# CONTINUOUS DELIVERY PIPELINE

# Rodrigo Botti

Backend, Frontend, Ops, Fullstack (?)

**NEXA**: Integrations Team - Crawler, APIs, Ops, Fullstack (?)

**GDG Campinas**: Organizer

nexa

Blind Guardian
THE WITCHER
WILD HUNT

Ibanez

GOD OF WAR

IRON MAIDEN

# AGENDA

- Current Scenario
- Continuous Delivery
- Canary Release
- Tooling
- Future
- Q&A

# !!! DISCLAIMER !!!

# 1.

# SCENARIO

Technology
Platforms

# MAIN TECHNOLOGIES

kubernetes

# MAIN
# TECHNOLOGIES

Jenkins

wercker

# MICROSERVICES
## Advantages

### MODULARITY

Reduced codebase, easier to reason about, develop, test and deploy.

Single responsibility.

### PARALLELISM

Multiple teams working on separate services.

### SCALABLE

Faster feature delivery.

Independent horizontal/vertical scaling and failure handling.

# MICROSERVICES
## Requirements

### *** AUTOMATION ***

Automation is key for having speed, scalability / elasticity specially when dealing with a distributed services application.

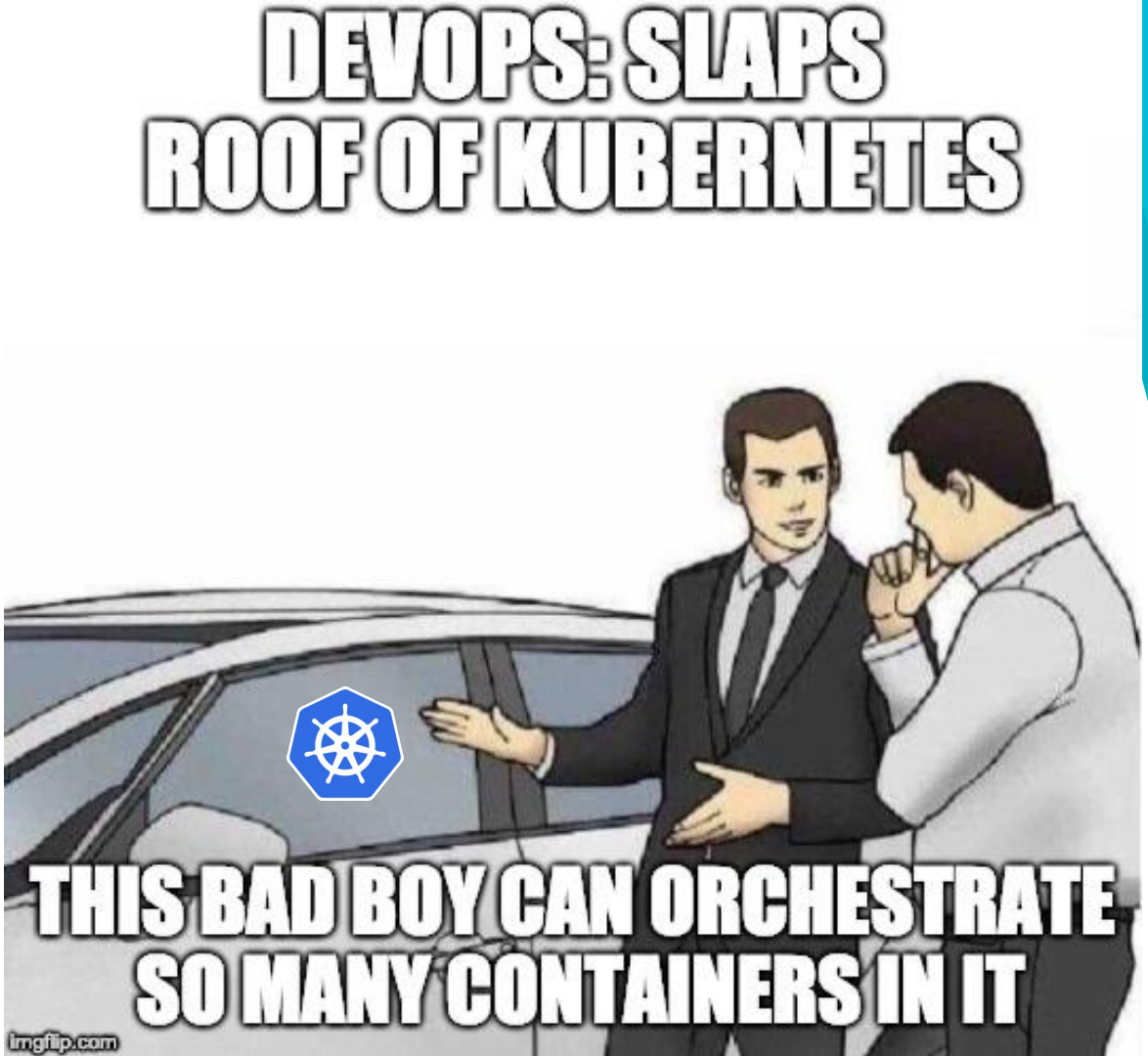
### DISCOVERY

Service discovery by name: cloud-level DNS, service-mesh level DNS, registration, client-side load balancing.

### OBSERVABILITY

Monitoring, metrics/health gathering, central tracing, APM, alarm management.

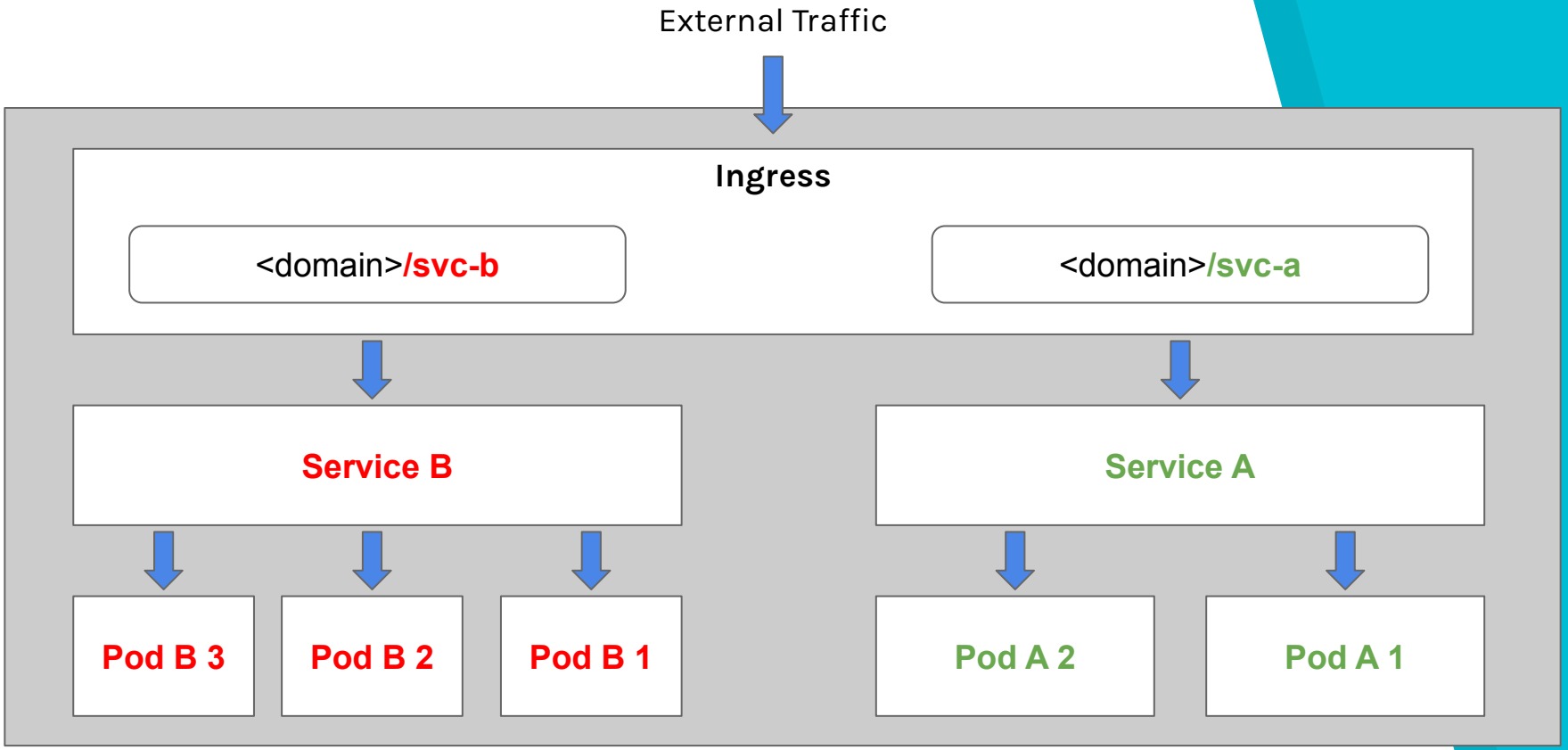
Avoid "needle in a haystack" debugging.


DEVOPS: SLAPS
ROOF OF KUBERNETES
THIS BAD BOY CAN ORCHESTRATE
SO MANY CONTAINERS IN IT
imgflip.com

# MICROSERVICES
## Anatomy (Kubernetes)

- **Deployment**
  - Pod: Container (Image) + Resource limits
  - Replica set: count + deploy strategy
- **Service**
  - Deployment load balancer
  - Ingress traffic
- **HorizontalPodAutoScaler**
  - Deployment replica set horizontal scaling
- **ServiceMonitor**
  - Prometheus metrics exporter
- **Ingress** (*)
  - Cluster gateway L7 router

# MICROSERVICES
## Anatomy (Kubernetes)



## SHOW ME THE CODE!

kubernetes

# Deployment

```
# Deployment      You, 18/01/2019 14:54 • Feature: k8s updated manifests
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: '${ trigger.properties["application"] }'
  namespace: production
  labels:
    app: '${ trigger.properties["application"] }'
spec:
  replicas: 1
  revisionHistoryLimit: 2
  selector:
    matchLabels:
      app: '${ trigger.properties["application"] }'
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: '${ trigger.properties["application"] }'
    spec:
      containers:
        - name: '${ trigger.properties["application"] }'
          image: '${ trigger.properties["imageUrl"] }'
          env:
            - name: NODE_ENV
              value: production
            - name: API_PORT
              value: '3000'
            - name: MONGODB_URI
              value: '${ trigger.properties["environment"]["production"]["MONGODB_URI"] }'
          livenessProbe:
            httpGet:
              path: /health
              port: 3000
            initialDelaySeconds: 30
            periodSeconds: 30
            successThreshold: 1
          ports:
            - containerPort: 3000
          resources:
            limits:
              cpu: 500m
```

# Service

```yaml
# Service
apiVersion: v1
kind: Service
metadata:
  name: '${ trigger.properties["application"] }'
  namespace: production
  labels:
    app: '${ trigger.properties["application"] }'
spec:
  ports:
    - name: http
      port: 80
      protocol: TCP
      targetPort: 3000
  selector:
    app: '${ trigger.properties["application"] }'
  type: ClusterIP
```

# HPA

```yaml
# HorizontalPodAutoscaler
apiVersion: autoscaling/v2beta1
kind: HorizontalPodAutoscaler
metadata:
  name: '${ trigger.properties["application"] }'
  namespace: production
spec:
  minReplicas: 2
  maxReplicas: 10
  scaleTargetRef:
    apiVersion: apps/v1beta2
    kind: Deployment
    name: '${ trigger.properties["application"] }'
  metrics:
    - resource:
        name: cpu
        targetAverageUtilization: 65
      type: Resource
    - resource:
        name: memory
        targetAverageUtilization: 65
      type: Resource
```

# Ingress

```yaml
# Ingress
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: '${ trigger.properties["application"] }'
  namespace: production
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
  labels:
    app: '${ trigger.properties["application"] }'
spec:
  rules:
  - host: '${ trigger.properties["gatewayUrl"] }'
    http:
      paths:
      - path: '${ trigger.properties["servicePath"] }'
        backend:
          serviceName: '${ trigger.properties["application"] }'
          servicePort: 80
```

# SVC Monitor

```yaml
# ServiceMonitor
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: '${ trigger.properties["application"] }'
  namespace: monitoring
  labels:
    app: '${ trigger.properties["application"] }'
    release: prometheus
spec:
  selector:
    matchLabels:
      app: '${ trigger.properties["application"] }'
  endpoints:
  - port: http
    path: /metrics
  namespaceSelector:
    matchNames:
      - production
```

# 2.

# CONTINUOUS DELIVERY

Concept
Pipeline

“

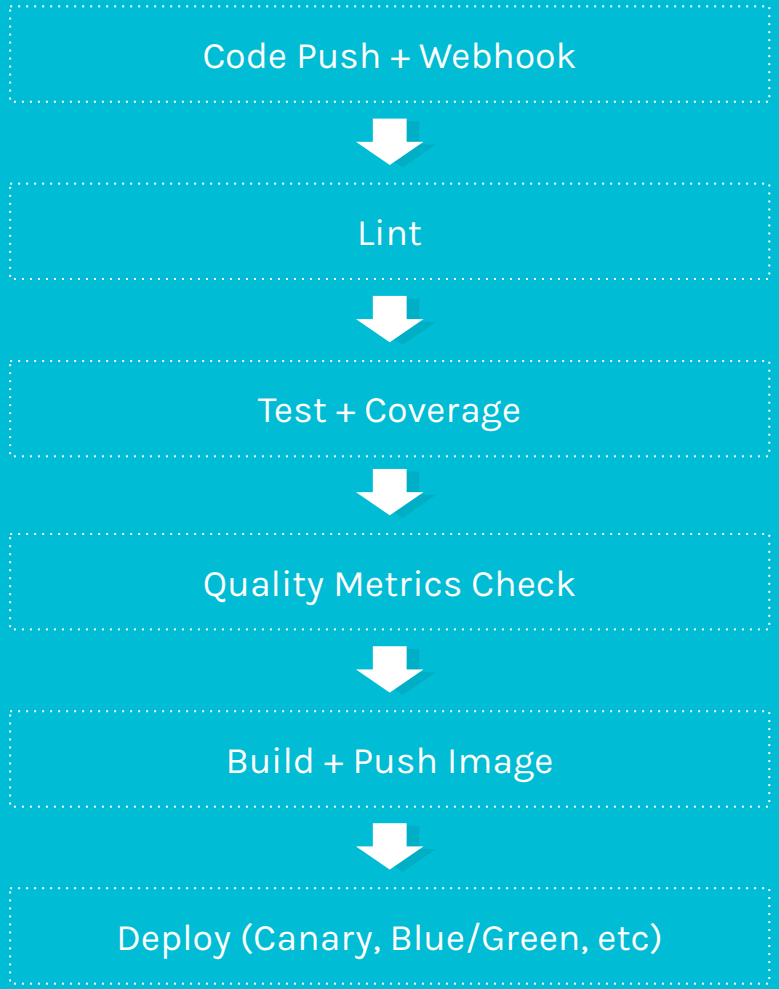
Produce software in short cycles

Reliably released at any time

Building, testing and releasing faster and more frequently

Straightforward and repeatable deployment process

# PIPELINE


Code Push + Webhook
Lint
Test + Coverage
Quality Metrics Check
Build + Push Image
Deploy (Canary, Blue/Green, etc)

# 3.

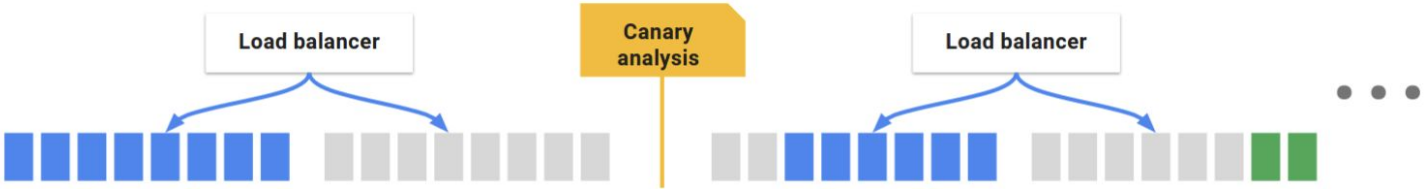
# CANARY RELEASE

Concept
Pipeline

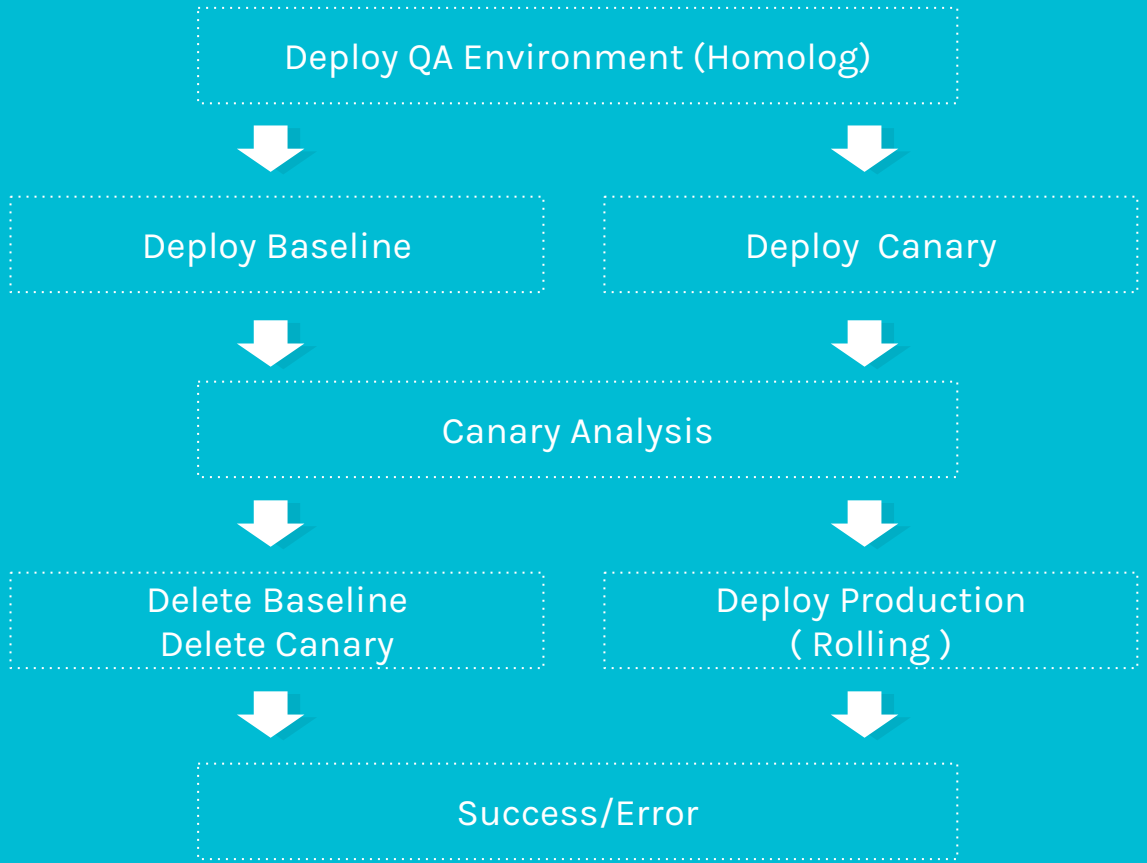
Red/black
(Blue/green)

Rolling red/black

Canary



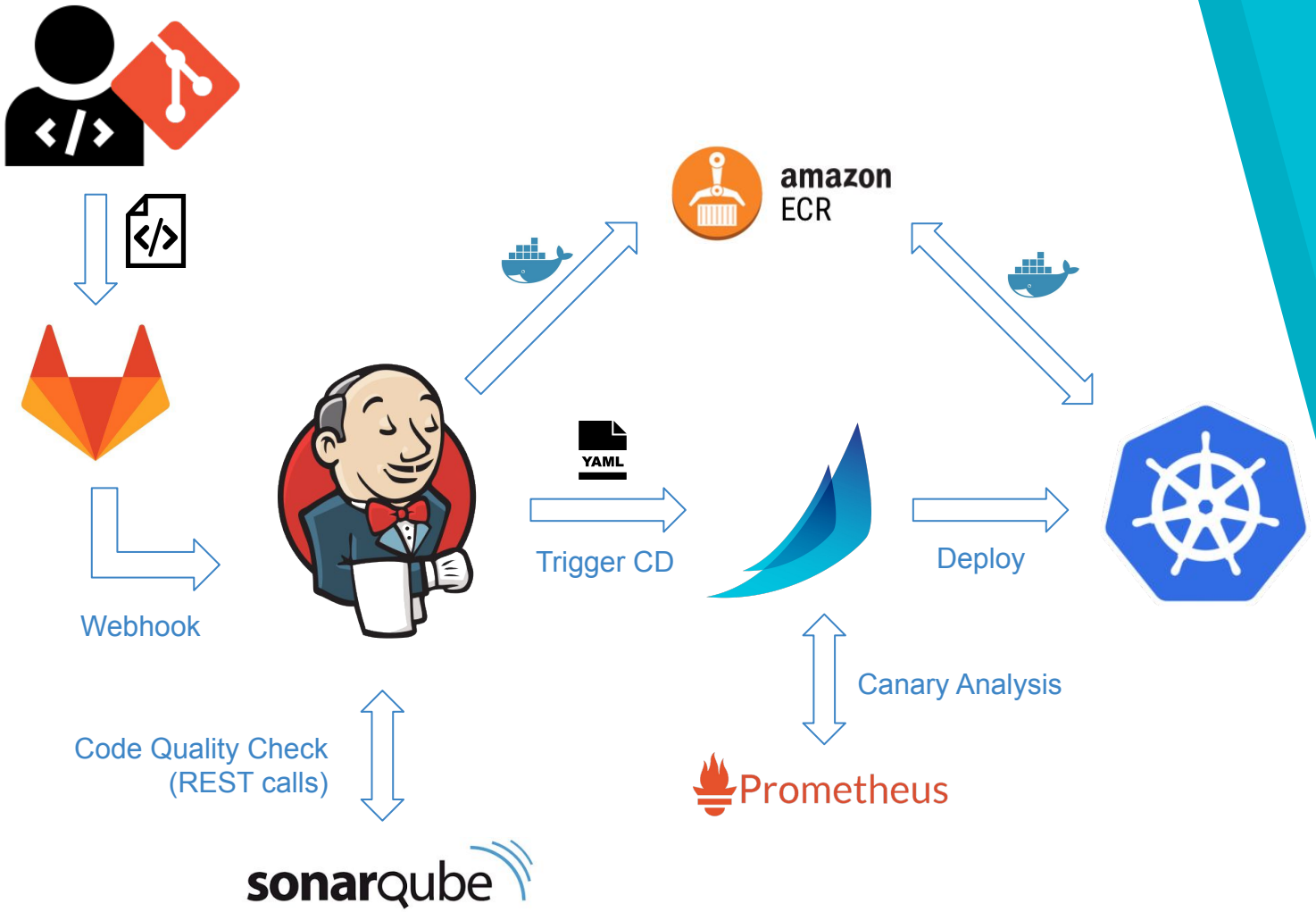
# PIPELINE


Deploy QA Environment (Homolog)
Deploy Baseline
Deploy Canary
Canary Analysis
Delete Baseline
Delete Canary
Deploy Production
( Rolling )
Success/Error

# 4.

# TOOLS

Topology
Automation Server
Integration Tests
Code Quality
Deploy
Canary


amazon
ECR
YAML
Webhook
Trigger CD
Deploy
Canary Analysis
Code Quality Check
(REST calls)
Prometheus
sonarqube

# (LEEEROOOOOOOOOOY...)
# JENKINS

- Automation Server
- Pipeline
- Groovy DSL
- Plugins

# (LEEEROOOOOOOOOOY...)
# JENKINS

- Kubernetes Plugin
- Custom Library
- Opinionated DSL

# tools-featureflag-api

Branches (4)

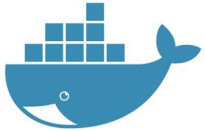
| S | W | Name ↓ | Last Success | Last Failure | Last Duration | Fav |
|---|---|---|---|---|---|---|
| | | adjust-import-scripts | 1 mo 5 days - #2 | N/A | 1 min 30 sec | |
| | | feature/livia-environment | 11 hr - #1 | N/A | 1 min 8 sec | |
| | | **master** | 1 day 14 hr - #23 | N/A | 2 min 38 sec | |
| | | postman-api-doc | N/A | N/A | N/A | |

## Stage View

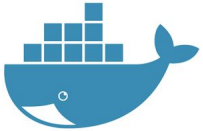
| | Checkout | Install Dependencies | Lint | Tests | Check quality: Scan | Check quality: Quality Gate | Build and publish image | Deploy properties |
|---|---|---|---|---|---|---|---|---|
| Average stage times: (Average full run time: ~2min 32s) | 2s | 22s | 4s | 7s | 21s | 6s | 57s | 6s |
| #23 Jan 16 11:58 — 1 commit | 2s | 21s | 5s | 7s | 22s | 6s | 1min 0s | 7s |
| #22 Jan 15 20:18 — No Changes | 2s | 26s | 4s | 6s | 19s | 4s | 51s | 6s |
| #21 Jan 15 18:33 — No Changes | 2s | 21s | 5s | 7s | 20s | 5s | 50s | 7s |

App Test Pod

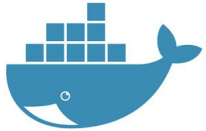
App Container



DB Container



Cache Container



# SHOW ME THE CODE!

# Pipeline

```groovy
@Library('livia-tools-jenkins-pipeline-lib@master') _

def serviceName = "tools-featureflag-api"
def servicePath = "/featureflag"
def registryRepository = "${serviceName}-prd"
def containers = [
  containerTemplate(
    name: 'mongo',
    image: 'mongo:latest',
    ttyEnabled: true,
    ports: [portMapping(name: 'mongodb', containerPort: 27017, hostPort: 27017)]
  )
]
def testEnv = [MONGODB_URI: "mongodb://localhost:27017/${serviceName}"]
def homologEnv = [
  NODE_ENV: 'production',
  MONGODB_URI: 'mongodb://****/featureflag'
]
def productionEnv = [
  NODE_ENV: 'production',
  MONGODB_URI: 'mongodb://****/featureflag?replicaSet=****'
]

nodeJSApi(
  name: serviceName,
  servicePath: servicePath,
  branch: env.BRANCH_NAME,
  nodeImage: 'node:dubnium',
  testEnv: testEnv,
  homologEnv: homologEnv,
  productionEnv: productionEnv,
  containers: containers,
  registryRepository: registryRepository,
  canaryEnabled: true
)
```

DSL

```groovy
def call(Map args) {
  // -- arguments --
  def name = args.name
  def branch = args.branch
  def servicePath = args.servicePath ?: args.name

  def nodeImage = args.nodeImage ?: 'node:dubnium'

  def testEnv = args.testEnv ?: [:]
  def homologEnv = args.homologEnv ?: [:]
  def productionEnv = args.productionEnv ?: [:]

  def containers = args.containers ?: []
  def volumes = args.volumes ?: []

  def registryRepository = args.registryRepository ?: name
  def canaryEnabled = args.canaryEnabled ?: false
  def masterBranch = branch == 'master'

  // -- global constants --
  def DEFAULT_CONTAINERS = [
    containerTemplate(name: 'node', image: nodeImage, ttyEnabled: true),
    containerTemplate(name: 'docker', image: 'docker', ttyEnabled: true),
    containerTemplate(name: 'kubectl', image: 'lachlanevenson/k8s-kubectl:v1.10.5', command: 'cat', ttyEnabled: true)
  ]
  def DEFAULT_VOLUMES = [hostPathVolume(hostPath: '/var/run/docker.sock', mountPath: '/var/run/docker.sock')]
  def REGISTRY_CONFIG = [
    host: '****.amazonaws.com',
    url: "https://****.amazonaws.com",
    credentials: "ecr:****"
  ]
```

# DSL

```groovy
// -- execution --
def label = "job-${UUID.randomUUID().toString()}"
def podContainers = DEFAULT_CONTAINERS + containers
def podVolumes = DEFAULT_VOLUMES + volumes

podTemplate(label: label, containers: podContainers, volumes: podVolumes, serviceAccount: 'jenkins') {
  node(label) {
    notifyStatus {
      // ** CLONE REPO **
      stage('Checkout') {
        checkout scm
      }

      container('node') {
        // ** INSTALL DEPENDENCIES **
        stage('Install Dependencies') {
          sh 'npm install'
        }
        // ** LINT CHECK **
        stage('Lint') {
          sh 'npm run lint'
        }
        // ** UNIT AND INTEGRATION TESTS **
        stage('Tests') {
          def testEnvList = testEnv.collect({ key, value -> "${key}=${value}" })
          withEnv(testEnvList) {
            sh 'npm run test:ci:cover'
          }
        }
```

# DSL

```
    if (masterBranch) {
      // ** SONAR SCANNER **
      stage('Check quality: Scan') {
        sh 'npm run sonar:scanner'
      }
    }
  }

  if (masterBranch) {
    // ** CODE QUALITY CHECKS **
    stage('Check quality: Quality Gate') {
      def versionFolders = listSubFolders "api/v*"
      def qualityGateStatus = getQualityGateStatus versionFolders
      if (!qualityGateStatus) {
        error "Failed because at least one version did not pass the Quality Gate"
      } else {
        echo "Passed Quality Gate check"
      }
    }

    // ** BUILD AND PUBLISH **
    def tag = getGitCommit()
    def imageName = "${registryRepository}:${tag}"
    container('docker') {
      stage('Build and publish image') {
        docker.withRegistry(REGISTRY_CONFIG.url, REGISTRY_CONFIG.credentials) {
          docker
            .build(imageName)
            .push()
        }
      }
    }
```

# DSL

```
stage("Deploy properties") {
  def imageUrl = "${REGISTRY_CONFIG.host}/${imageName}"

  def deployData = [
    application: name,
    servicePath: servicePath,
    imageUrl: imageUrl,
    environment: [
      homolog: homologEnv,
      production: productionEnv,
    ]
  ]

  if (canaryEnabled) {
    echo 'Canary enabled: Building canary config'
    container('kubectl') {
      def currentProductionImageUrl = getRunningPodImage app: name, namespace: 'production'
      def currentProductionState = getRunningPodState app: name, namespace: 'production'
      deployData.put 'canary', [
        baselineState: currentProductionState,
        baselineImageUrl: currentProductionImageUrl,
        canaryImageUrl: imageUrl
      ]
    }
  }

  archiveDeployTriggerYaml name: 'deploy.yml', data: deployData
}
```

# Running Pipeline Pod

```
Rodrigos-MacBook-Pro-2:~ rodrigobotti$ kubectl get pods -n jenkins -w
NAME                       READY     STATUS    RESTARTS   AGE
jenkins-7669df5fd8-wvzpl   1/1       Running   0          29d
job-cedb6d14-1c3f-4caf-8d82-b009d5443f82-krnjg-fcmm3   0/5       Pending   0         1s
job-cedb6d14-1c3f-4caf-8d82-b009d5443f82-krnjg-fcmm3   0/5       Pending   0         1s
job-cedb6d14-1c3f-4caf-8d82-b009d5443f82-krnjg-fcmm3   0/5       ContainerCreating   0         1s
job-cedb6d14-1c3f-4caf-8d82-b009d5443f82-krnjg-fcmm3   5/5       Running   0         3s
```

# QUALITY CHECKS
# SONARQUBE

- Quality Metrics
  - Code smells
  - Bugs
  - Vulnerabilities
  - Code coverage
- Quality Profiles
- Quality Gate

Microservices

Rename | Copy | Unset as Default | Delete

## Conditions

Only project measures are checked against thresholds. Sub-projects, directories and files are ignored. More

| Metric | Over Leak Period | Operator | Warning | Error | |
|---|---|---|---|---|---|
| Blocker Issues | ☐ | is greater than | | 0 | Update Delete |
| Bugs | ☐ | is greater than | | 0 | Update Delete |
| Code Smells | ☐ | is greater than | | 0 | Update Delete |
| Confirmed Issues | ☐ | is less than | | 0 | Update Delete |
| Coverage | ☐ | is less than | 95 | 92 | Update Delete |
| Critical Issues | ☐ | is less than | | 0 | Update Delete |
| Duplicated Blocks | ☐ | is greater than | | 0 | Update Delete |
| Maintainability Rating | Never | is worse than | | A × | Update Delete |
| Major Issues | ☐ | is greater than | | 0 | Update Delete |
| Minor Issues | ☐ | is less than | | 0 | Update Delete |
| Open Issues | ☐ | is less than | | 0 | Update Delete |
| Reliability Rating | Never | is worse than | | A × | Update Delete |
| Security Rating | Never | is worse than | | A × | Update Delete |
| Vulnerabilities | ☐ | is greater than | | 0 | Update Delete |

Add Condition

[Tools] [API] FeatureFlag - v1 master

January 16, 2019, 12:00 PM Version 1.0.0

Overview Issues Measures Code Activity Administration

Quality Gate Passed

Bugs Vulnerabilities

| 0 A | 0 A | Leak Period: since previous version started last month | |
|---|---|---|---|
| Bugs | Vulnerabilities | 0 A New Bugs | 0 A New Vulnerabilities |

Code Smells

| 0 A | 0 | 0 A | 0 |
|---|---|---|---|
| Debt started last month | Code Smells | New Debt | New Code Smells |

Coverage

| 98.9% | — |
|---|---|
| Coverage | Coverage on New Code |

Duplications

| 0.0% | 0 | — |
|---|---|---|
| Duplications | Duplicated Blocks | Duplications on New Code |

XS 980 Lines of Code

JavaScript 980

No tags

**Activity**

**January 16, 2019**
1.0.0

**January 15, 2019**
Project Analyzed

**December 21, 2018**
Project Analyzed

Show More

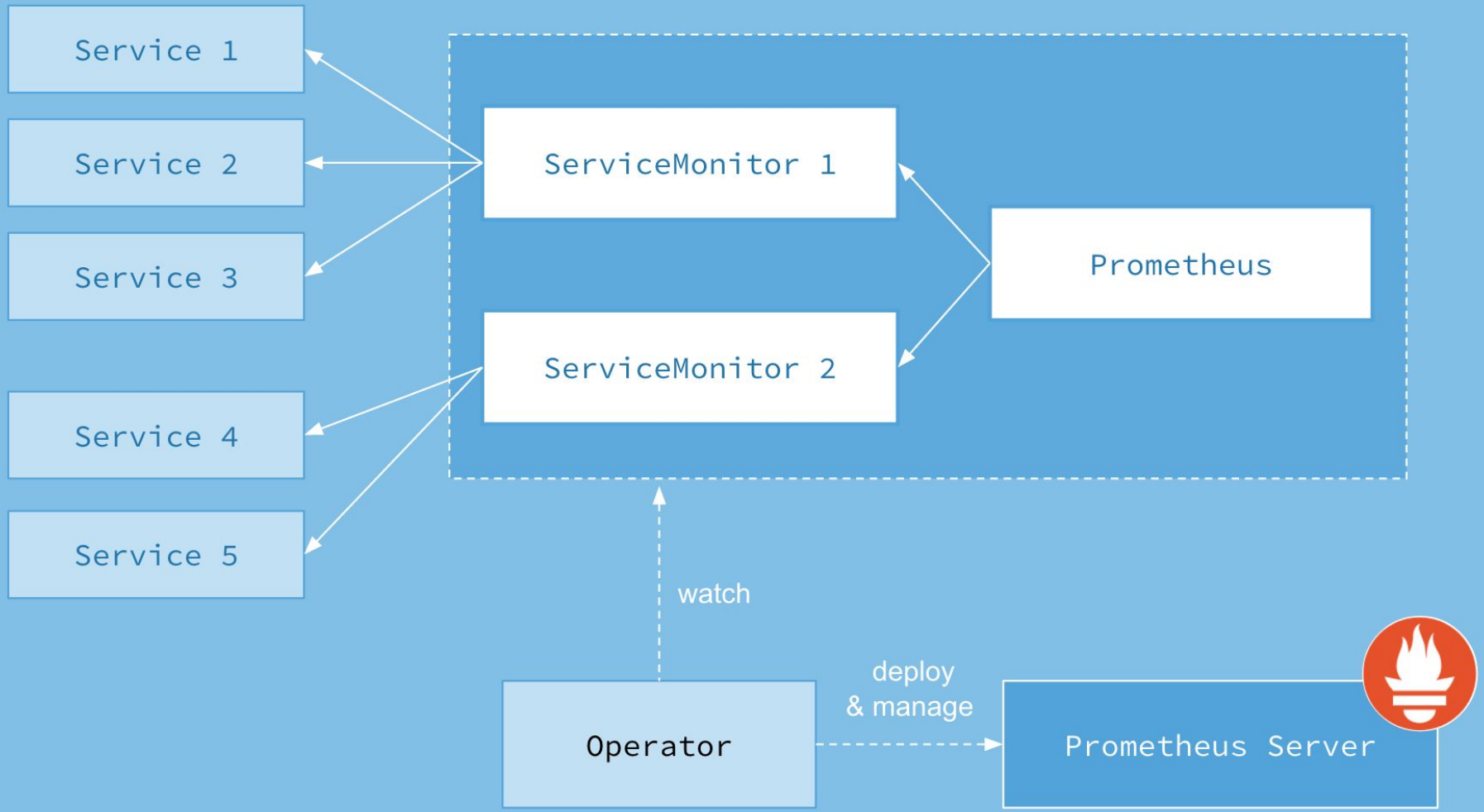
**Quality Gate**
(Default) Microservices

**Quality Profiles**
(JavaScript) Sonar way Recommended

**Key**
tools:featureflag:api:v1

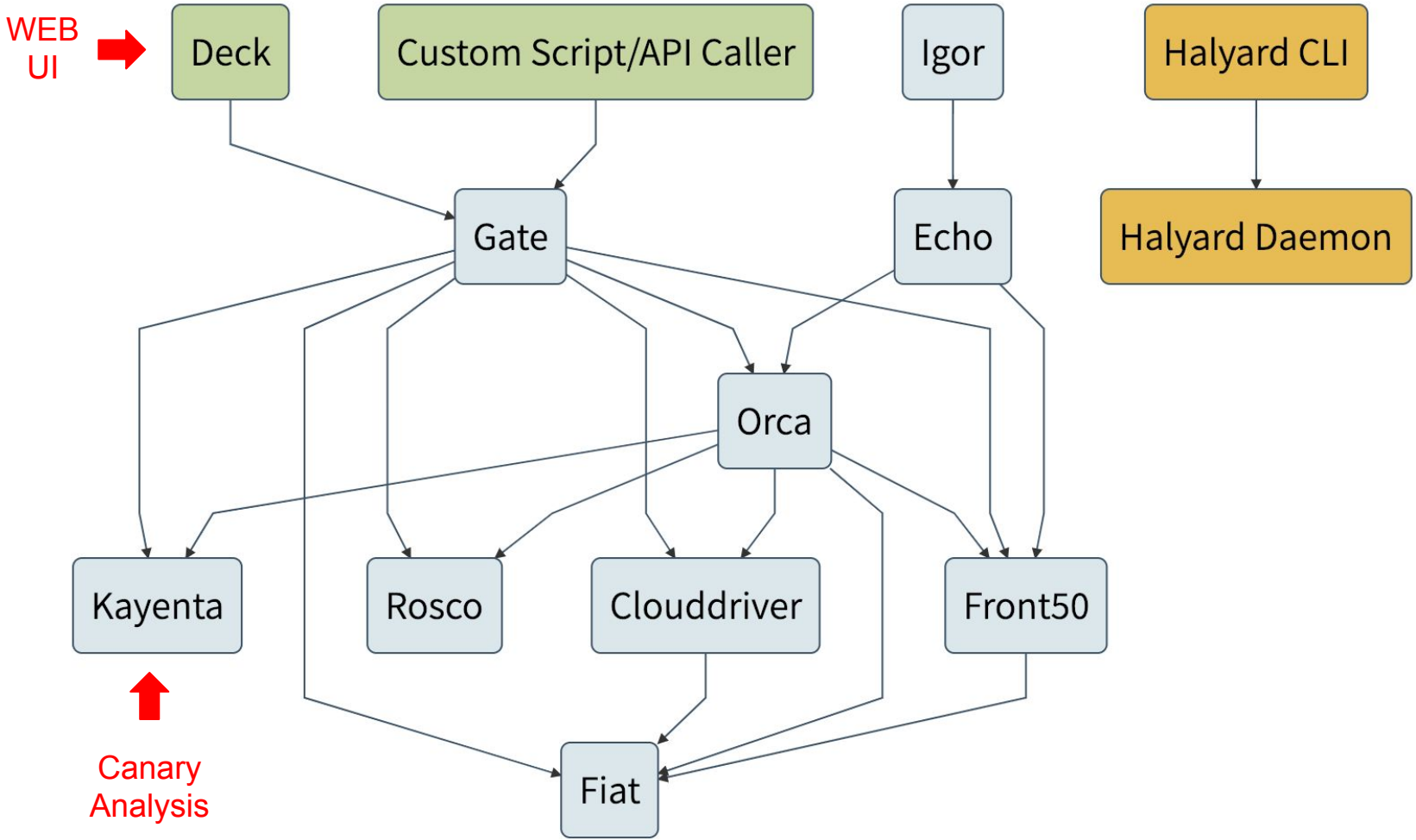
# MONITORING
# PROMETHEUS (OPERATOR)

- Monitoring
- Time-series database
- PromQL
- Alerting


Service 1
Service 2
Service 3
Service 4
Service 5
ServiceMonitor 1
ServiceMonitor 2
Prometheus
watch
Operator
deploy
& manage
Prometheus Server

# DELIVERY
# SPINNAKER

- CD Pipeline
- Canary Release
- Multi-cloud
  - Kubernetes
- Abstracts infrastructure elements (*)
- Netflix


WEB UI
Deck
Custom Script/API Caller
Igor
Halyard CLI
Gate
Echo
Halyard Daemon
Orca
Kayenta
Rosco
Clouddriver
Front50
Canary Analysis
Fiat

YO DAWG, HEARD YOU LIKE SERVICES
SO WE PUT SERVICES IN A SERVICE SO
YOU CAN DEPLOY SERVICES USING A SERVICE
imgflip.com

# CONCEPTS/ABSTRACTIONS
# SPINNAKER

- Application
- Cluster == **Namespace**
- Server Group == **Deployment**
- Load Balancer == **Service**

SPINNAKER-AWS | deployment tools-featureflag-api | 3 ▲ : 100%

## HOMOLOG

**V018**: 454706396284.dkr.ecr.sa-east-1.amazonaws.com/tools-featureflag-api-prd:644996926ea265c9a4f2ec140b7335c732f7de05

1 ▲ : 100%

| Instance | Launch Time | Zone | Provider |
|---|---|---|---|
| ▲pod tools-featureflag-api-fcc - 8957c4-h48nz | | homolog | Up |

**V017**: 454706396284.dkr.ecr.sa-east-1.amazonaws.com/tools-featureflag-api-prd:23b31adddcb2aefad2a1eba7a6dc2bd164400e16

## PRODUCTION

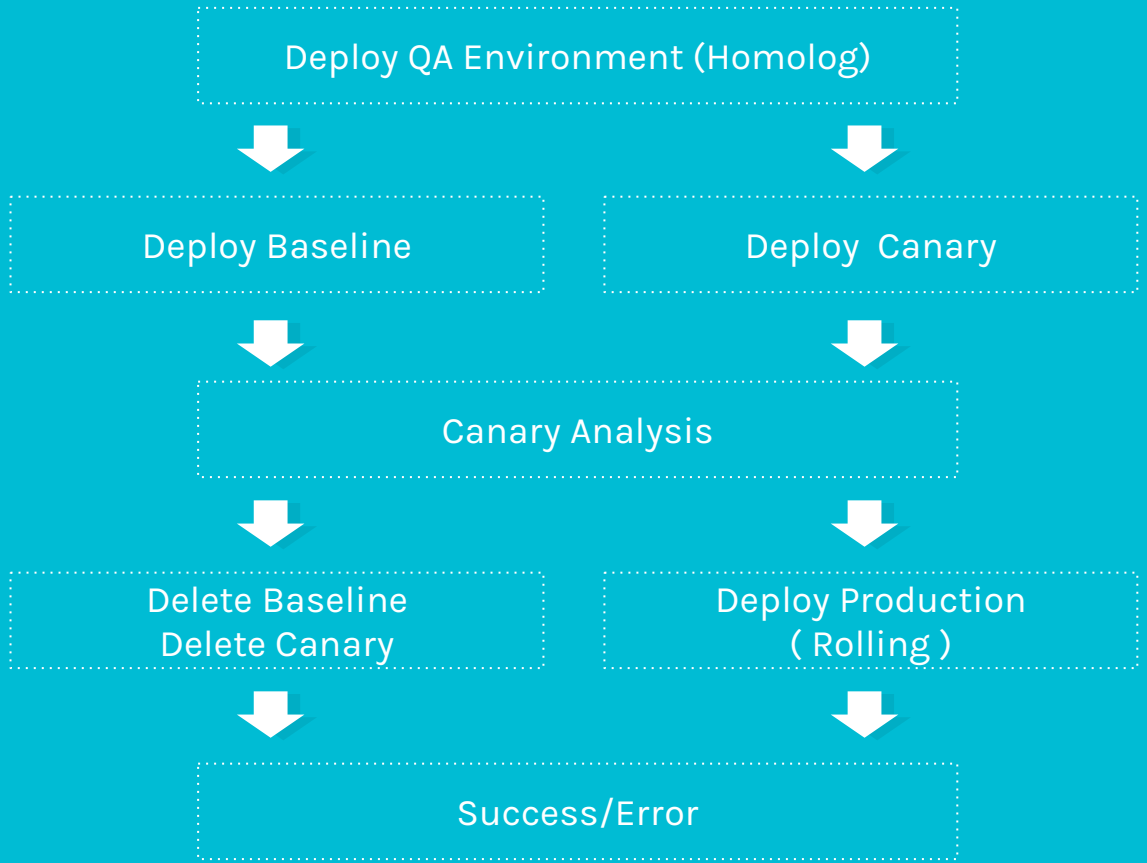
**V022**: 454706396284.dkr.ecr.sa-east-1.amazonaws.com/tools-featureflag-api-prd:23b31adddcb2aefad2a1eba7a6dc2bd164400e16

2 ▲ : 100%

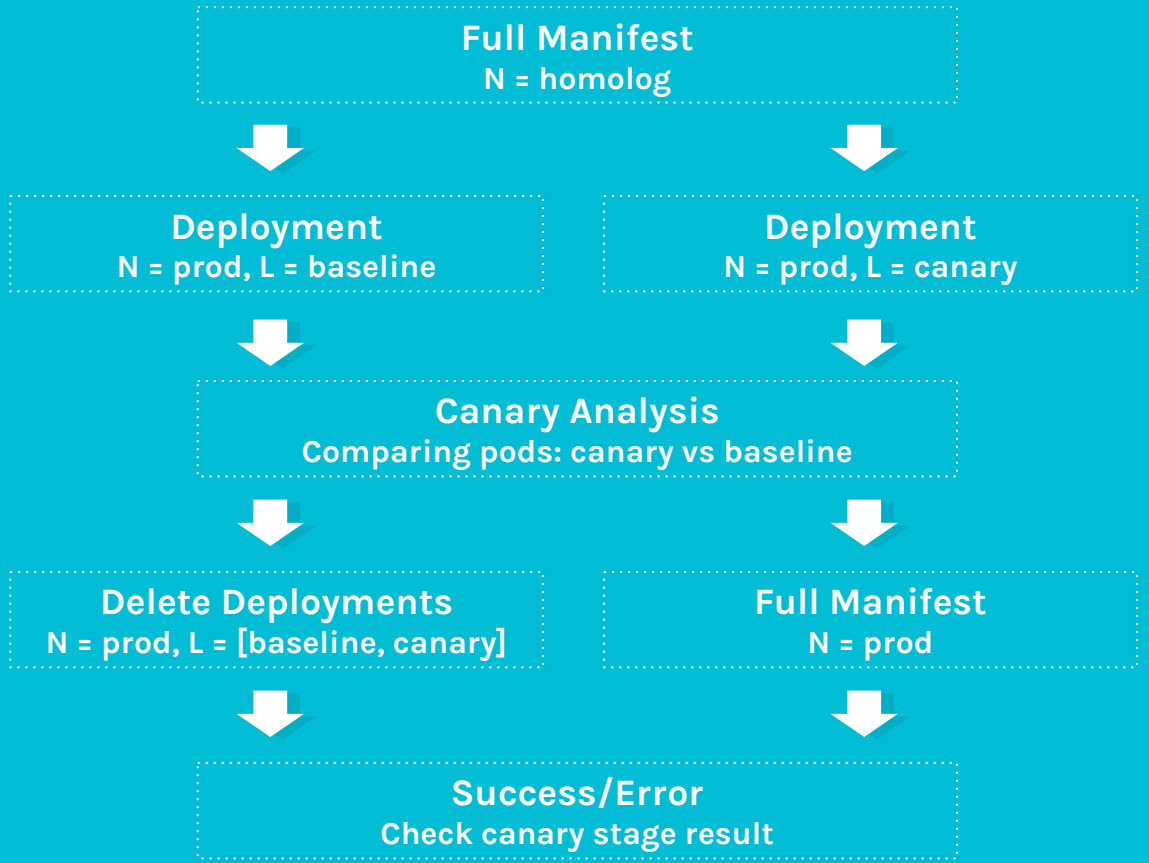
| Instance | Launch Time | Zone | Provider |
|---|---|---|---|
| ▲pod tools-featureflag-api-65 - 4f8c5669-f4zhn | | production | Up |
| ▲pod tools-featureflag-api-65 - 4f8c5669-pzdrg | | production | Up |

**V021**: 454706396284.dkr.ecr.sa-east-1.amazonaws.com/tools-featureflag-api-prd:d4f217e2623d1920564cc4c980e12eddb3e0635e

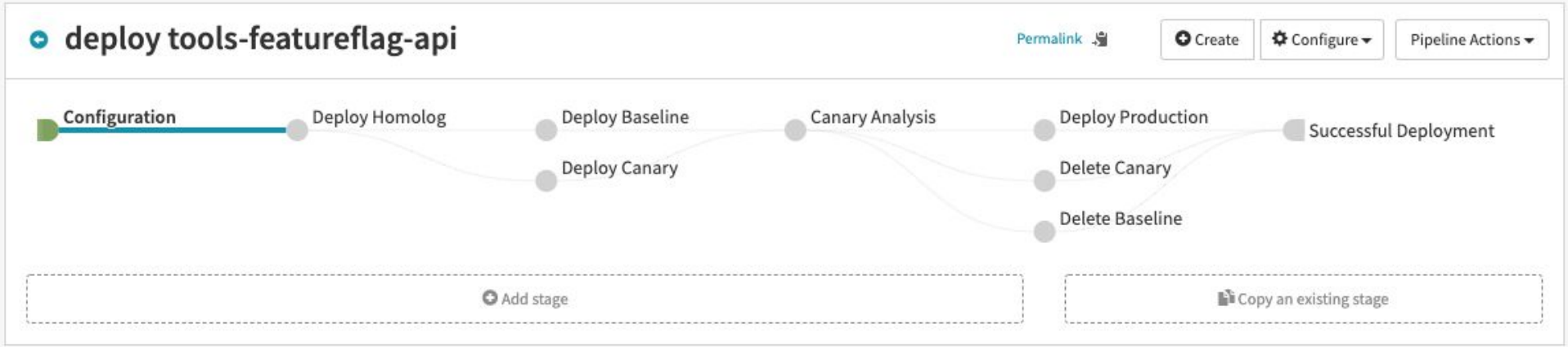
# PIPELINE


Deploy QA Environment (Homolog)
Deploy Baseline
Deploy Canary
Canary Analysis
Delete Baseline
Delete Canary
Deploy Production
( Rolling )
Success/Error

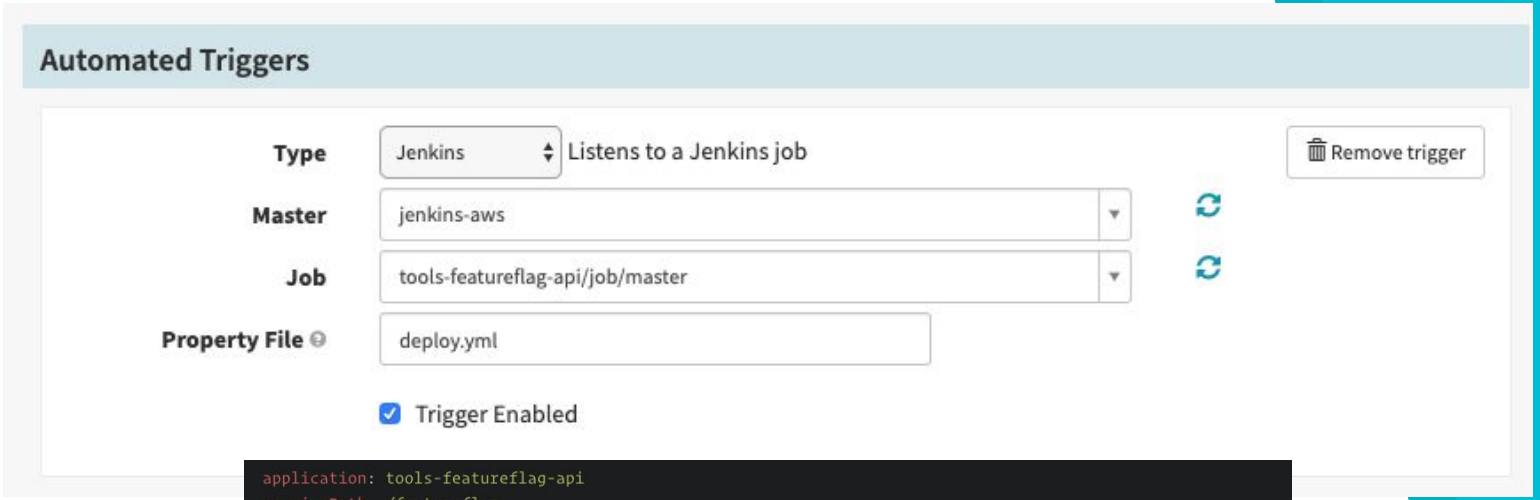
# PIPELINE


Full Manifest
N = homolog
Deployment
N = prod, L = baseline
Deployment
N = prod, L = canary
Canary Analysis
Comparing pods: canary vs baseline
Delete Deployments
N = prod, L = [baseline, canary]
Full Manifest
N = prod
Success/Error
Check canary stage result

# Pipeline Overview

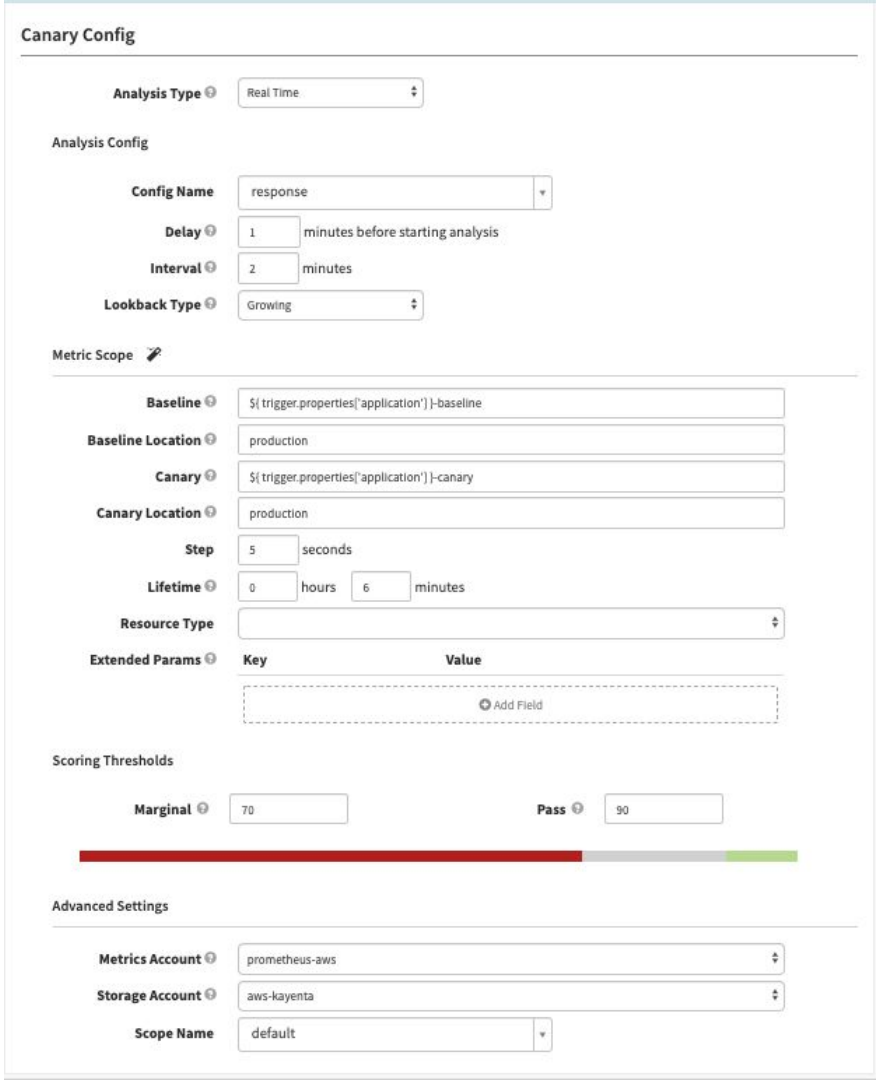


# Stage: Configuration

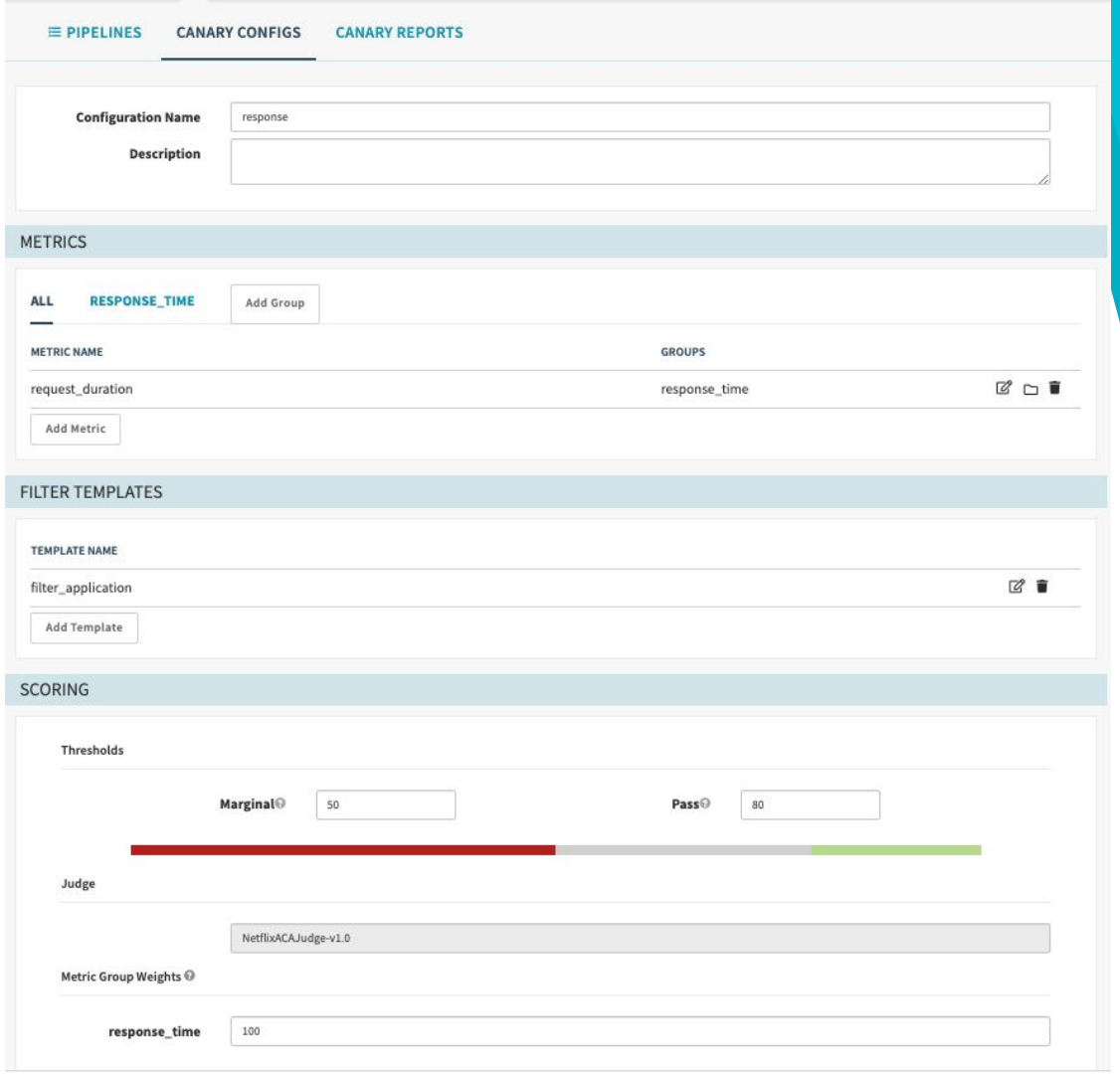



```
application: tools-featureflag-api
servicePath: /featureflag
imageUrl: '****.amazonaws.com/tools-featureflag-api-prd:644996926ea265c9a4f2ec140b7335c732f7de05'
environment:
  homolog:
    NODE_ENV: production
    MONGODB_URI: mongodb://****/featureflag
  production:
    NODE_ENV: production
    MONGODB_URI: mongodb://****/featureflag?replicaSet=****
canary:
  baselineState:
    imageUrl: '****.amazonaws.com/tools-featureflag-api-prd:23b31adddcb2aefad2a1eba7a6dc2bd164400e16'
    environment:
      NODE_ENV: production
      API_PORT: '3000'
      MONGODB_URI: mongodb://****/featureflag?replicaSet=****
  baselineImageUrl: '****.amazonaws.com/tools-featureflag-api-prd:23b31adddcb2aefad2a1eba7a6dc2bd164400e16'
  canaryImageUrl: '****.amazonaws.com/tools-featureflag-api-prd:644996926ea265c9a4f2ec140b7335c732f7de05'
```

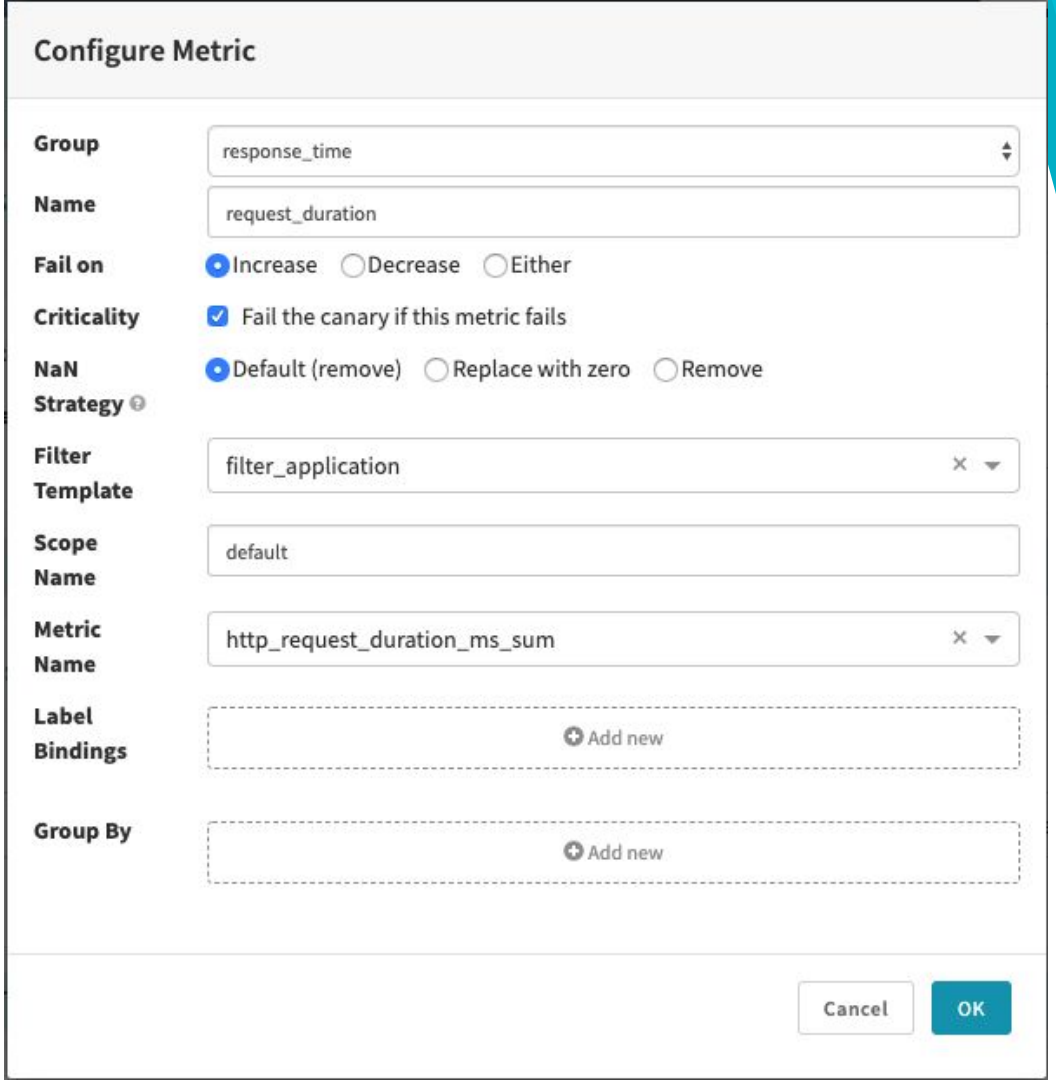
# Stage: Canary Analysis



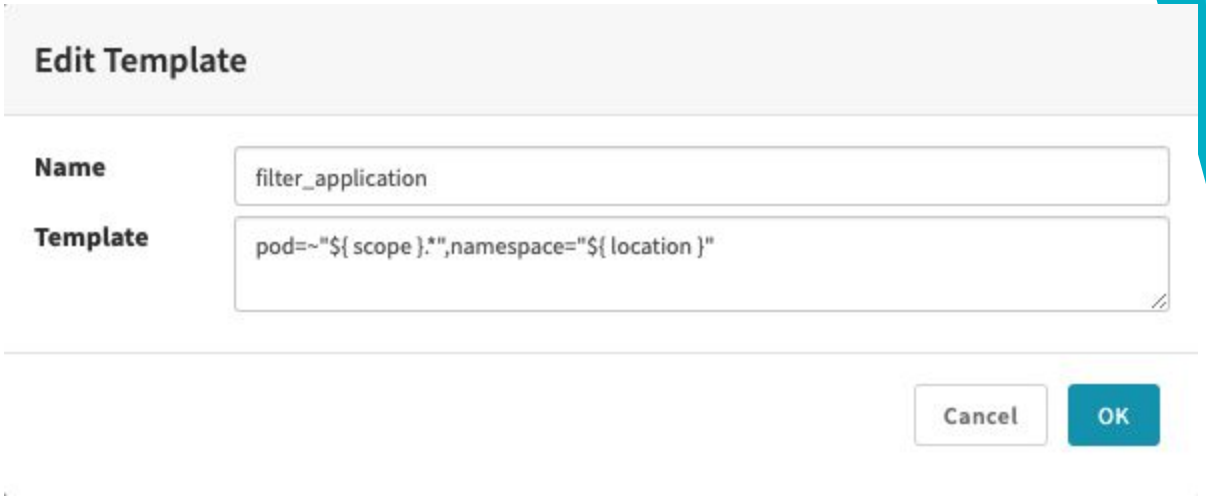
# Canary Config



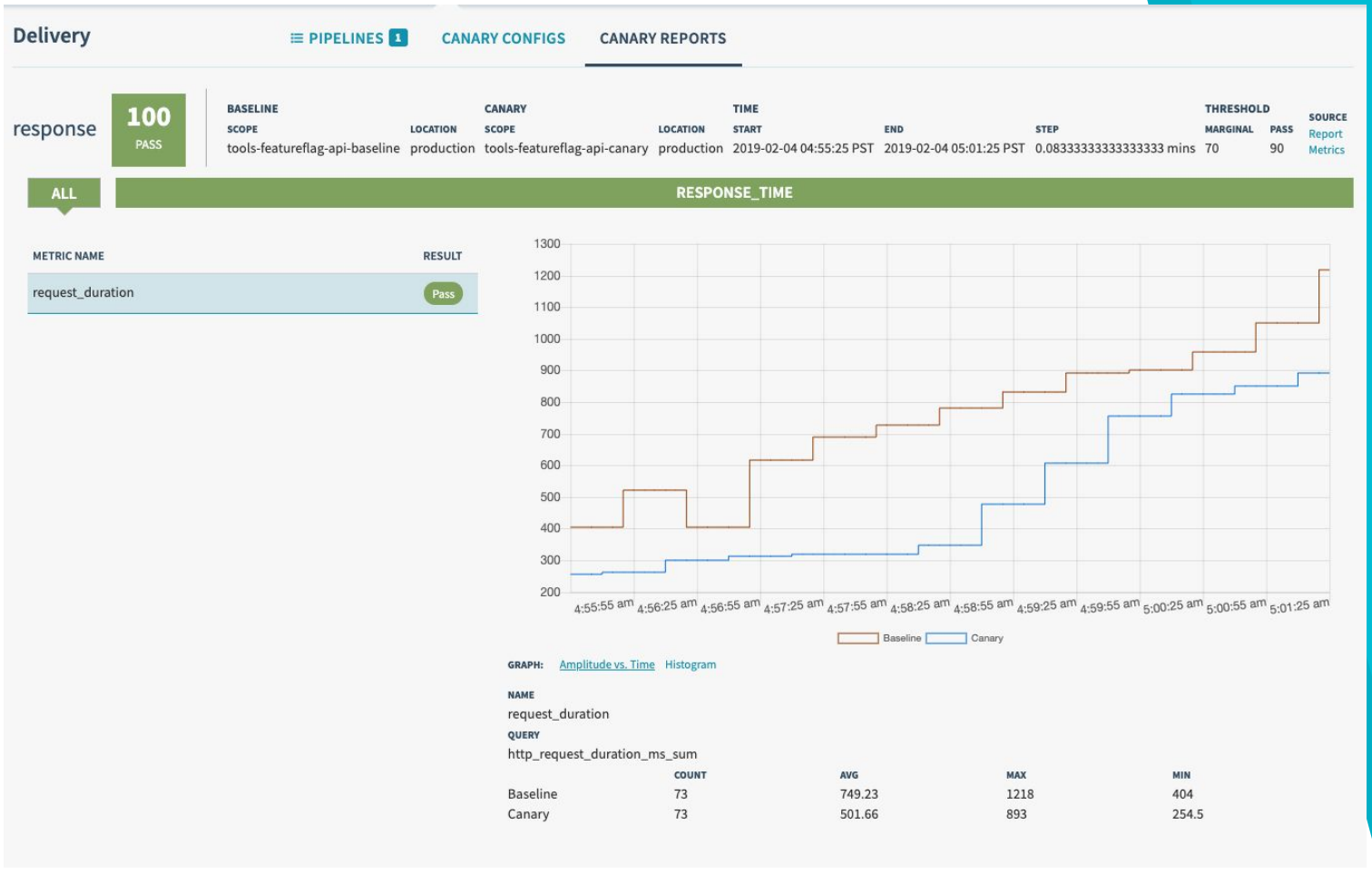
# Canary Config Metric



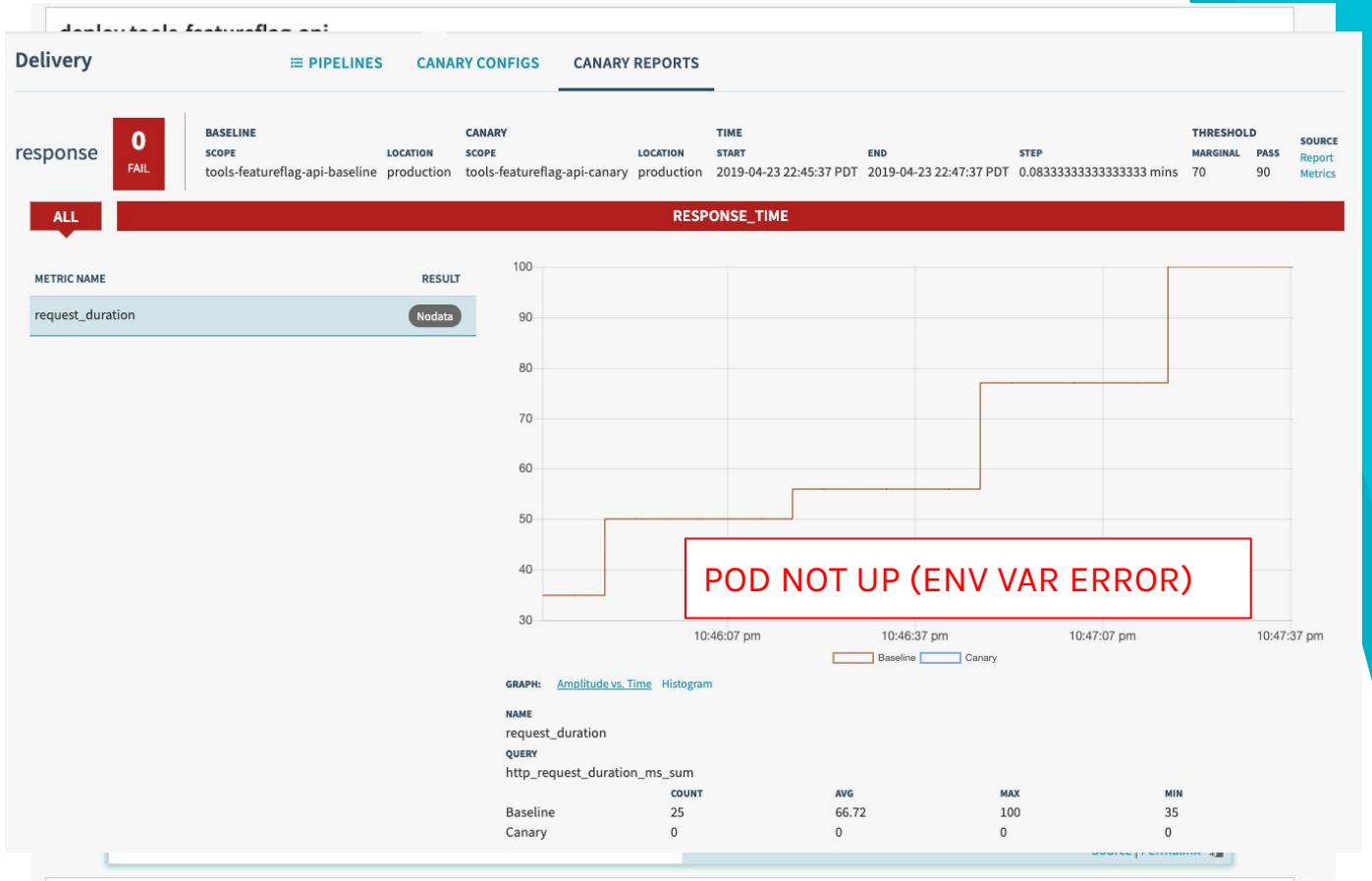
# Canary Config Filter



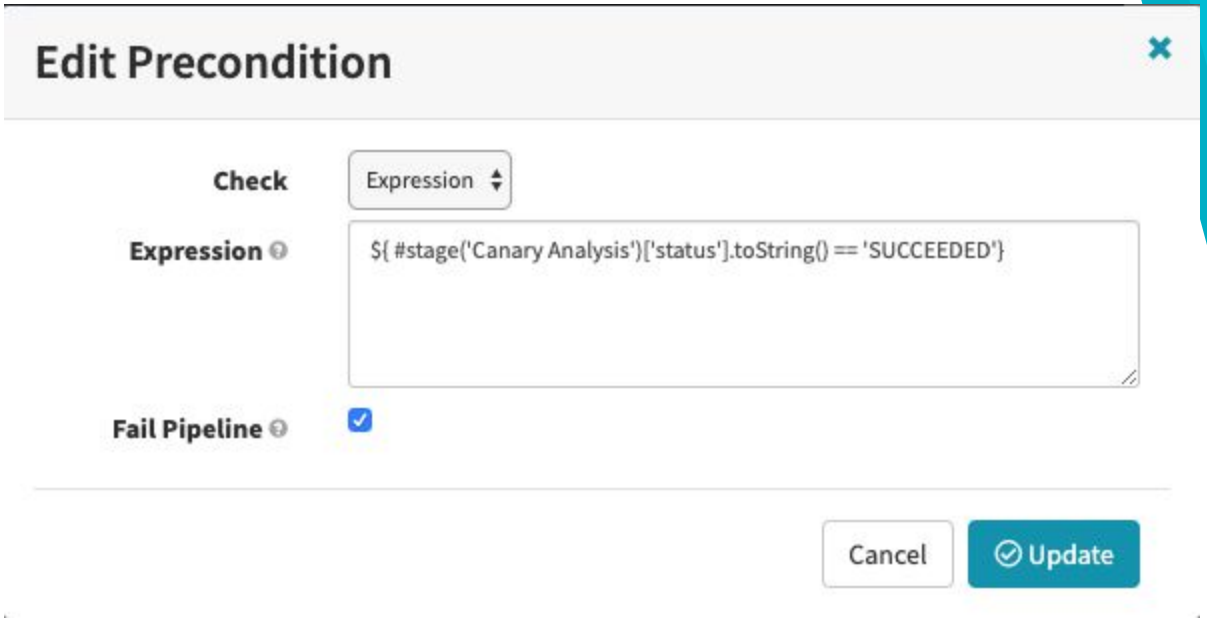
# Canary Analysis Report: Success



# Canary Analysis Report: Failed



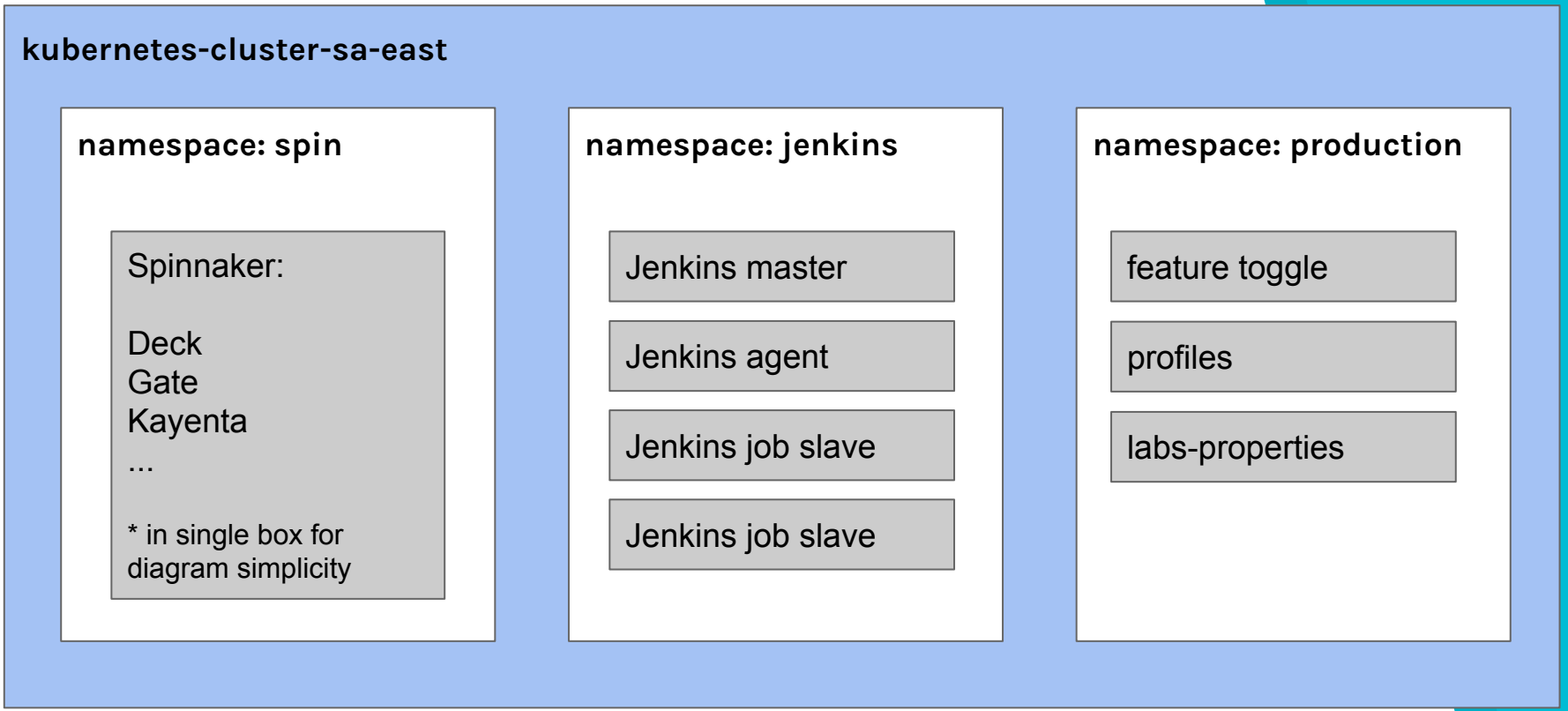
# Stage: Check


Edit Precondition
Check
Expression
Expression
${ #stage('Canary Analysis')['status'].toString() == 'SUCCEEDED'}
Fail Pipeline
Cancel
Update

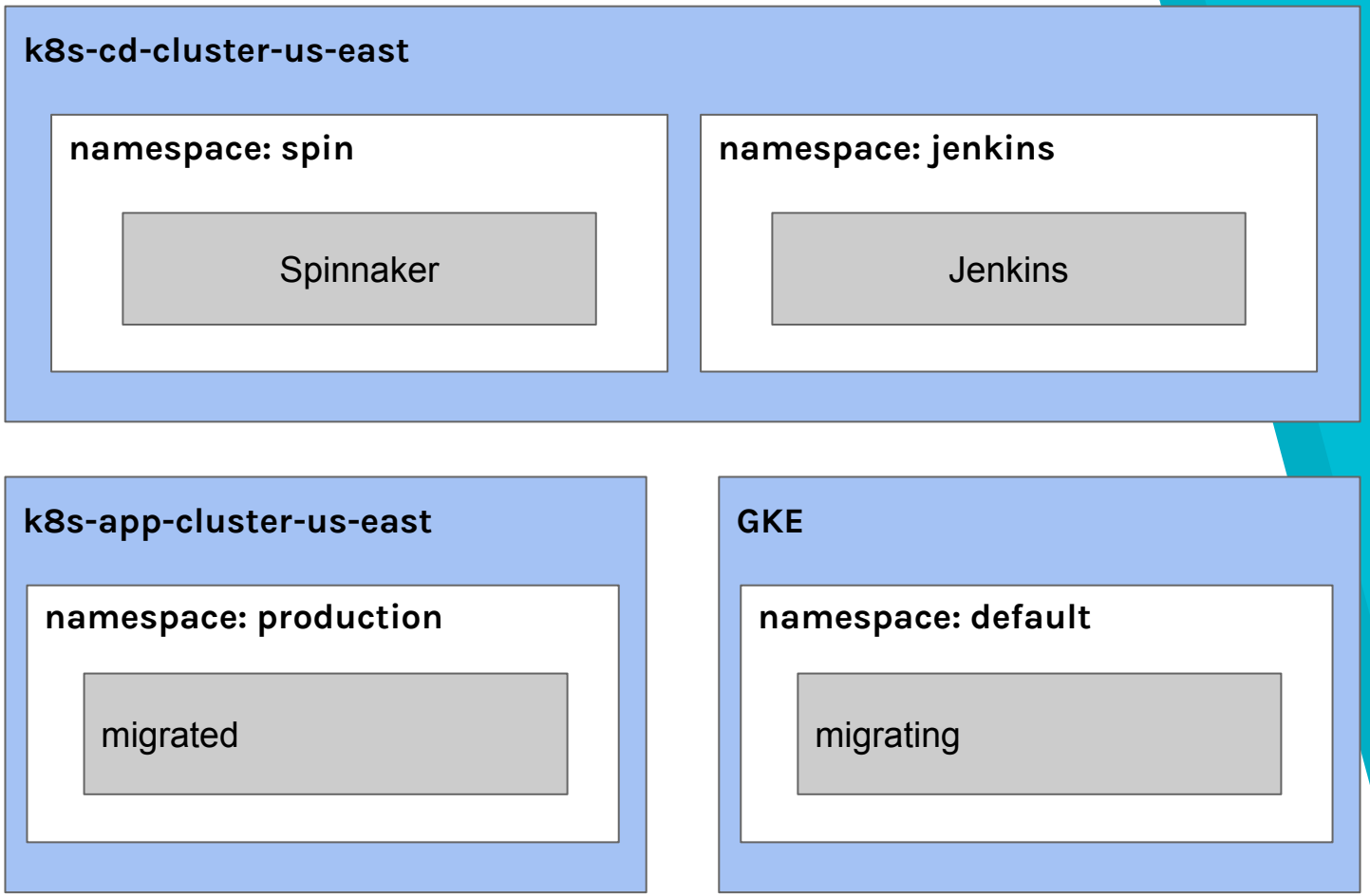
# 5.

# FUTURE

# P.O.C.


kubernetes-cluster-sa-east
namespace: spin
Spinnaker:
Deck
Gate
Kayenta
...
* in single box for diagram simplicity
namespace: jenkins
Jenkins master
Jenkins agent
Jenkins job slave
Jenkins job slave
namespace: production
feature toggle
profiles
labs-properties

# ONGOING

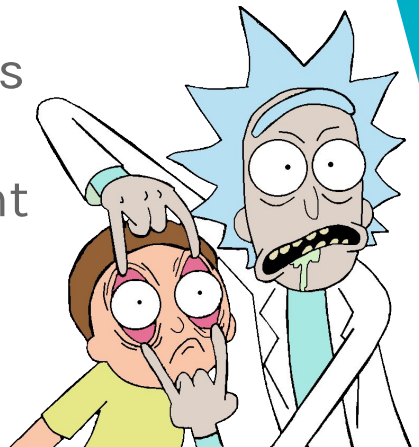

k8s-cd-cluster-us-east
namespace: spin
Spinnaker
namespace: jenkins
Jenkins
k8s-app-cluster-us-east
namespace: production
migrated
GKE
namespace: default
migrating

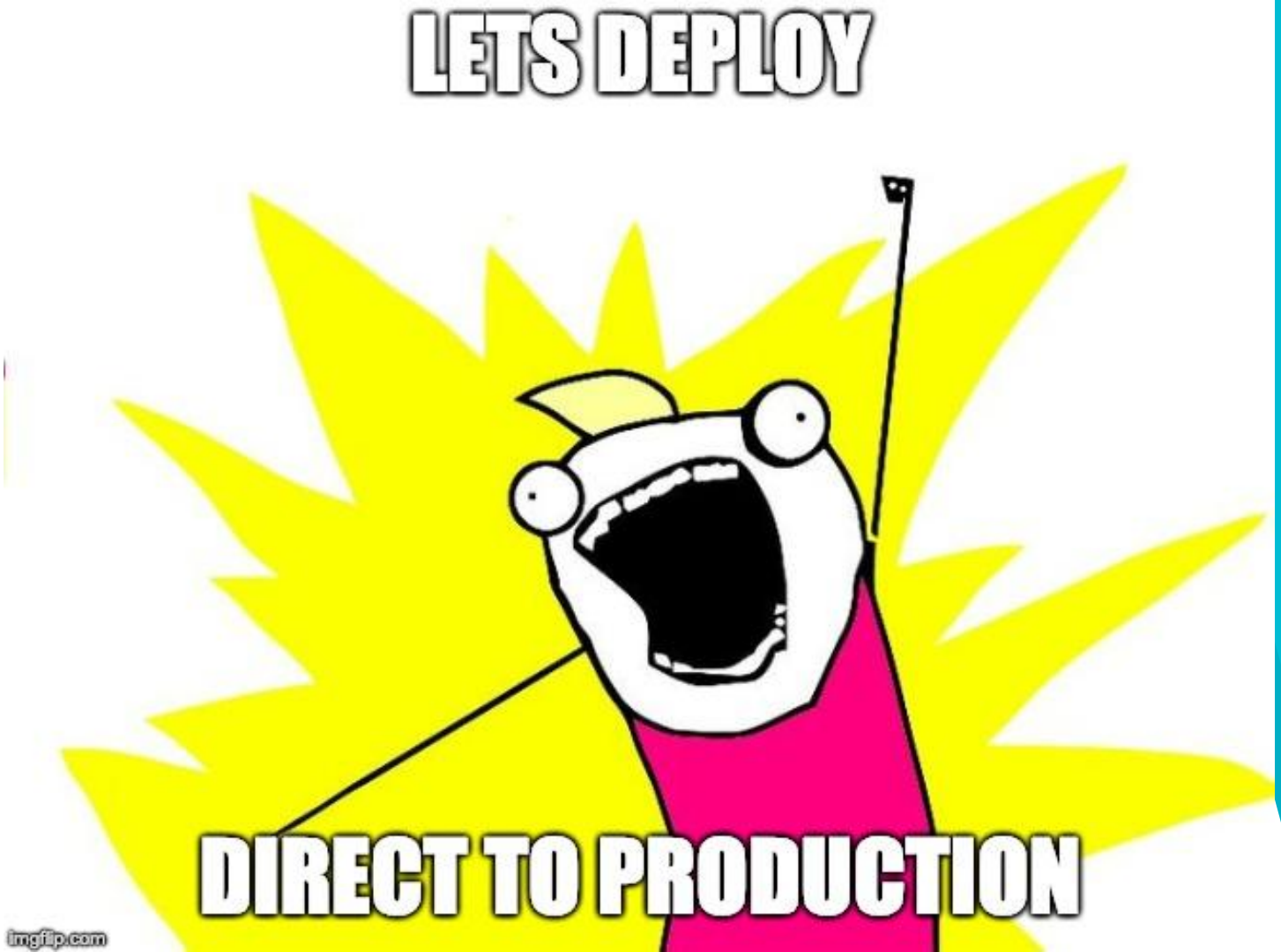
# ONGOING

- **Migrating CIs to Jenkins**
  - Wercker
  - * **CircleCI** *
  - Saves $ + Higher concurrency
  - **Standardizes** pipeline and **quality** measures
- **Jenkins pushing to both registries**
  - ECR
  - GCR
- **Spinnaker aware of both clusters**
- **Kayenta metrics**
  - Response time
  - Error rates
  - Resource consumption

# FUTURE

- Canary Analysis config improvement
  - What metrics?
  - Groups weights
  - Length/Windows
- Service Mesh?
- Manifests from SCM
- Pipeline in SCM
- Canary Analysis with business metrics
  - Custom collector?
- Automatic feature toggle management




LETS DEPLOY
DIRECT TO PRODUCTION
imgflip.com

# 6.

## Q&A

# 7.

# "THANK YOU" NOTES

rivendel

Livia

# Reúna todos os seus exames em um só lugar

https://liviasaude.com.br

# THE DEVELOPER'S CONFERENCE

# Contact

rodrigo.botti@gmail.com

rodrigo.botti@nexadigital.com.br

https://www.linkedin.com/in/rodrigo-botti/