



Playtika

Containerising bootiful microservices

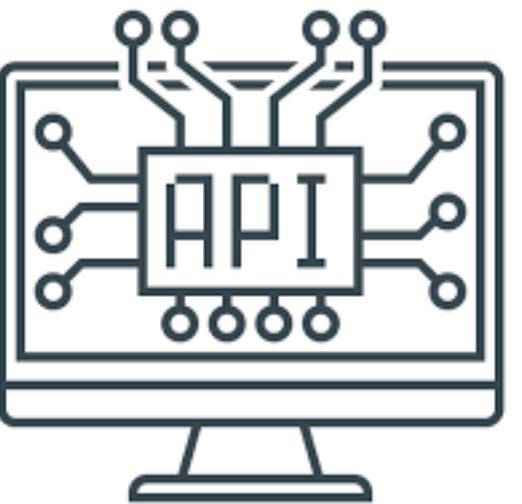
**by Ivan Vasyliev
Systems Architect**

About presenter



Ivan Vasyliev

Systems Architect at Playtika



15+ years



12+ years



[/vasilievip](#)



[/vasilievip](#)

Agenda

- Motivation
- Spring Cloud on Kubernetes
- Building containers from Spring Boot services
- QA and deployment automation



[Slides available by URL](#)

Motivation

Slotomania case study

~100+
spring cloud based services



~45+
one-pizza scrum teams



~250k+
simultaneous players

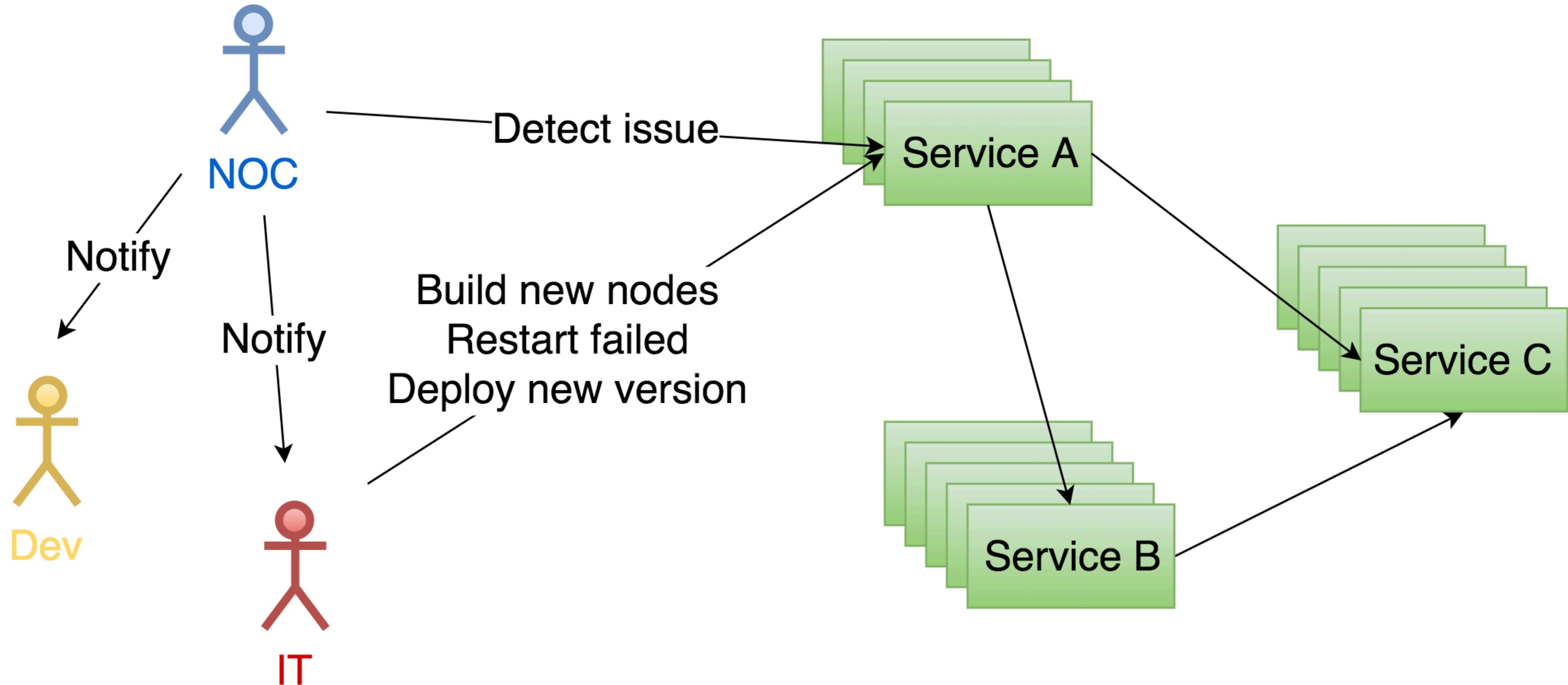


Evolved to implement as many features as possible as soon as possible

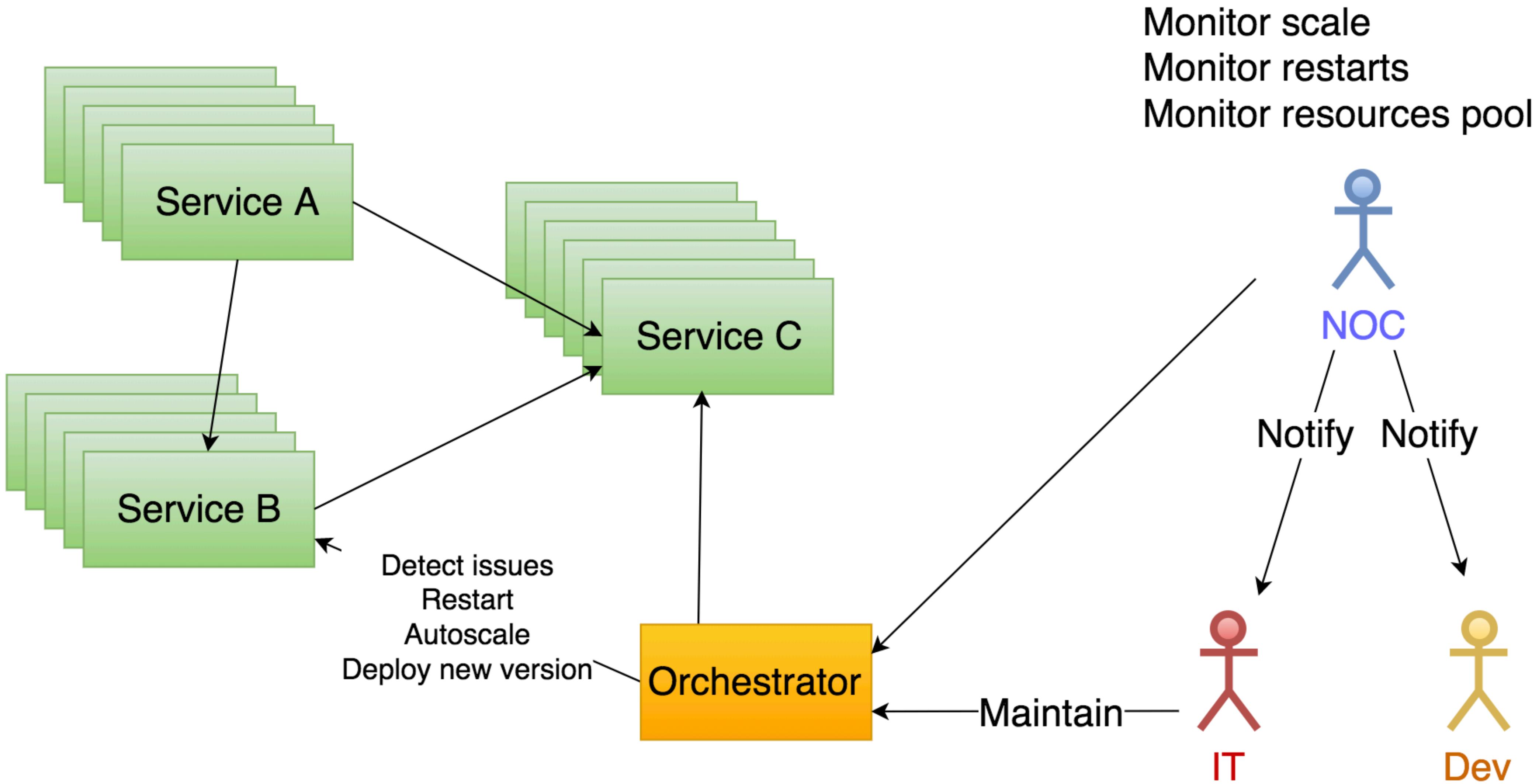
Technology stack

- Private datacenter + own hardware via VMware
- Microservices with Spring Boot/Cloud Netflix
- Couchbase/Kafka/MariaDB/Neo4j/Redis/Aerospike/MemSQL

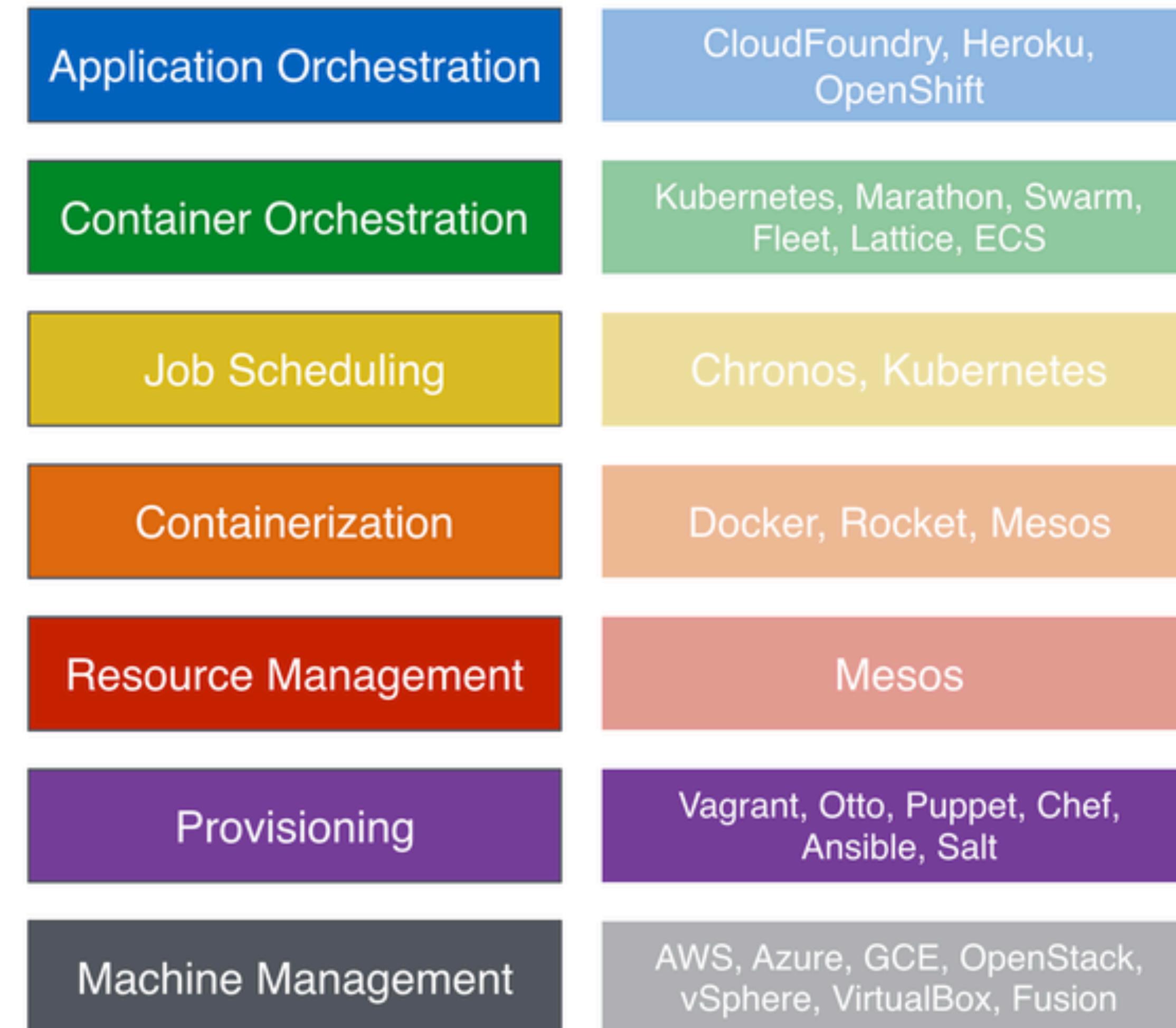
Microservices, first iteration



Microservices, next iteration



Orchestration landscape



[Docker vs Kubernetes vs Mesos vs OpenShift vs OpenStack](#)

Container technologies

	Docker	Rocket (rkt)	Mesos Universal Container Runtime
Github contributors	1,700+	200+	200+(?)
Stackoverflow questions	35,000+	40+	300+(?)
Orchestrators	Kubernetes Mesosphere Marathon Swarm Nomad	Kubernetes Nomad	Mesosphere Marathon

https://github.com/vasilievip/javaeeconf-spring-cloud-to-k8s/blob/master/container_selection.pdf

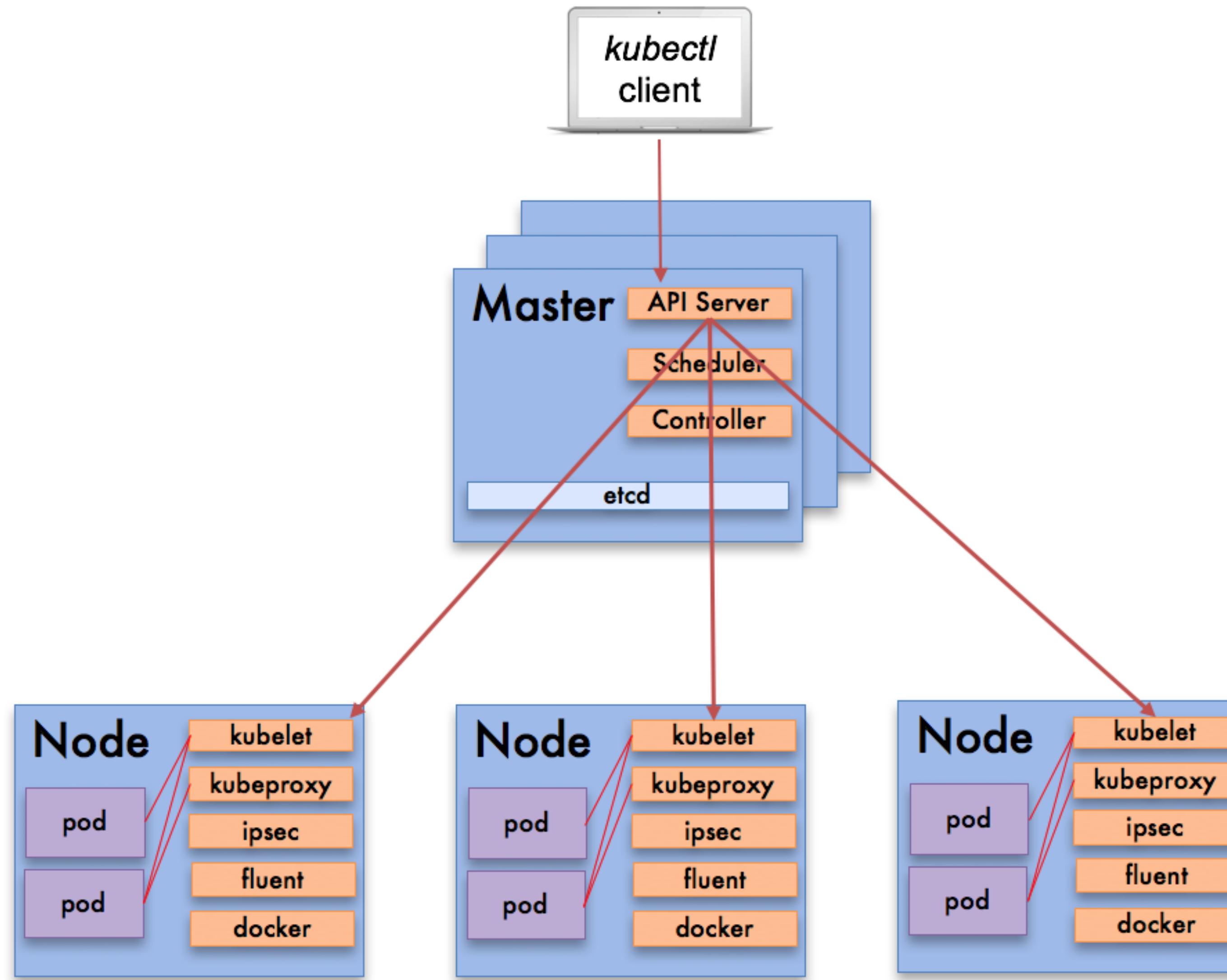
Container orchestration

	Kubernetes	Docker Swarm	Mesosphere Marathon	Hashicorp Nomad
Github contributors	1,600+	150+	250+	250+
Stackoverflow questions	7,000+	140+	450+	30+
Pipelining tool	Spinnaker Wercker Shippable Jenkins Cloudmunch	GoCD jenkins	jenkins?	jenkins?

https://github.com/vasilievip/javaeeconf-spring-cloud-to-k8s/blob/master/orchestration_selection.pdf

Lets talk about
Kubernetes and Spring Cloud Netflix

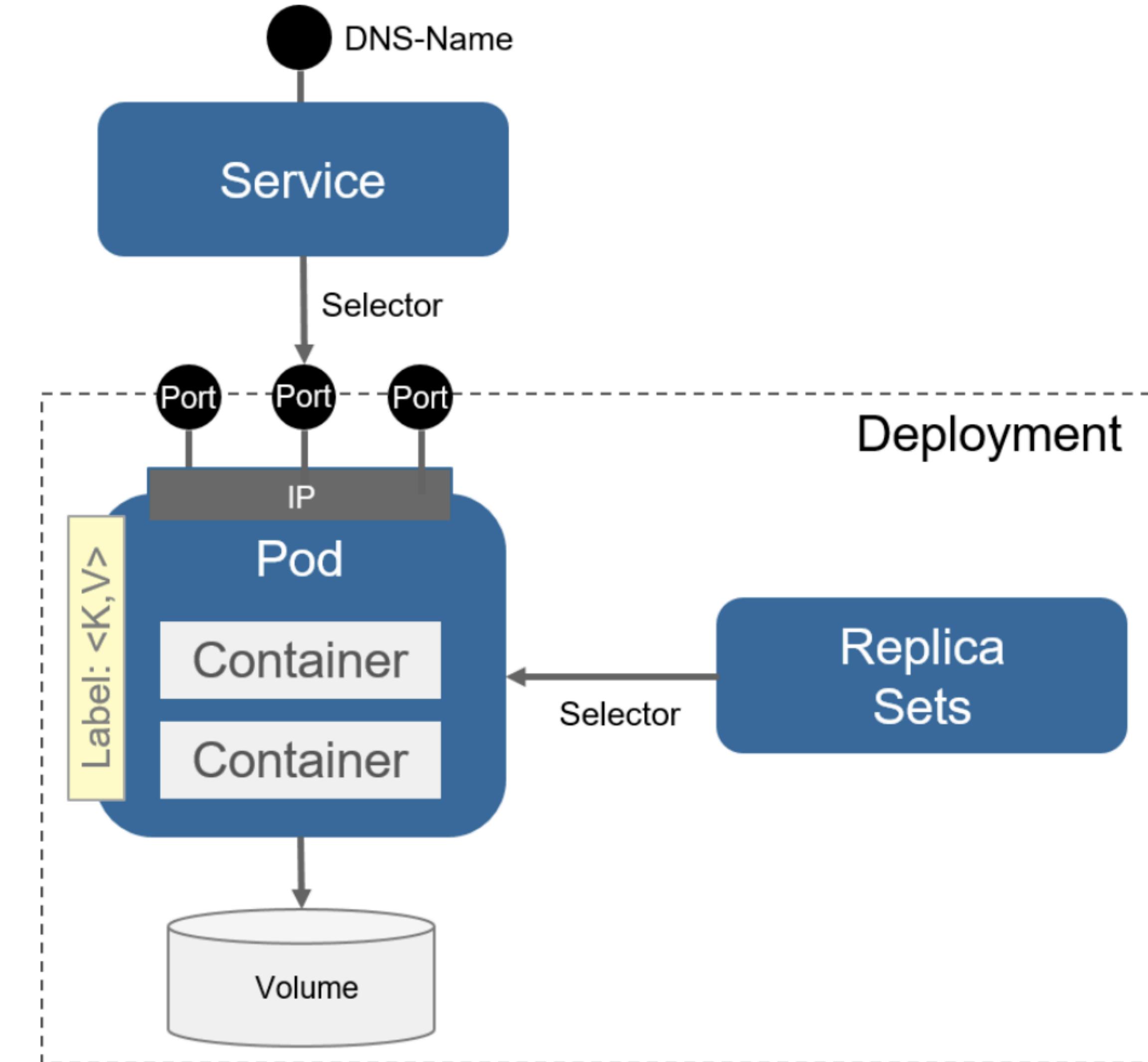
What is Kubernetes?



<https://www.dasher.com/containers-os-virtualization-to-workload-virtualization/>

Kubernetes concepts

- “Service” used for DNS/Load balancer
- “POD” represents coupled containers
- “Replica” set ensures you have desired amount of POD’s running
- “Deployment” provides declarative way to setup
- “Labels” are tags that can be attached to any object and used for selection



Kubernetes deployment

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: hello-world
spec:
  replicas: 1
  template:
    metadata:
      labels:
        tier: web
    spec:
      containers:
      - name: hello-world
        image: "nginx"
      ports:
      - containerPort: 80
```

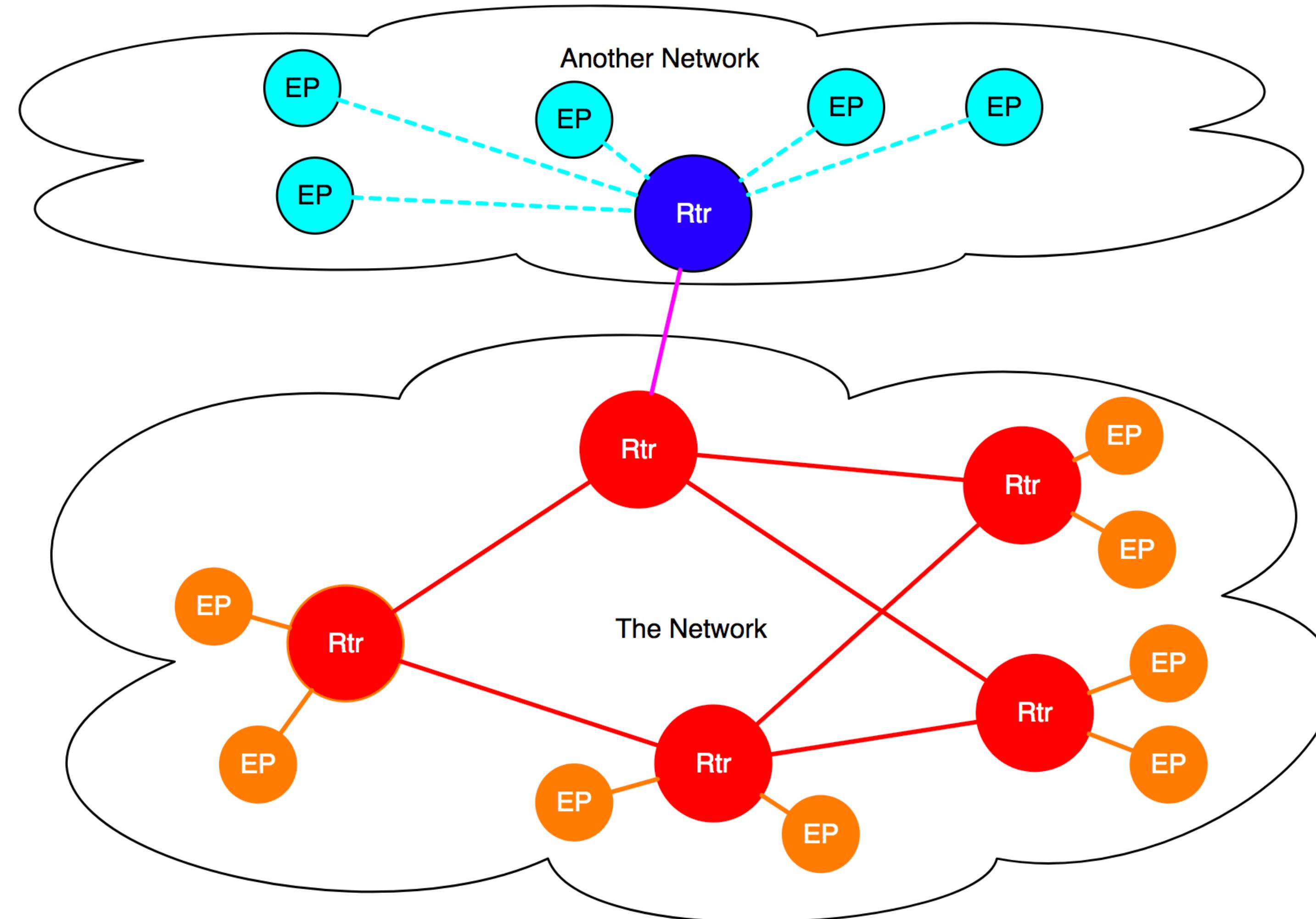
kubectl create -f nginx.yml

GKE looks good, but what about private Cloud?

- Kops installer - runs only AWS
- Kubeadm installer - semi-automated
- Kismatic installer - fully automated (support contract available)

<https://ramitsurana.gitbooks.io/awesome-kubernetes/content/#installers>

Linking subnets “IP to IP” via Calico



Kubernetes vs Spring Cloud Netflix

- Failover for service nodes
- Maintaining state of cluster
- Autoscaling
- Deployment automation
- Service discovery
- Load balancing
- API gateway
- Config maps
- Service discovery
- Client side load balancing
- Circuit breaking
- API gateway
- Configuration service

<https://thenewstack.io/kubernetes-an-overview/>

<https://cloud.spring.io/spring-cloud-netflix/>

Service discovery via Eureka

- Pre-stop hook talks to Netflix Eureka on shutdown
- On shutdown put service as OUT OF SERVICE in Eureka and wait for 90 secs
- Eureka “hook” is part of the Docker image template for service

<https://kubernetes.io/docs/tasks/configure-pod-container/attach-handler-lifecycle-event/>

Ribbon vs partial degradation

- k8s load balancing features is out of use for spring cloud
- Default load balancing rule is round robin
- WeightedResponseTimeRule is an option to try
- Lower timeouts + add retries for GET operations

<https://github.com/Netflix/ribbon/wiki/Working-with-load-balancers>

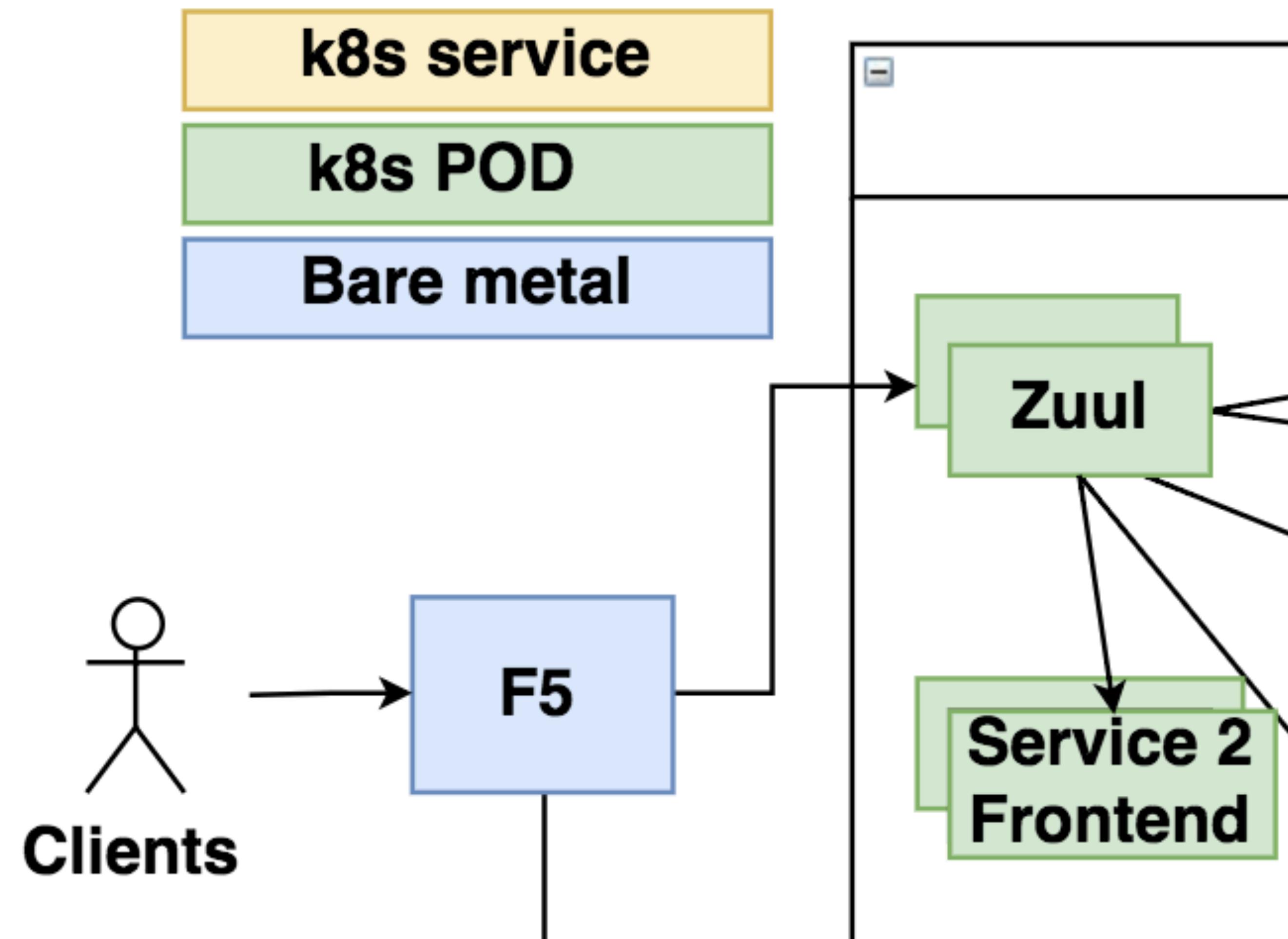
Gotchas - Docker engine crashes

- Start Docker daemon with live-restore option
- Setup Docker daemon watchdog to mitigate crashes

<https://docs.docker.com/config/containers/live-restore/>

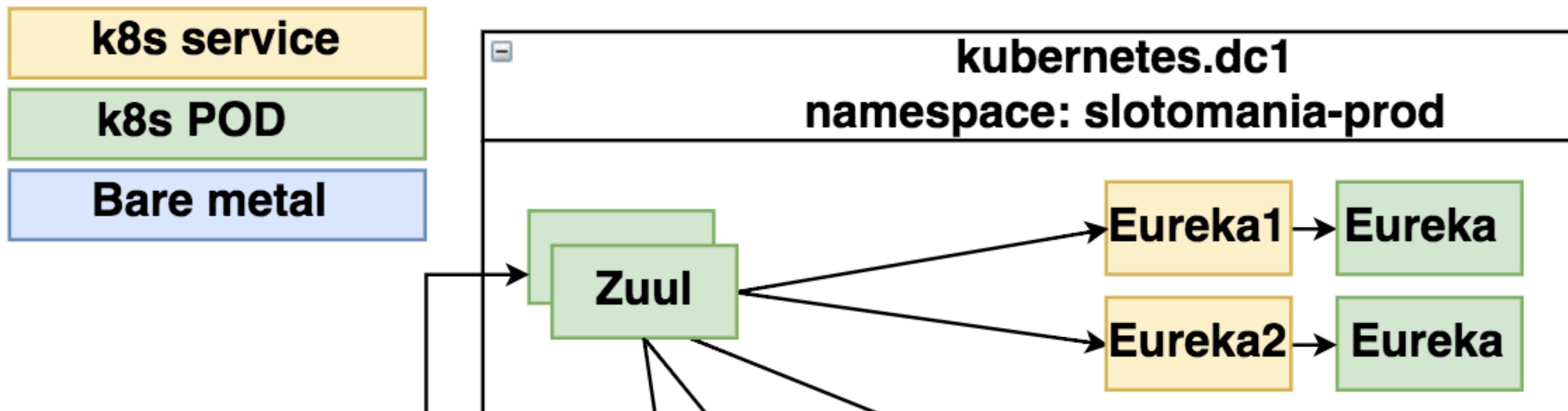
<https://github.com/apprenda/kismatic/issues/855>

F5 Load Balancer BIG-IP Controller

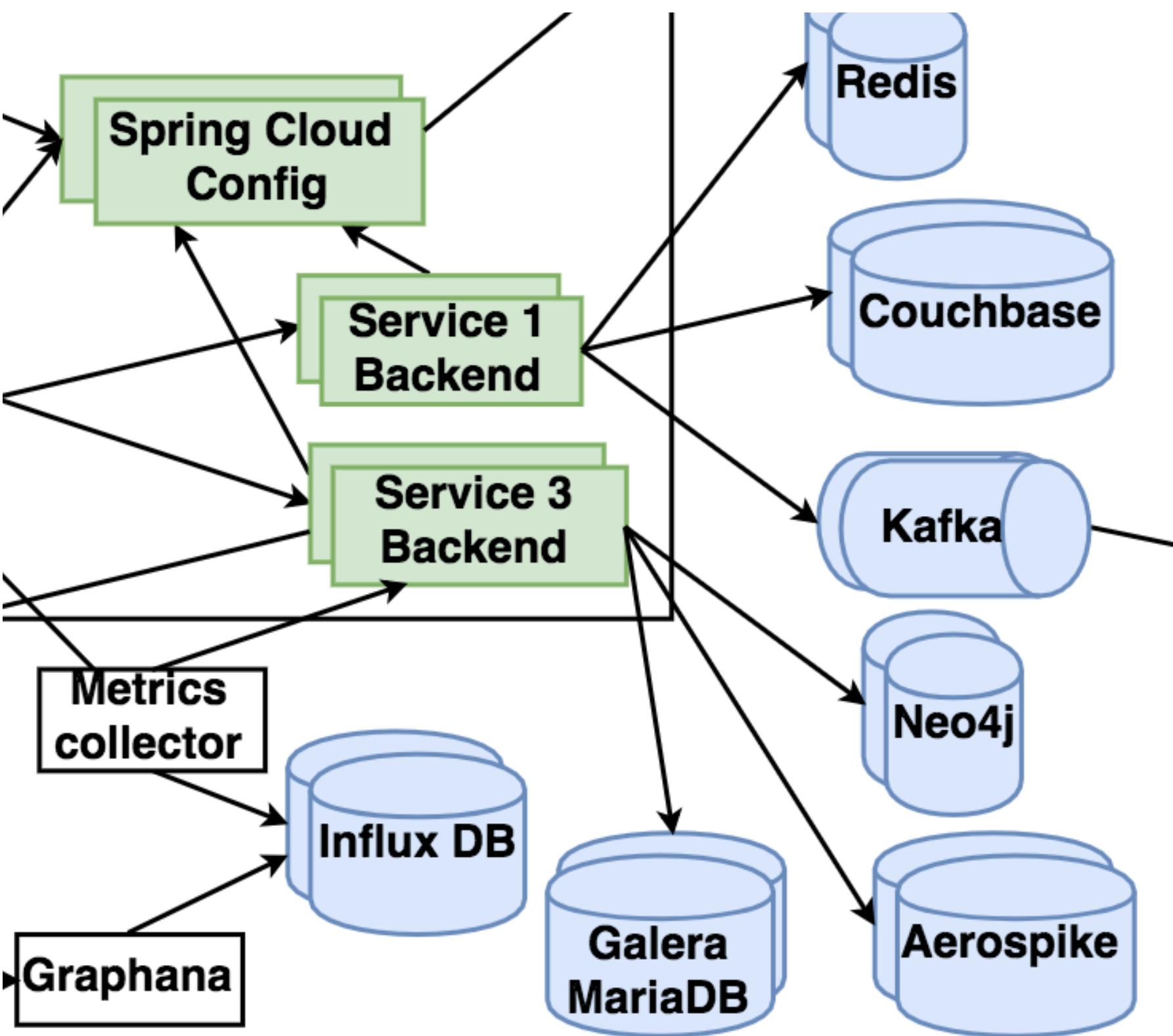


<http://clouddocs.f5.com/containers/v2/kubernetes/>

Eureka stateful service

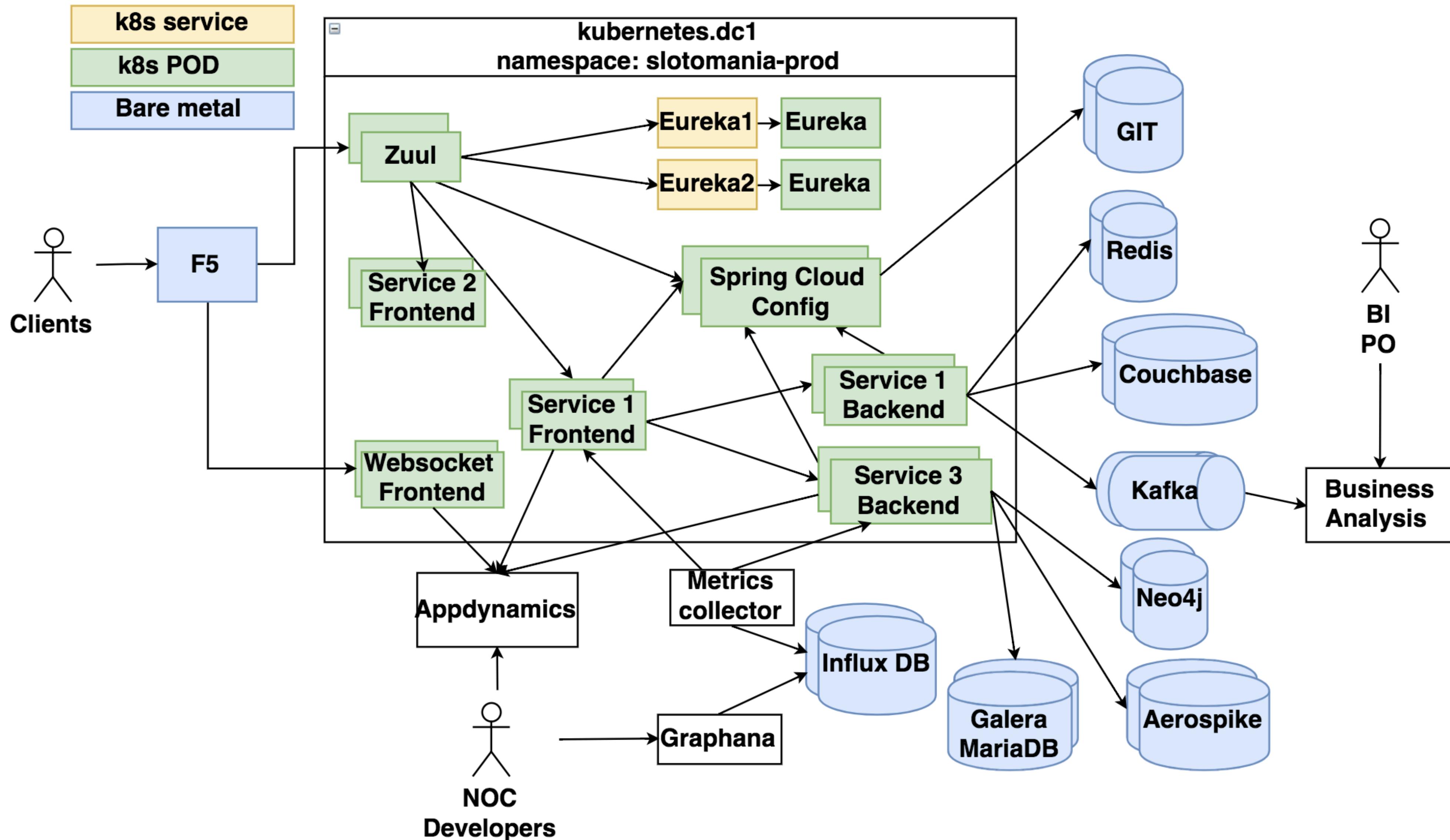


Databases still bare metal



[Should database live inside or outside the container](#)

Bird eye view



Try it at home

The screenshot shows a web-based interface for managing Kubernetes services. The top navigation bar includes 'Dashboard', 'Namespaces', 'Pods', 'Events', 'Services', 'Ingress', 'ConfigMaps', 'Secrets', 'PersistentVolume', 'PersistentVolumeClaim', 'Service Accounts', 'Roles', 'Role Bindings', and 'Cluster Roles'. The 'Services' tab is active.

The main content area is titled 'Services' and displays a table with the following data:

Name	Labels	Cluster IP	Internal endpoints	External endpoints	Age
✓ hystrix-dashboard	run: hystrix	10.27.248.67	hystrix-dashboard:... hystrix-dashboard:...	104.155.186.217:8...	a minute
✓ eureka-server	run: eureka-server	10.27.254.240	eureka-server:8761... eureka-server:3231...	130.211.185.23:87...	an hour
✓ kubernetes	component: apis...	10.27.240.1	kubernetes:443 TCP	-	an hour

<https://luizkowalski.net/deploying-spring-cloud-netflix-apps-on-kubernetes/>

Lets talk about
Dockerising Spring Boot

Docker file template

```
FROM @docker.baseImage.serverId@/com.playtika.containers/service-container-base-image:@docker.baseImage.version@

ADD service-container-eureka-client.jar      /opt/scripts/eureka-client.jar
ADD preStop.sh                                /opt/scripts/preStop.sh
ADD runApplication                           /opt/scripts/runApplication
ADD log4j2-console.xml                      /etc/playtika/log4j2-console.xml
ADD log4j2-kibana-console.xml                /etc/playtika/log4j2-kibana-console.xml
ADD log4j2-kibana-eat.xml                   /etc/playtika/log4j2-kibana-eat.xml
ADD @project.build.finalName@.jar            /etc/playtika/application.jar

RUN \
    chmod +x /opt/scripts/runApplication && \
    chmod +x /opt/scripts/preStop.sh

ENV JAVA_OPTS="-Dsun.misc.URLClassPath.disableJarChecking=true -Djava.security.egd=file:/dev/urandom"
ENV SERVICE_NAME="@service.name"

# configuration service should provide following options in JAVA_OPTS
# -Dappdynamics.controller.hostName
# -Dappdynamics.controller.port
# -Dappdynamics.controller.ssl.enabled
# -Dappdynamics.agent.accountAccessKey
# -Dappdynamics.agent.accountName
# -Dappdynamics.agent.applicationName=

ENTRYPOINT exec /usr/bin/python3 /opt/scripts/runApplication
```

This results into:
exec java \$JAVA_OPTS -jar application.jar \$RUN_ARGS

Docker build injection via parent

```
<project>
...
<parent>
    <groupId>com.playtika.services</groupId>
    <artifactId>infra-libraries-bom</artifactId>
    <version>FIND LATEST VERSION</version>
</parent>
...
<properties>
    <service.name>your-service-name</service.name>
</properties>
...
</project>
```

Extends spring-boot-parent

Docker build injection via mixin

```
<plugin>
  <groupId>com.github.odavid.maven.plugins</groupId>
  <artifactId>mixin-maven-plugin</artifactId>
  <version>${mixin-maven-plugin.version}</version>
  <extensions>true</extensions>
  <configuration>
    <mixins>
      <mixin>
        <groupId>com.playtika.containers</groupId>
        <artifactId>service-container-maven-mixin</artifactId>
        <version>${service.container.maven.mixin.version}</version>
      </mixin>
    </mixins>
  </configuration>
</plugin>
```

Notes for Java on Docker

- Java SE 8u131 adds experimental support for docker
- You still need to setup XMX for JVM :(
- Use “exec java” to get pid 1 for java
- Escape whitespaces when passing parameters via ENV

[Java support for docker cpu and memory limits](#)

Lets talk about QA

Deployment automation

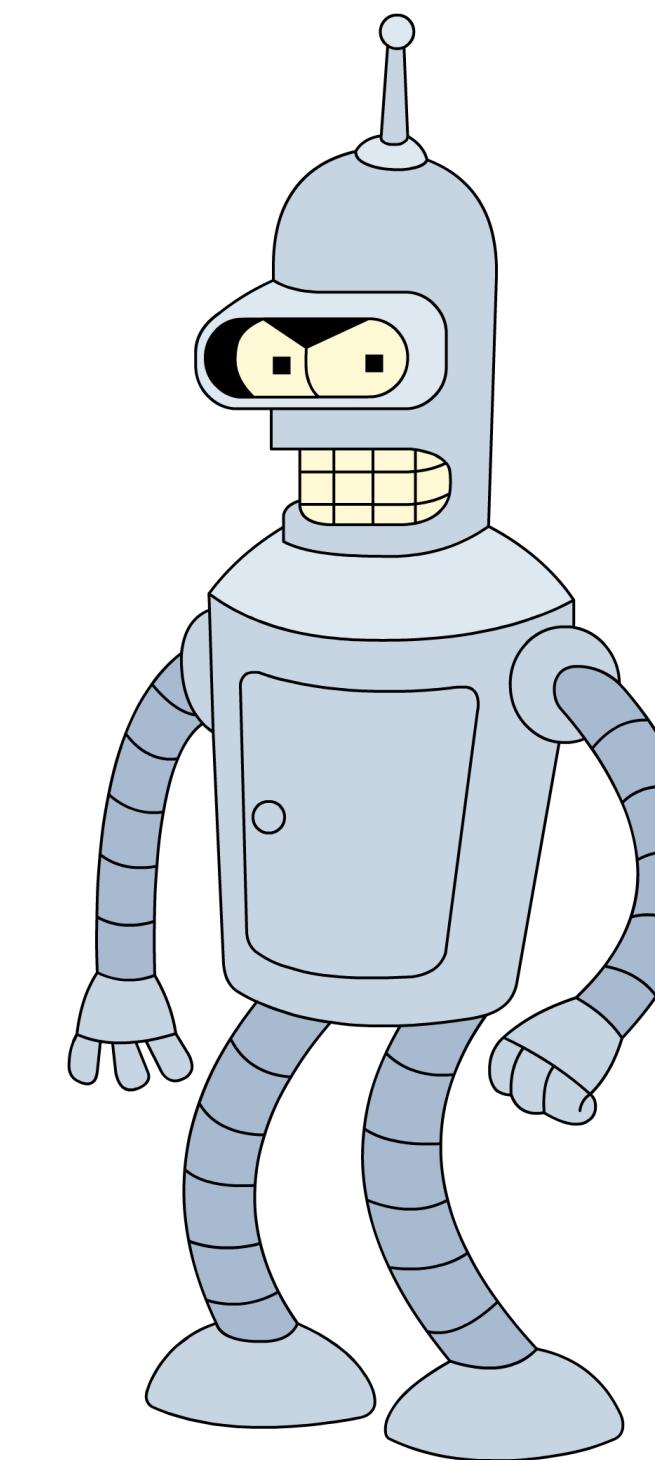
Continuous delivery

Each commit is “deployable”



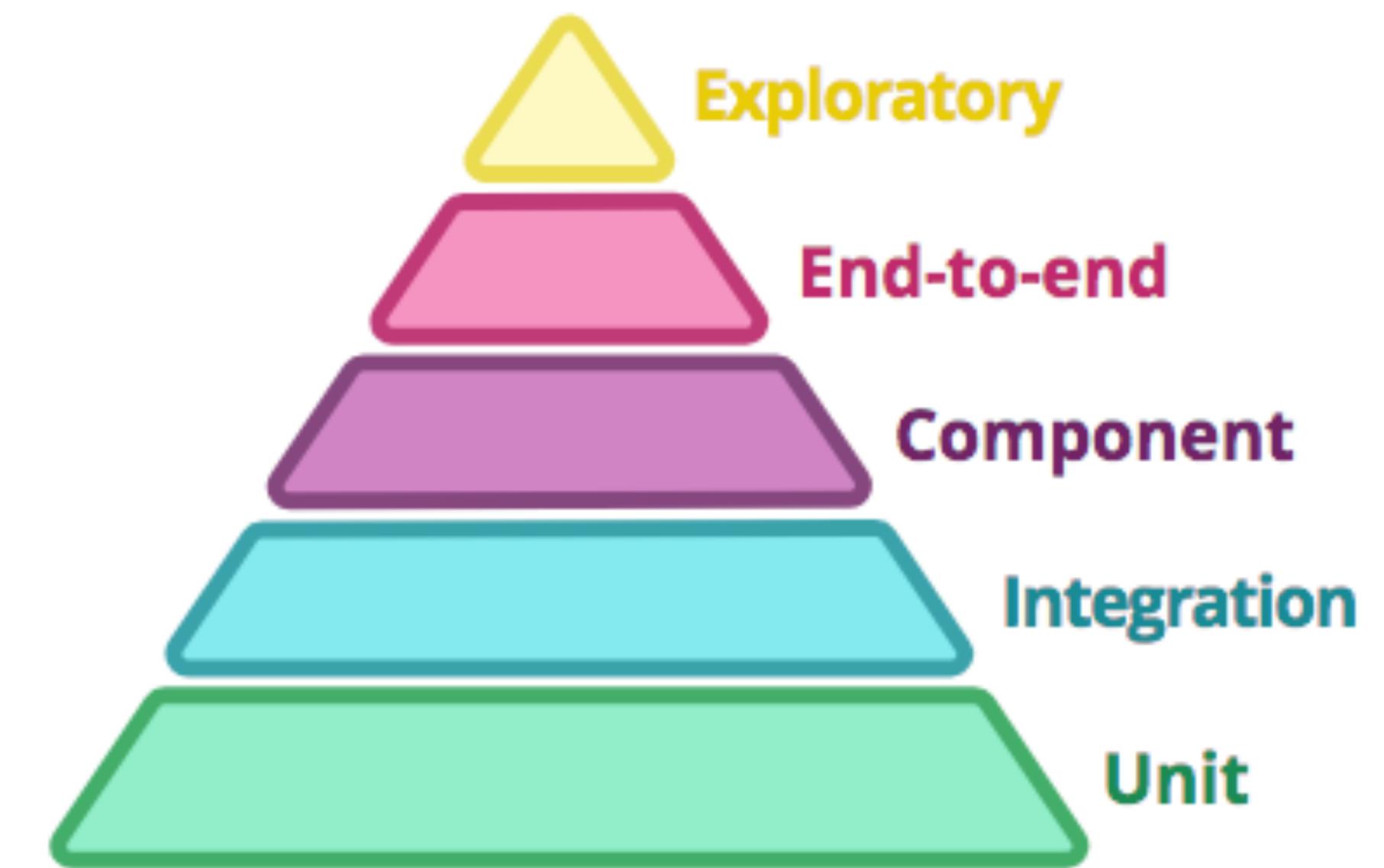
Continuous deployment

Each commit is “deployed”



QA strategy for microservices

- Run ALL tests during build
- Run exploratory/end-to-end on production
- Feature toggles per user
- Evaluate Spring Cloud Contracts



<https://martinfowler.com/articles/microservice-testing/#conclusion-test-pyramid>

External smoke tests vs Health indicators

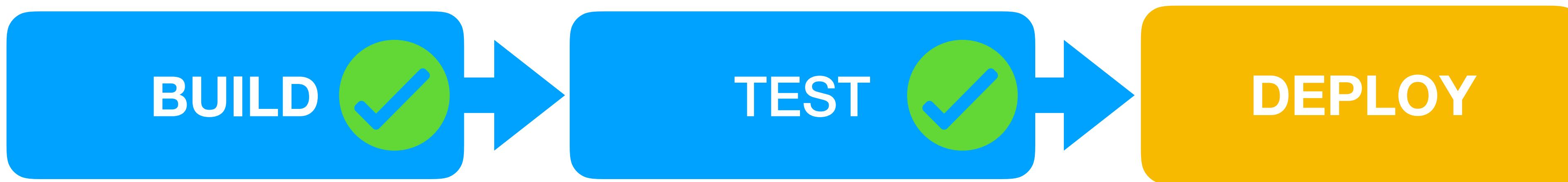
Health checks		
Service name	Status	Stacktrace
Database	UP	
Email	DOWN	

[Readiness and liveness probes](#)

[Spring boot health indicators](#)

Lets talk about deployment

Version release



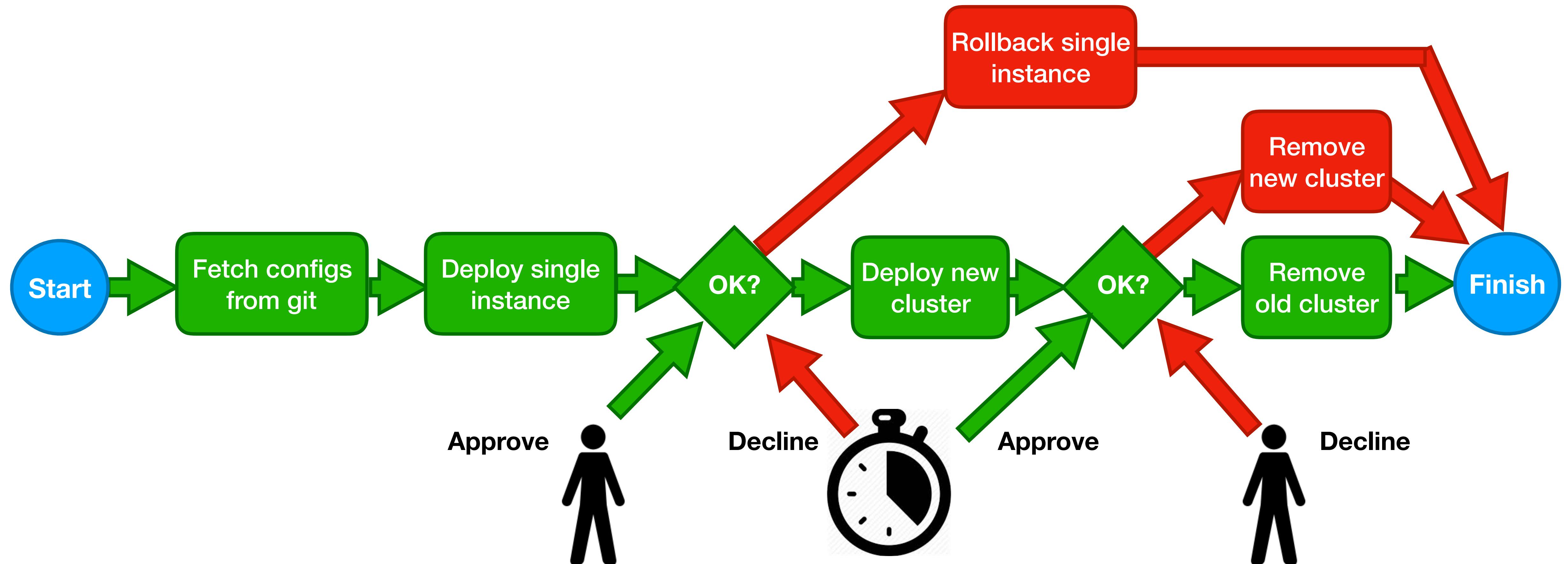
<https://www.thoughtworks.com/radar/techniques/automated-deployment-pipeline>

Deployment patterns

- Zero downtime deployments
- Canary + rolling update
- Canary + blue-green
- Pipeline as a code

<https://www.thoughtworks.com/radar/techniques/pipelines-as-code>

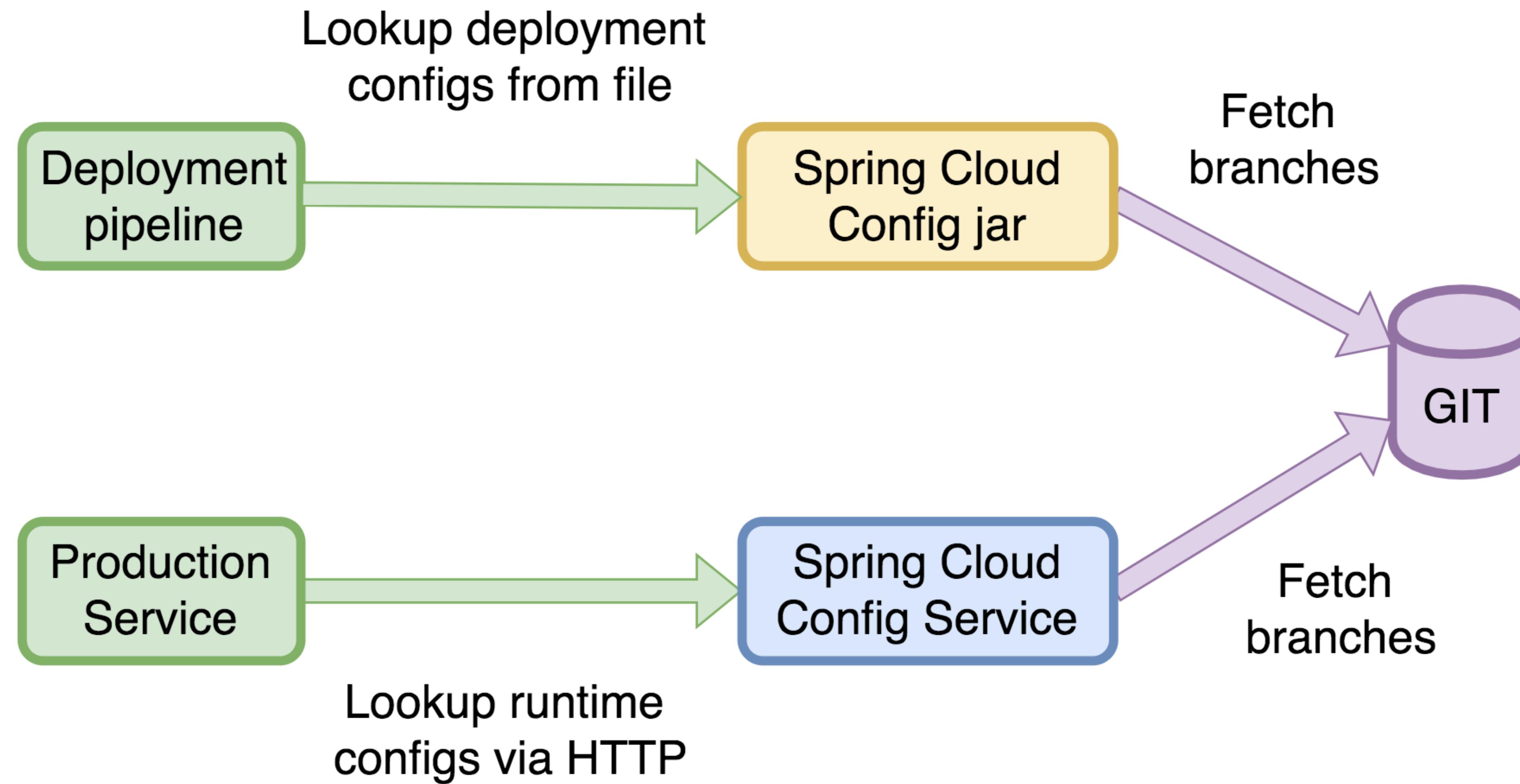
Deployment steps



Implementation details

- kubectl to talk to k8s
- k8s “deployment controller” to push new versions
- Deployment parameters via spring cloud config

Spring cloud config in pipeline



Jenkins pipeline DSL

The screenshot shows a code editor window for a Jenkins Pipeline script named `UpgradeServicePipeline.groovy`. The script is written in Groovy and defines a pipeline structure. The code includes comments, variables, stages, and error handling logic. The editor uses color coding and syntax highlighting to distinguish between different parts of the code.

```
1 package com.playtika.deployment.jenkins.pipelines
2
3 ...
4
5 node() {
6
7     sendInProgressEmail()
8
9     lock( resource: "EnvironmentName: ${EnvironmentName}" ) {
10        try {
11
12            stage( name: 'Upgrade Service: Get service name by branch' ) {...}
13
14            stage( name: 'Upgrade Service: Trigger deployment pipeline' ) {...}
15
16            stage( name: 'Send Report' ) {sendResultingEmail( status: 'SUCCESS', reportEntriesHolder)}
17
18        } catch (Exception e) {
19            echo message: "Caught error while pipeline execution: ${e}"
20            sendResultingEmail( status: 'FAILURE', reportEntriesHolder)
21            currentBuild.result = 'FAILURE'
22
23        }
24    }
25
26
27 }
```

Jenkins job DSL

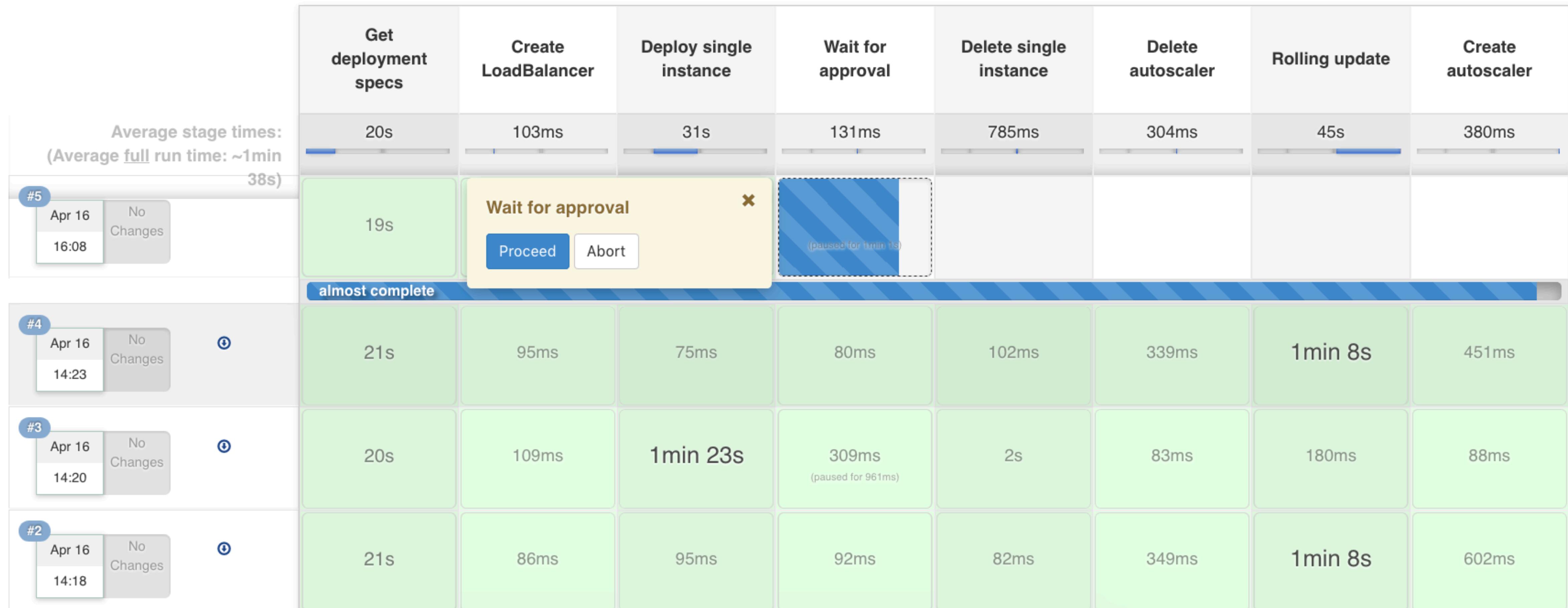
The screenshot shows a code editor window with the title "CreateDeploymentAutoscalerJob.groovy". The code is written in Groovy DSL for Jenkins jobs. It defines a job within a folder named 'deployService'. The job has various configurations like parameters, properties, concurrent build, log rotator, wrappers, lockable resources, and steps. The steps include printing job parameters and running Kubernetes commands to rollout and autoscale a deployment.

```
1 package com.playtika.deployment.jenkins.jobs
2
3 String parentFolder = 'deployService'
4
5 folder(parentFolder)
6
7 job(s: "$parentFolder/CreateDeploymentAutoscaler") {
8     parameters {...}
9     properties {
10         buildFailureAnalyzer()
11     }
12     concurrentBuild(allowConcurrentBuild: true)
13     logRotator {...}
14     wrappers {...}
15     lockableResources {...}
16     steps {
17         buildDescription regexp: '', description: '[serviceName: $serviceName, namespace: $namespace]'
18         shell """
19             echo "Job parameter: namespace=\${namespace}"
20             echo "Job parameter: serviceName=\${serviceName}"
21             echo "Job parameter: teamName=\${teamName}"
22             echo "Job parameter: parentJobId=\${parentJobId}"
23             echo "Job parameter: deploymentId=\${deploymentId}"
24
25             sanitizedServiceName=\$(echo "\${serviceName}" | sed -e 's/[A-Za-z0-9]//g' | tr '[:upper:]' '[:lower:]')
26             kubectl -n "\$namespace" rollout status "deploy/\$sanitizedServiceName"
27             kubectl -n "\$namespace" autoscale "deploy/\$sanitizedServiceName" --min="\$min" --max="\$max" --cpu-percent="\$cpu"
28             """.stripIndent().trim()
29     }
30 }
```

<https://github.com/jenkinsci/job-dsl-plugin>

Canary+rolling Jenkins pipeline

Stage View



Namespace overview page

All checks are based on [Slotomania dynamic deployment page](#) [Refresh](#)

SLOTOMANIA KUBERNETES PROD HEALTH

<p>Service: bundles-service 3 3 0</p> <table border="1" style="width: 100%; border-collapse: collapse;"><tbody><tr><td style="width: 33%; padding: 5px;">Server: bundlesservice-v000-bsbj7 1.0.14 UP</td><td style="width: 33%; padding: 5px;">Server: bundlesservice-v000-dmlpq 1.0.14 UP</td><td style="width: 33%; padding: 5px;">Server: bundlesservice- 1.0.14 UP</td></tr></tbody></table>	Server: bundlesservice-v000-bsbj7 1.0.14 UP	Server: bundlesservice-v000-dmlpq 1.0.14 UP	Server: bundlesservice- 1.0.14 UP	<p>Service: cash-or-smash 3 3 0</p> <table border="1" style="width: 100%; border-collapse: collapse;"><tbody><tr><td style="width: 33%; padding: 5px;">Server: cashorsmash-v000-jdjw4 1.0.1 UP</td><td style="width: 33%; padding: 5px;">Server: cashorsmash-v000-lztnv 1.0.1 UP</td><td style="width: 33%; padding: 5px;">Server: cashorsmash-v000- 1.0.1 UP</td></tr></tbody></table>	Server: cashorsmash-v000-jdjw4 1.0.1 UP	Server: cashorsmash-v000-lztnv 1.0.1 UP	Server: cashorsmash-v000- 1.0.1 UP	<p>Service: coupons-backend-aerospike 8 8 0</p> <table border="1" style="width: 100%; border-collapse: collapse;"><tbody><tr><td style="width: 33%; padding: 5px;">Server: couponsbackendaerospike-v000-7x72h 1.9-AEROSPIKE UP</td><td style="width: 33%; padding: 5px;">Server: couponsbackendaerospike-v000-dt5wi 1.9-AEROSPIKE UP</td><td style="width: 33%; padding: 5px;"></td></tr></tbody></table>	Server: couponsbackendaerospike-v000-7x72h 1.9-AEROSPIKE UP	Server: couponsbackendaerospike-v000-dt5wi 1.9-AEROSPIKE UP	
Server: bundlesservice-v000-bsbj7 1.0.14 UP	Server: bundlesservice-v000-dmlpq 1.0.14 UP	Server: bundlesservice- 1.0.14 UP									
Server: cashorsmash-v000-jdjw4 1.0.1 UP	Server: cashorsmash-v000-lztnv 1.0.1 UP	Server: cashorsmash-v000- 1.0.1 UP									
Server: couponsbackendaerospike-v000-7x72h 1.9-AEROSPIKE UP	Server: couponsbackendaerospike-v000-dt5wi 1.9-AEROSPIKE UP										
<p>Service: coupons-frontend-aerospike 12 12 0</p> <table border="1" style="width: 100%; border-collapse: collapse;"><tbody><tr><td style="width: 100%; padding: 5px;">Server: couponsfrontendaerospike-v003-4f84v 1.8-AEROSPIKE UP</td></tr></tbody></table>	Server: couponsfrontendaerospike-v003-4f84v 1.8-AEROSPIKE UP	<p>Service: coupons-new-backend 8 8 0</p> <table border="1" style="width: 100%; border-collapse: collapse;"><tbody><tr><td style="width: 100%; padding: 5px;">Server: couponsnewbackend-v001-5w9n9 1.114 UP</td></tr></tbody></table>	Server: couponsnewbackend-v001-5w9n9 1.114 UP	<p>Service: coupons-new-frontend 12 12 0</p> <table border="1" style="width: 100%; border-collapse: collapse;"><tbody><tr><td style="width: 100%; padding: 5px;">Server: couponsnewfrontend-v000-2jsxq 1.112 UP</td></tr></tbody></table>	Server: couponsnewfrontend-v000-2jsxq 1.112 UP						
Server: couponsfrontendaerospike-v003-4f84v 1.8-AEROSPIKE UP											
Server: couponsnewbackend-v001-5w9n9 1.114 UP											
Server: couponsnewfrontend-v000-2jsxq 1.112 UP											

Version release page

Active versions - 1 :: infra.MobyDick [+ Create new release](#)

SM: feature/AngryBeavers

Version creation	Stage deployment	Pre-prod deployment	Prod deployment
<input checked="" type="checkbox"/> Jira version created <input type="checkbox"/> PV branch created <input type="checkbox"/> Order stage <input type="checkbox"/> Start deployment to stage	<input type="checkbox"/> Deploying to stage <input type="checkbox"/> Deployed to stage <input type="checkbox"/> QG Tests in process <input type="checkbox"/> QG Tests completed <input type="checkbox"/> Create pull request (PR) <input type="checkbox"/> Waiting for PR approval <input type="checkbox"/> Create pre-prod ticket	<input type="checkbox"/> Waiting for assignee <input type="checkbox"/> Deploying to pre-production <input type="checkbox"/> Waiting for Test Rail creation <input type="checkbox"/> S2P by version in process <input type="checkbox"/> Acceptance test <input type="checkbox"/> Waiting for QA <input type="checkbox"/> Live and verified <input type="checkbox"/> Create prod ticket	<input type="checkbox"/> Waiting for assignee <input type="checkbox"/> Waiting for approval <input type="checkbox"/> Approved. Waiting for deployment <input type="checkbox"/> Deploying to production <input type="checkbox"/> Waiting for QA <input type="checkbox"/> Live and verified <input type="checkbox"/> Prod ticket closed

feature/AngryBeavers [Version details](#)

<input checked="" type="checkbox"/> Version name SomeRelease 1.12	<input checked="" type="checkbox"/> Current state Version was created
<input checked="" type="checkbox"/> Game Slotomania	<input type="checkbox"/> SM pool status Ready - 0 Updating - 0
<input type="checkbox"/> Created by Ivan Vasylyev	
<input type="checkbox"/> PV branch feature/AngryBeavers	
<input type="checkbox"/> BIG tests branch master	
<input type="checkbox"/> UI tests branch production	

[Order stage](#) [Cancel version](#)

Summary

- Kubernetes is winning orchestration solution at the moment
- You can mix spring cloud and k8s to meet your setup
- Jenkins is not only for CI, can be used as pipelining tool
- Consider implementing own namespace overview UI

Useful links

- Kubernetes vs Spring Cloud
 - <https://dzone.com/articles/deploying-microservices-spring-cloud-vs-kubernetes>
- Docker files for java on ubuntu
 - <https://github.com/dockerfile/java>
- How to guides
 - <https://cloud.google.com/solutions/continuous-delivery-spinnaker-kubernetes-engine>
 - <https://cloud.google.com/solutions/continuous-delivery-jenkins-kubernetes-engine>
- Component/integration testing with Docker
 - <https://github.com/vasilievip/testcontainers-component-test>
 - <https://github.com/Playtika/testcontainers-spring-boot>
- Kubernetes LDAP/DNS integration
 - <https://github.com/apprenda-kismatic/kubernetes-ldap>
 - <https://coredns.io/>

Questions?



Playtika
