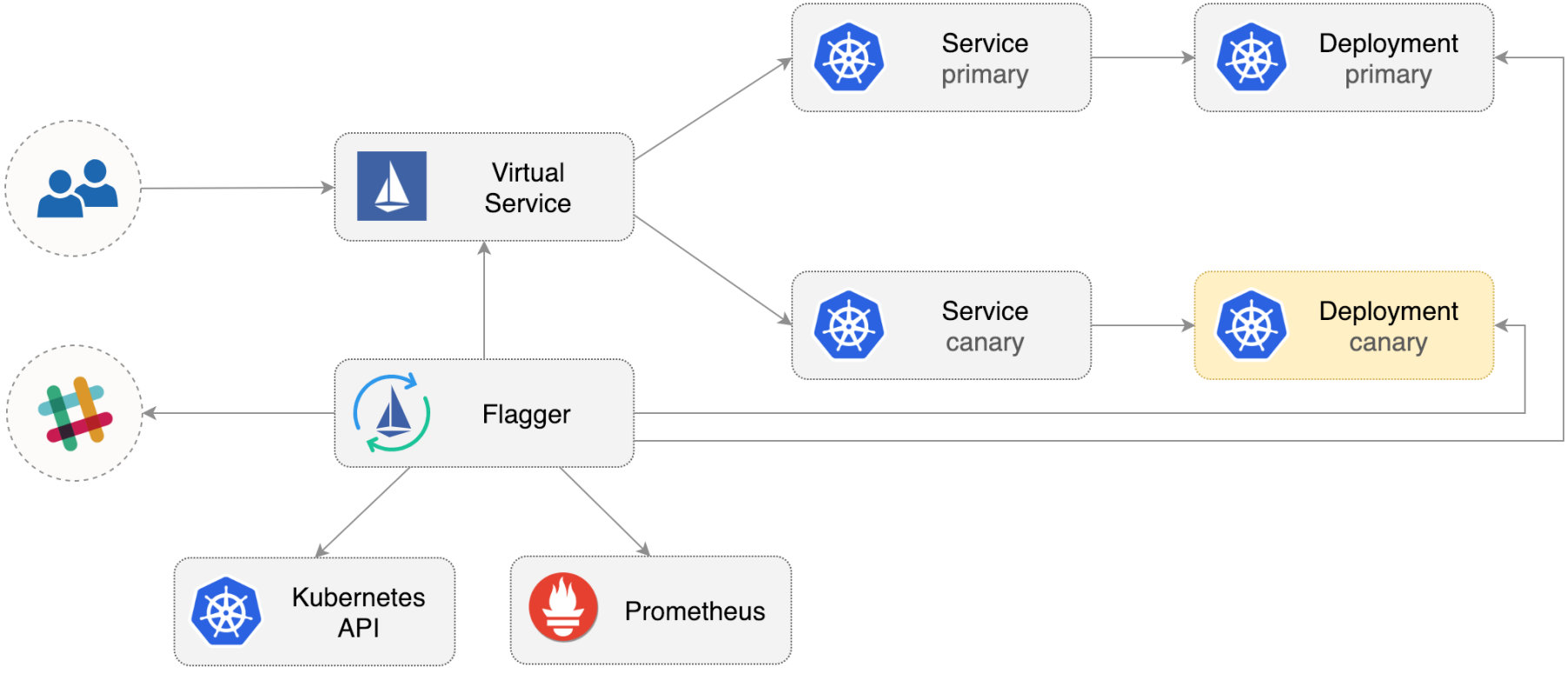
Add the `canary` section to our application Helm chart `values.yaml`

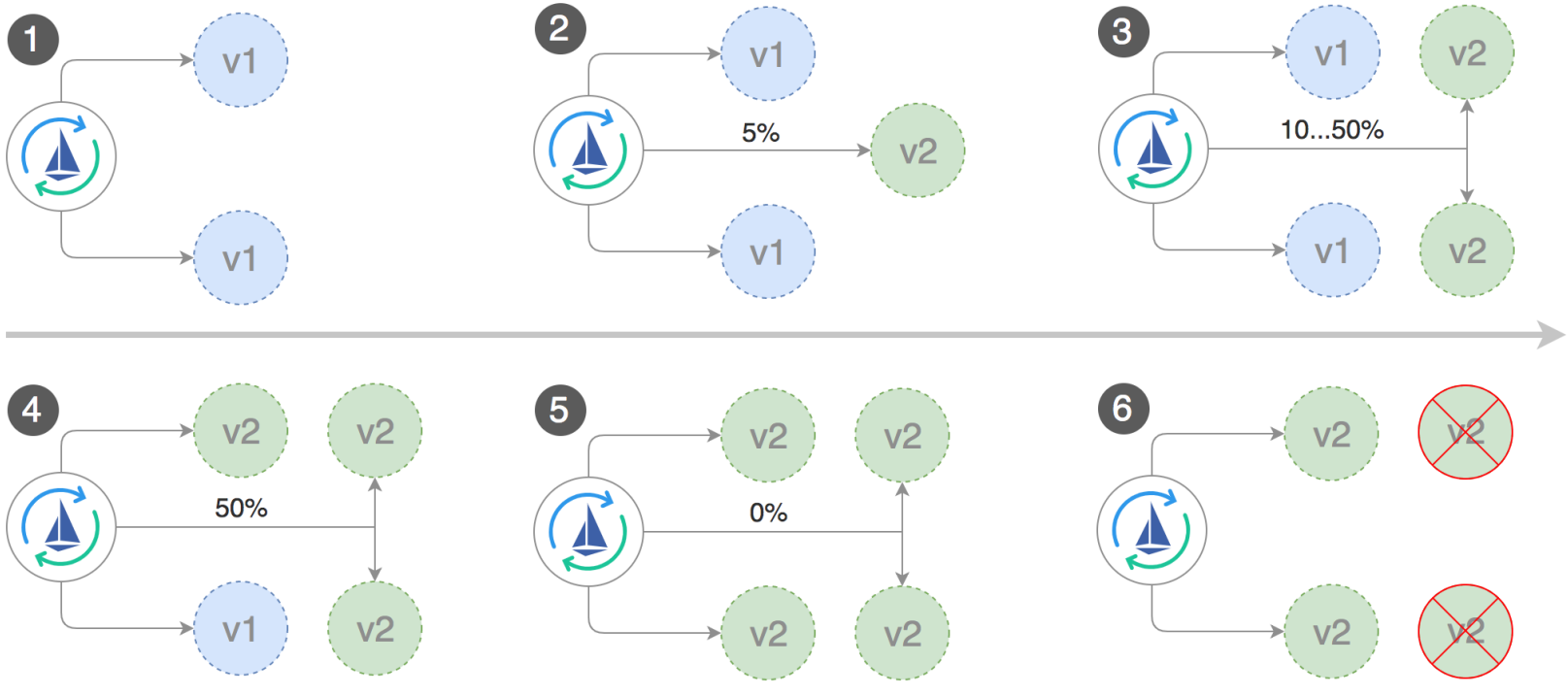
```yaml
...
canary:
  enable: true
  service:
    hosts:
    - croc-hunter.istio.us.g.csanchez.org
    gateways:
    - jx-gateway.istio-system.svc.cluster.local
  canaryAnalysis:
    interval: 60s
    threshold: 5
    maxWeight: 50
    stepWeight: 10
```

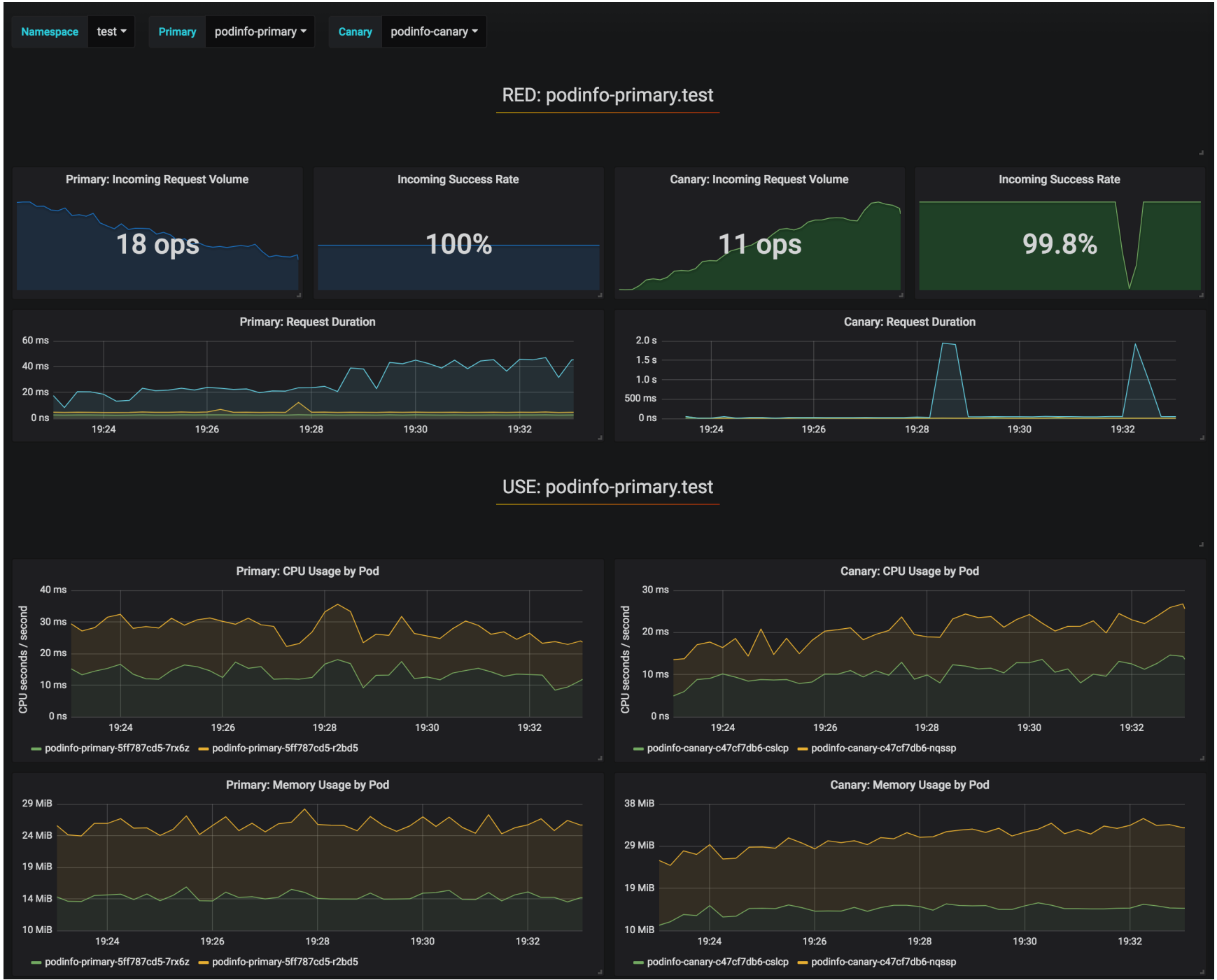
```yaml
    metrics:
    - name: request-success-rate
      # minimum req success rate (non 5xx responses)
      # percentage (0-100)
      threshold: 99
      interval: 60s
    - name: request-duration
      # maximum req duration P99
      # milliseconds
      threshold: 500
      interval: 60s
```

PROFIT!

```
jx promote croc-hunter-java \
  --version 0.0.130 \
  --env production
```


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


1
v1
v1
2
v1
5%
v2
v1
3
v1
v2
10...50%
v1
v2
4
v2
v2
50%
v1
v2
5
v2
v2
0%
v2
v2
6
v2
v2
v2
v2


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

carlossg/croc-hunter-jenkinsx

carlossg/croc-hunter-java

QUARKUS

quarkus.io

A Kubernetes Native Java stack tailored for GraalVM & OpenJDK HotSpot, crafted from the best of breed Java libraries and standards

csanchez.org


csanchez


carlossg