Playtika

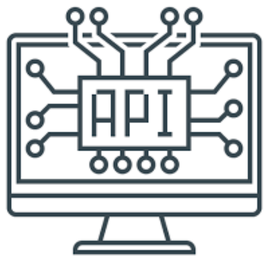
Containerising bootiful microservices

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

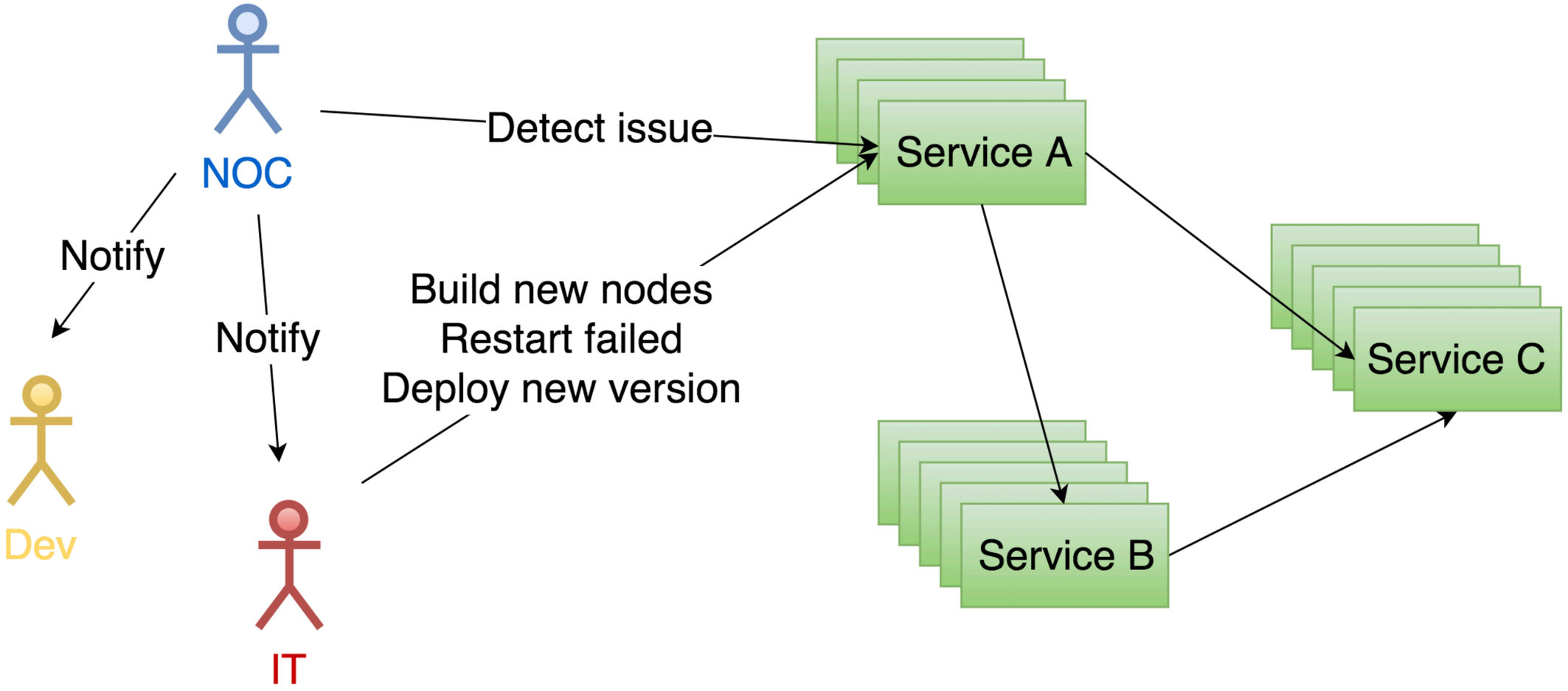
~250k+
simultaneous players

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

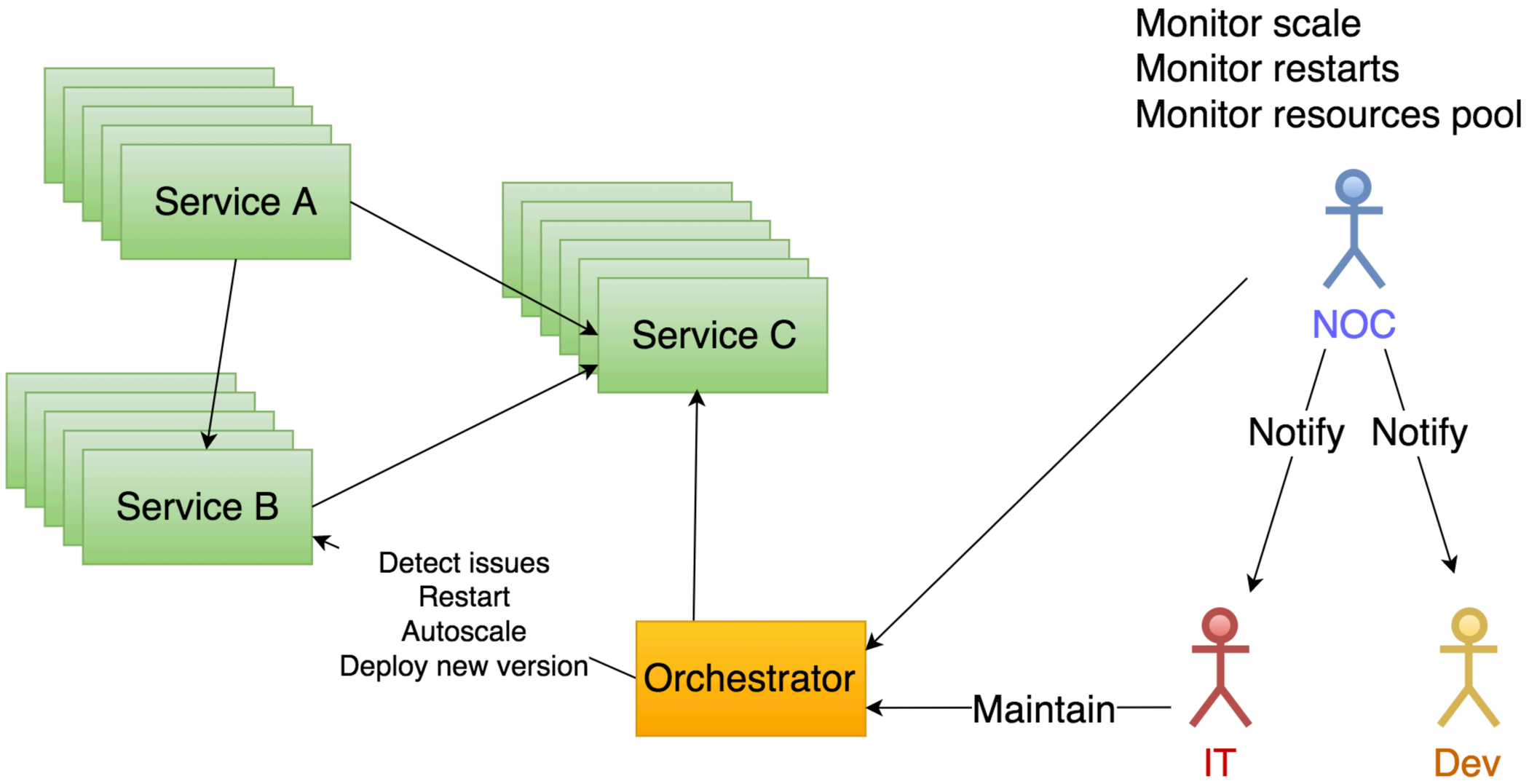
Technology stack

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

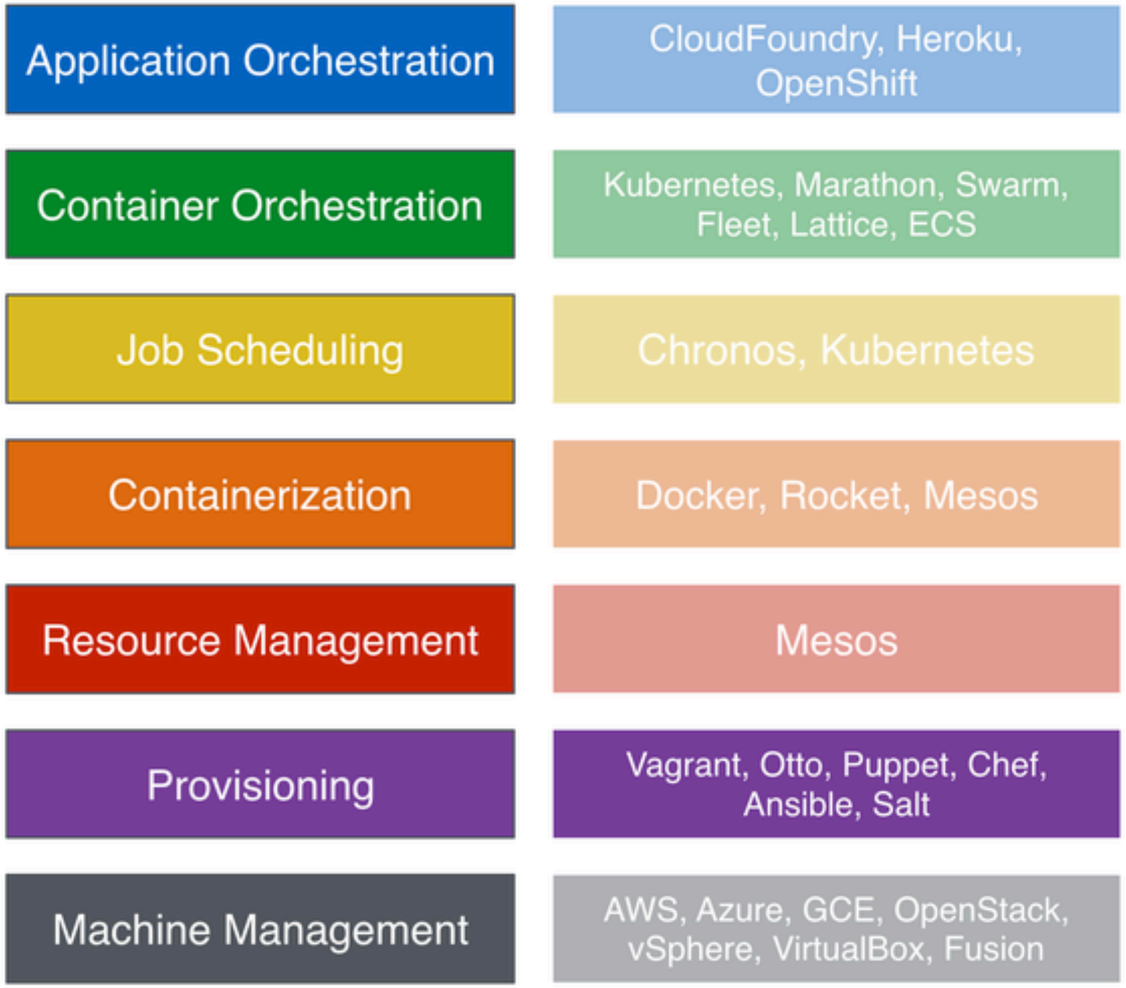
Microservices, first iteration



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

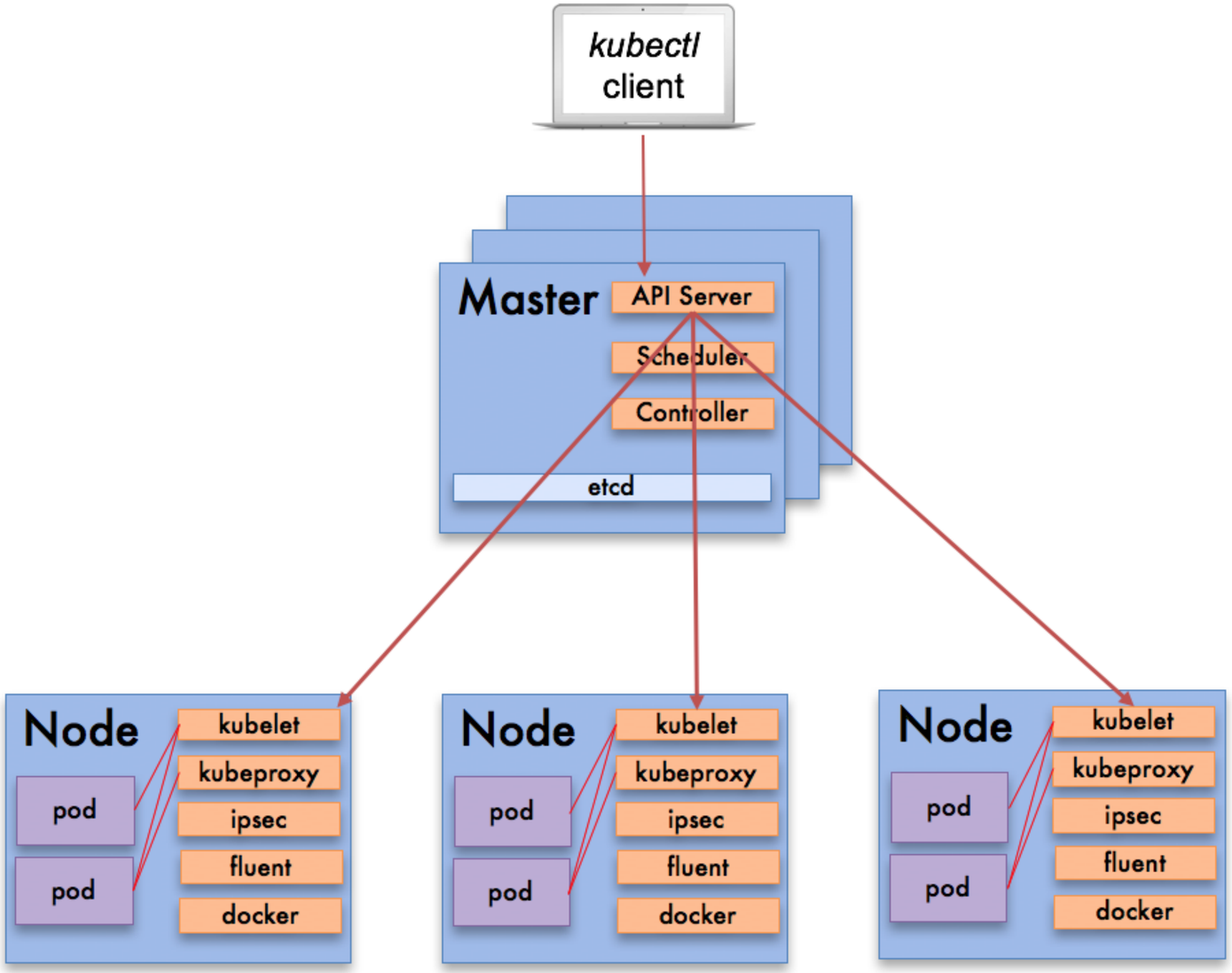
Container orchestration

	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

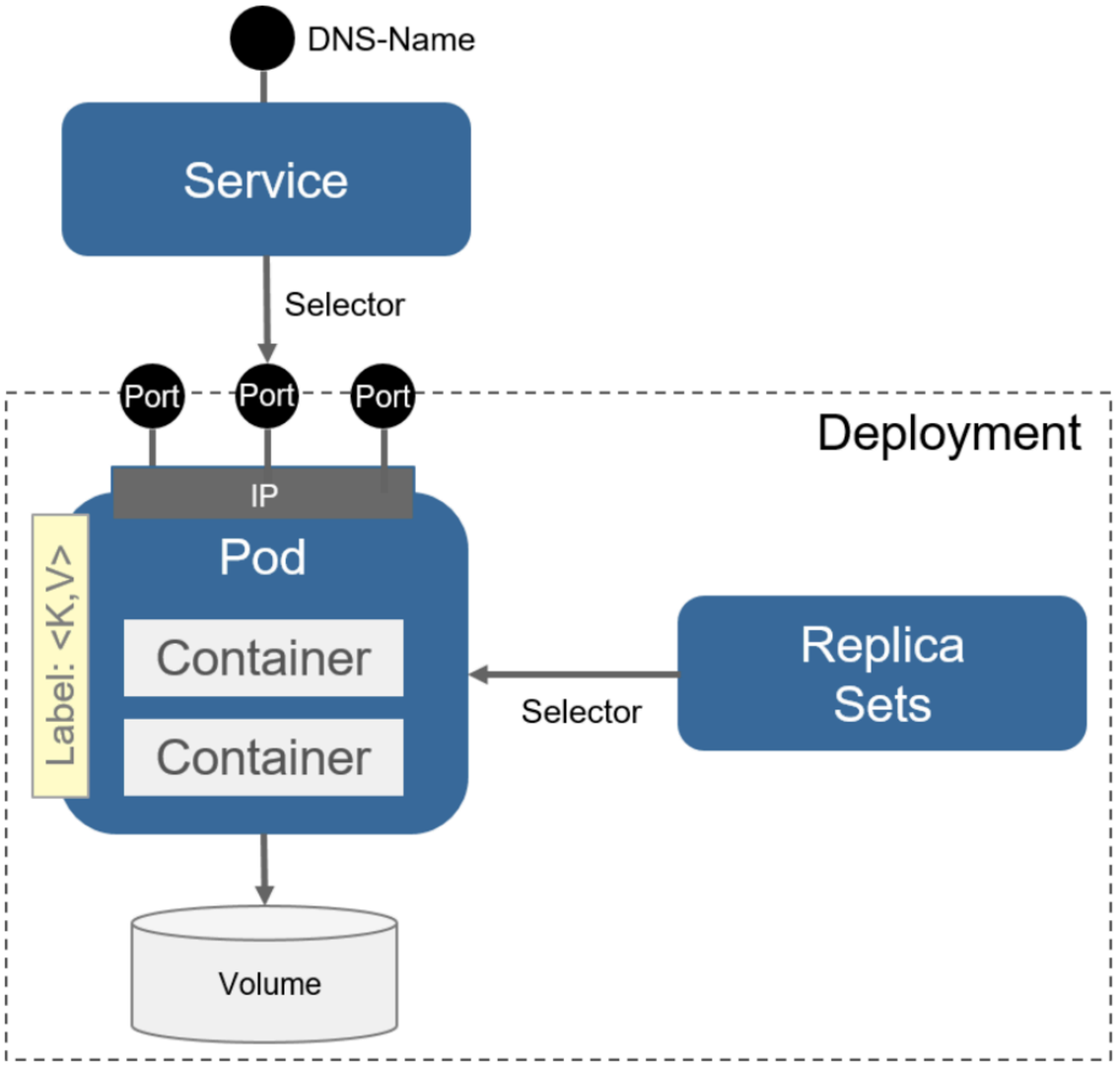
What is Kubernetes?



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



https://www.slideshare.net/MarioLeanderReimer/kubernetes-101-and-fun

Kubernetes deployment

```yaml
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

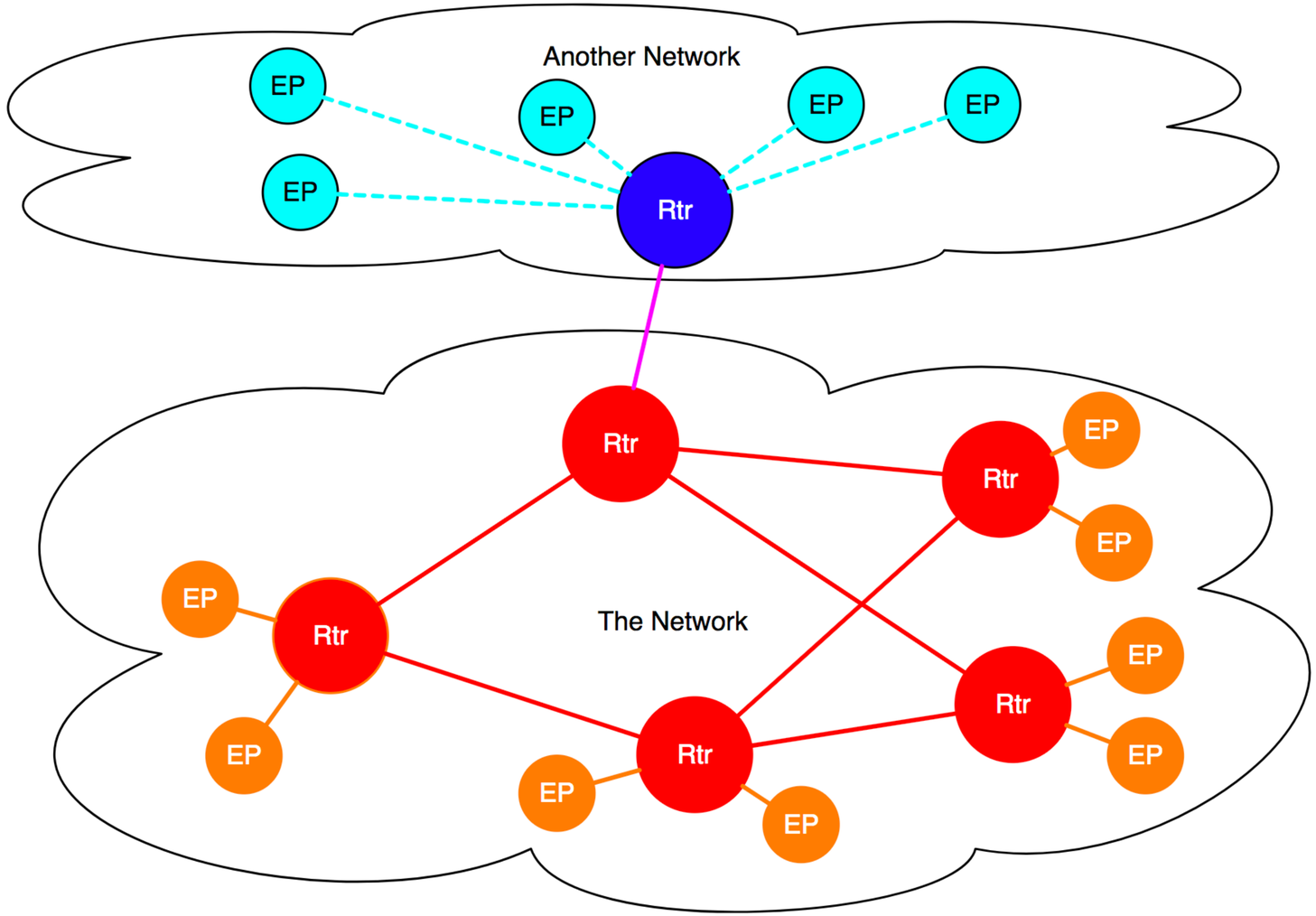
https://www.slideshare.net/MarioLeanderReimer/kubernetes-101-and-fun

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



https://ramitsurana.gitbooks.io/awesome-kubernetes/content/#networking

Kubernetes vs Spring Cloud Netflix

- Failover for service nodes
- Maintaining state of cluster
- Autoscaling
- Deployment automation
- ~~Service discovery~~
- ~~Load balancing~~
- ~~API gateway~~
- ~~Config maps~~

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

- Service discovery
- Client side load balancing
- Circuit breaking
- API gateway
- Configuration service

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

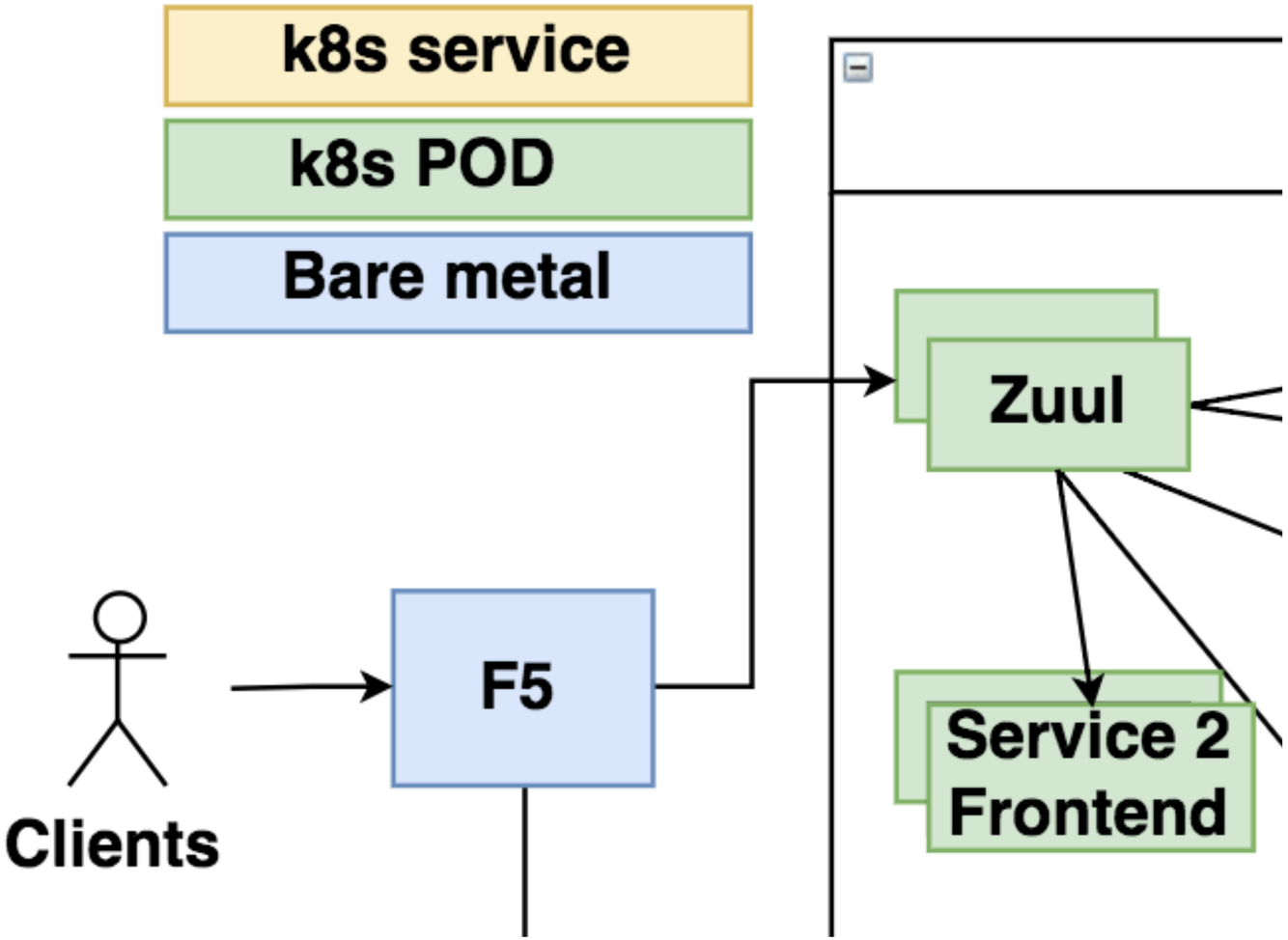
Gotchas - Docker engine crashes

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

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

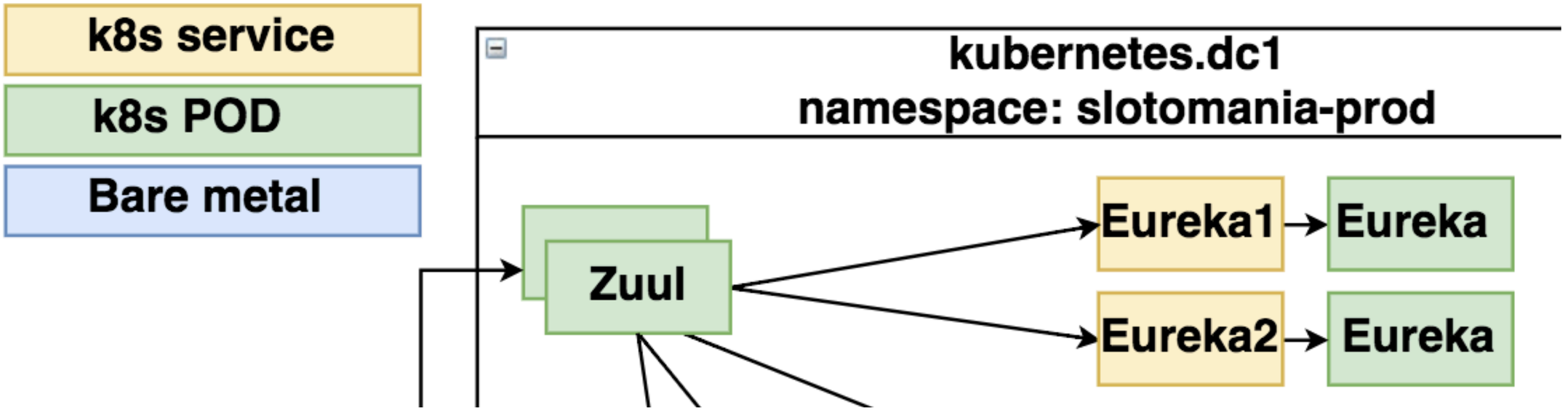
https://github.com/apprenda/kismatic/issues/855

F5 Load Balancer BIG-IP Controller



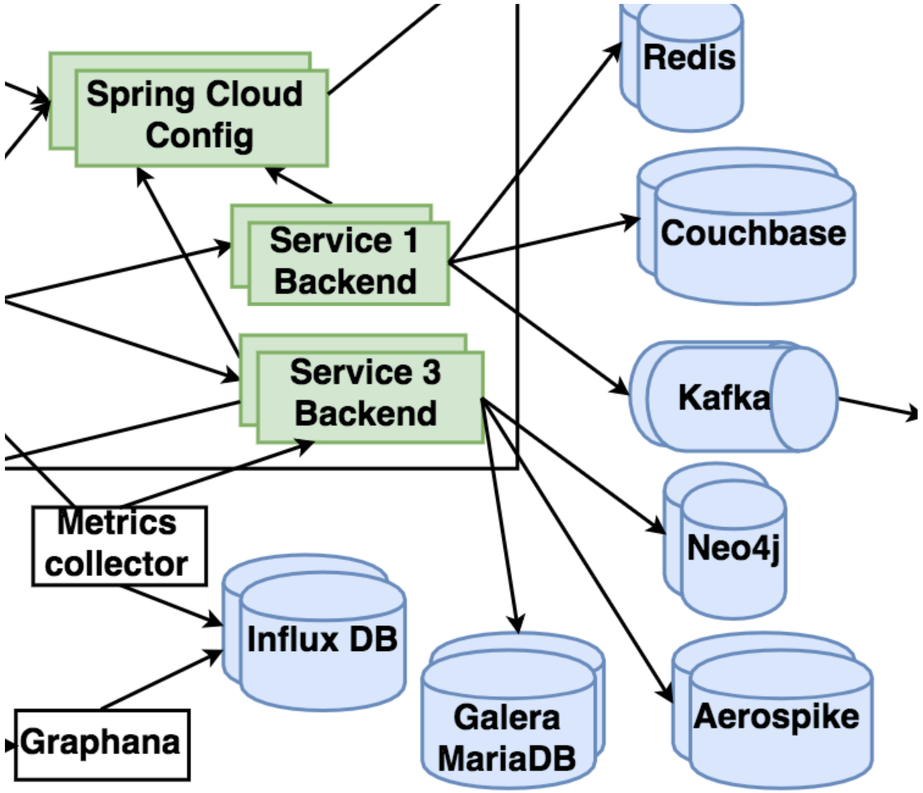
http://clouddocs.f5.com/containers/v2/kubernetes/

Eureka stateful service



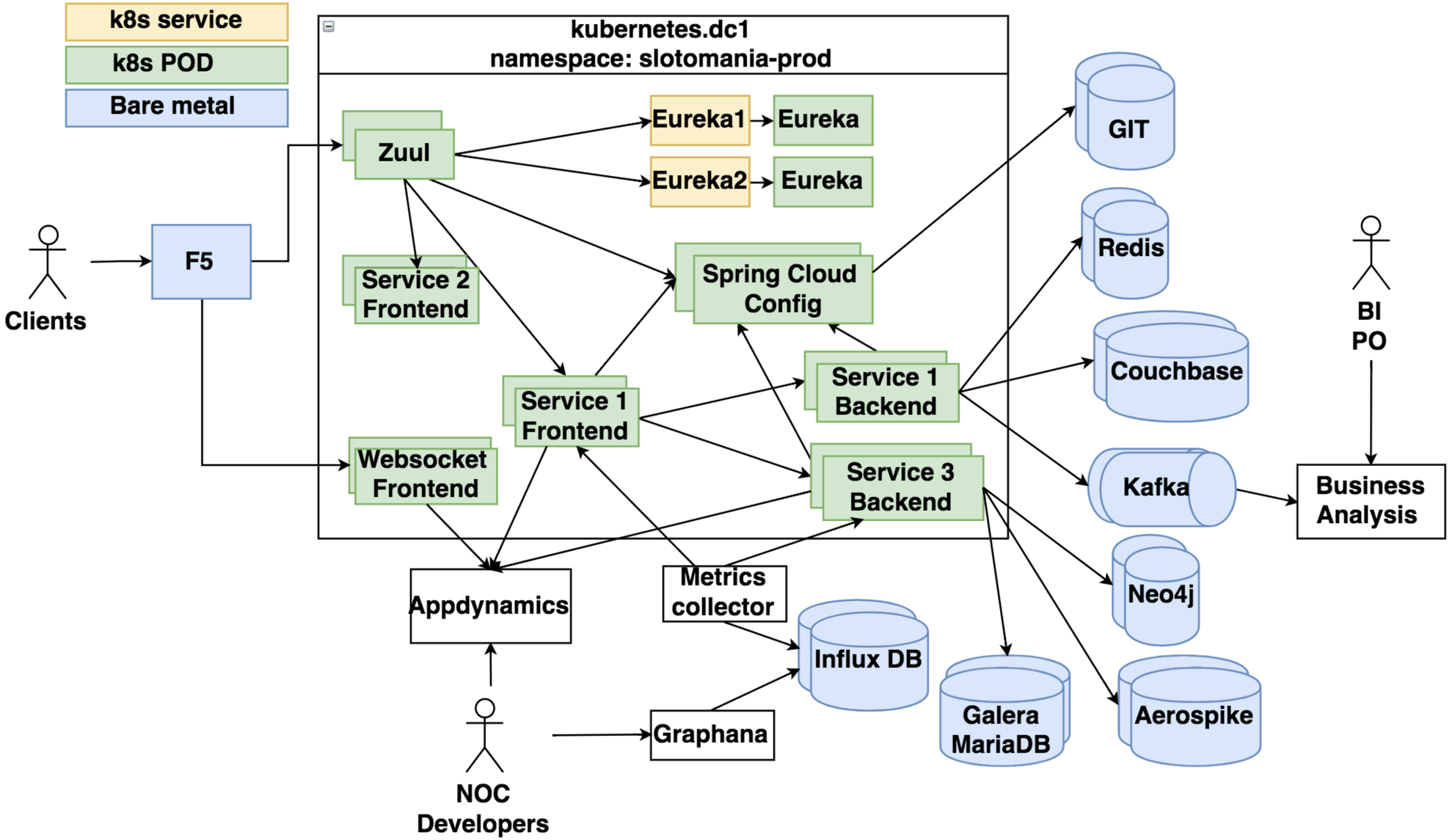
https://stackoverflow.com/questions/40567429/eureka-and-kubernetes

Databases still bare metal

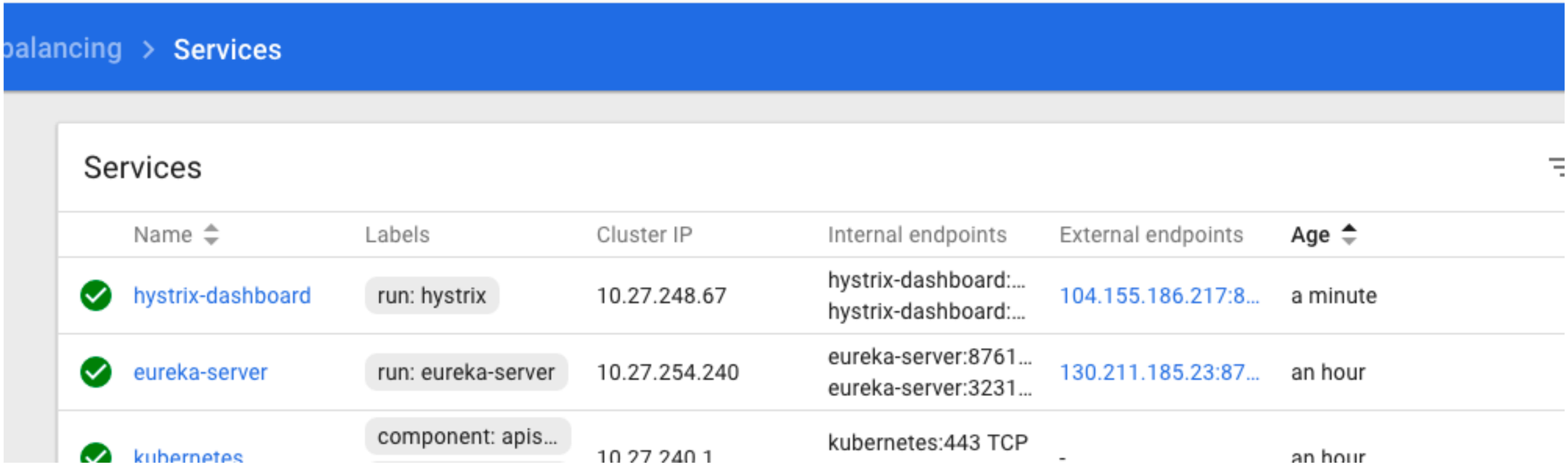


[Should database live inside or outside the container]

Bird eye view



Try it at home



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 JOPTS="-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

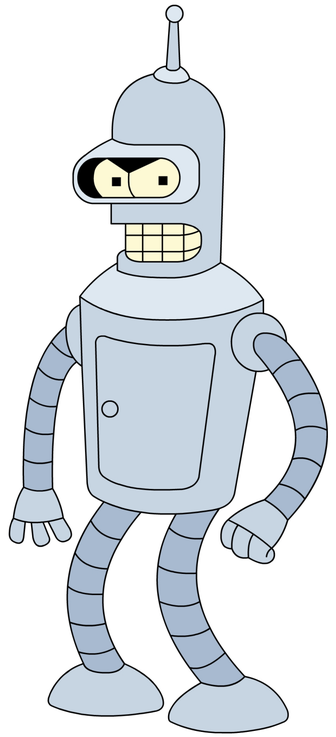
Java support for docker cpu and memory limits

Lets talk about QA

Deployment automation

Continuous delivery
Each commit is “deployable”

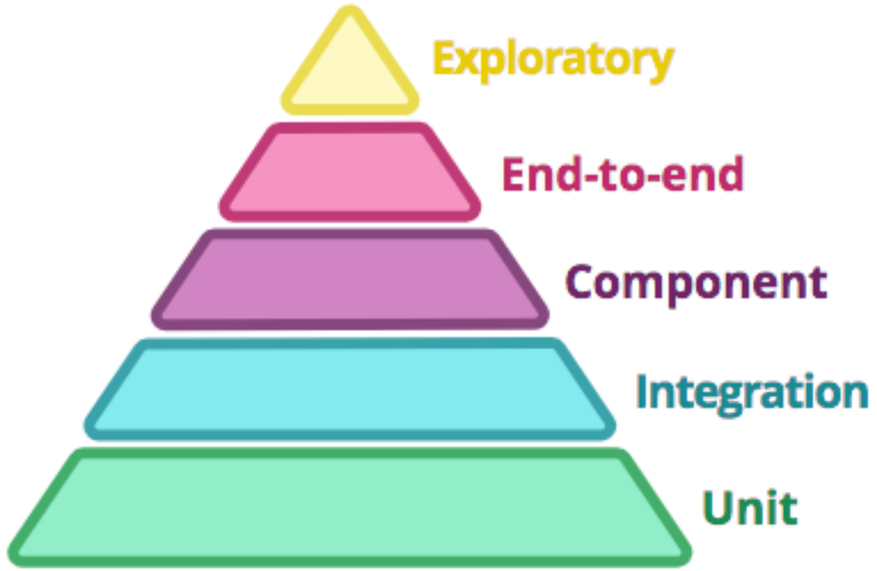
Continuous deployment
Each commit is “deployed”



https://dzone.com/articles/continuous-delivery-vs-continuous-deployment-an-ov

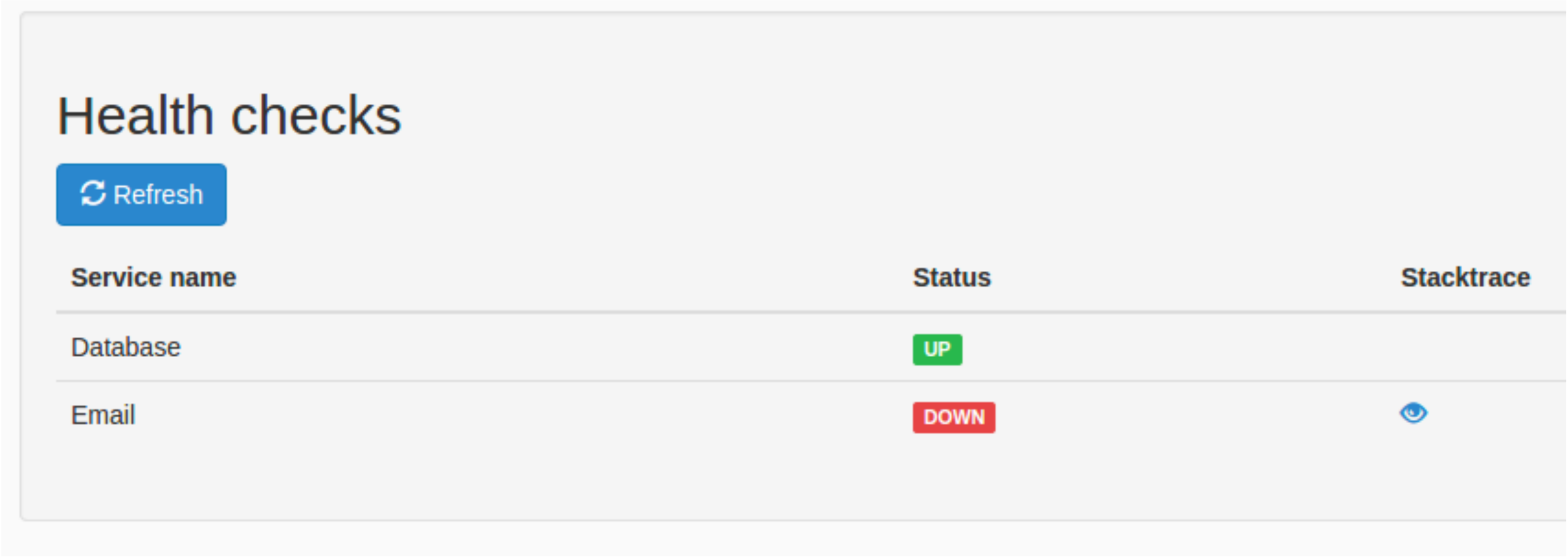
QA strategy for microservices

- 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



[Readiness and liveness probes]

[Spring boot health indicators]

Lets talk about deployment

Version release

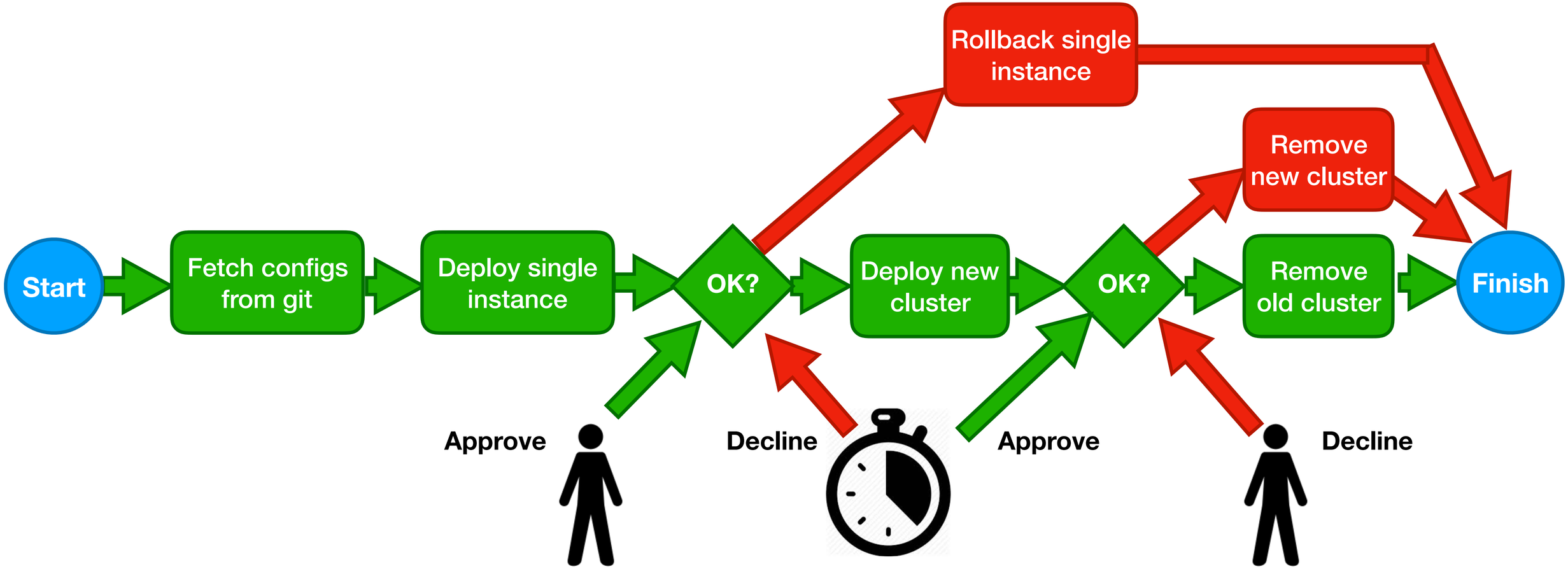
https://www.thoughtworks.com/radar/techniques/automated-deployment-pipeline

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

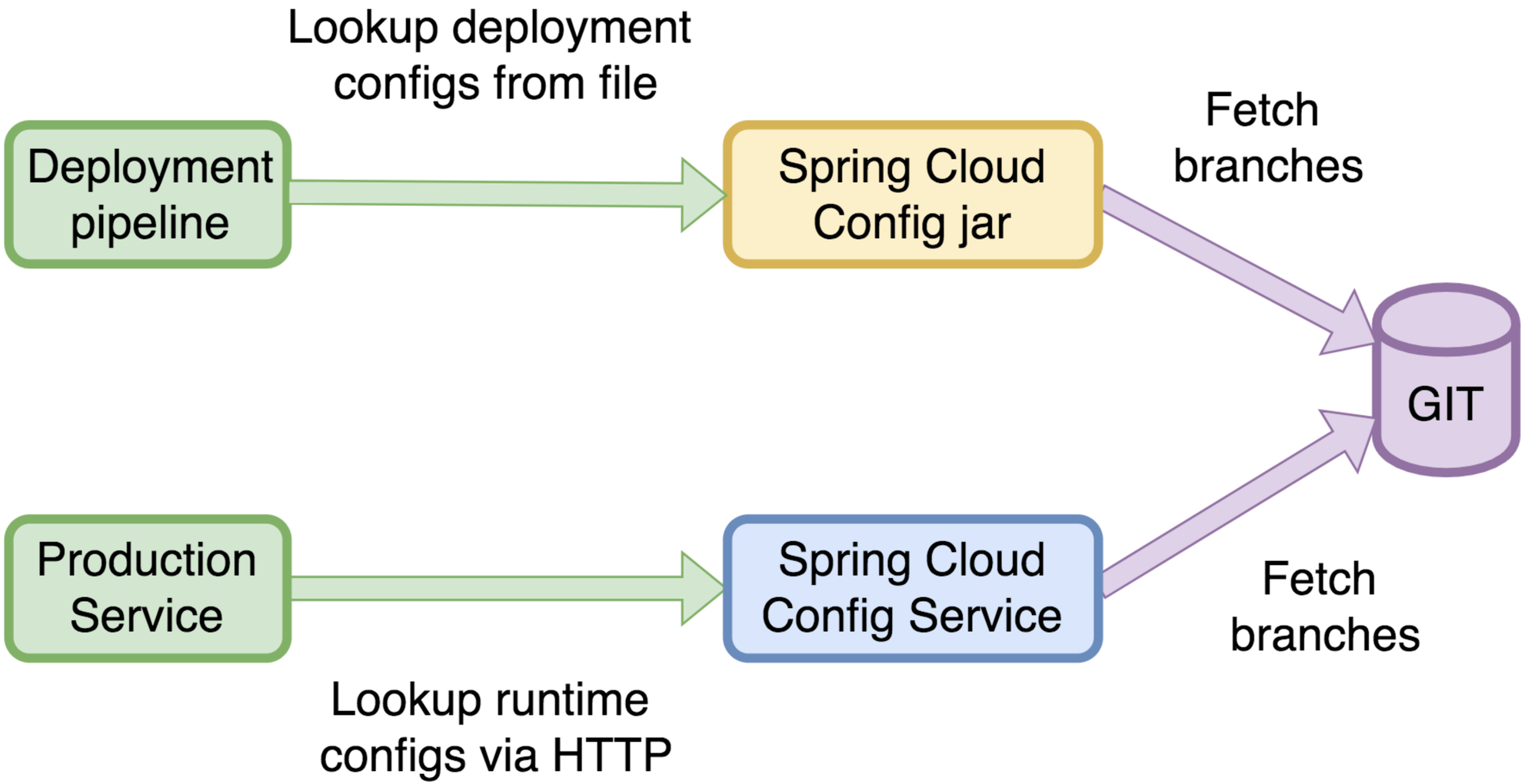


https://ramitsurana.gitbooks.io/awesome-kubernetes/content/#continuous-delivery

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

UpgradeServicePipeline.groovy

```groovy
package com.playtika.deployment.jenkins.pipelines

...

node() {

    sendInProgressEmail()

    lock( resource: "EnvironmentName: ${EnvironmentName}") {
        try {

            stage( name: 'Upgrade Service: Get service name by branch') {...}

            stage( name: 'Upgrade Service: Trigger deployment pipeline') {...}

            stage( name: 'Send Report') {sendResultingEmail( status: 'SUCCESS', reportEntriesHolder)}
        }
        catch (Exception e) {
            echo message: "Caught error while pipeline execution: ${e}"
            sendResultingEmail( status: 'FAILURE', reportEntriesHolder)
            currentBuild.result = 'FAILURE'
        }
    }
}
```

https://jenkins.io/doc/book/pipeline/

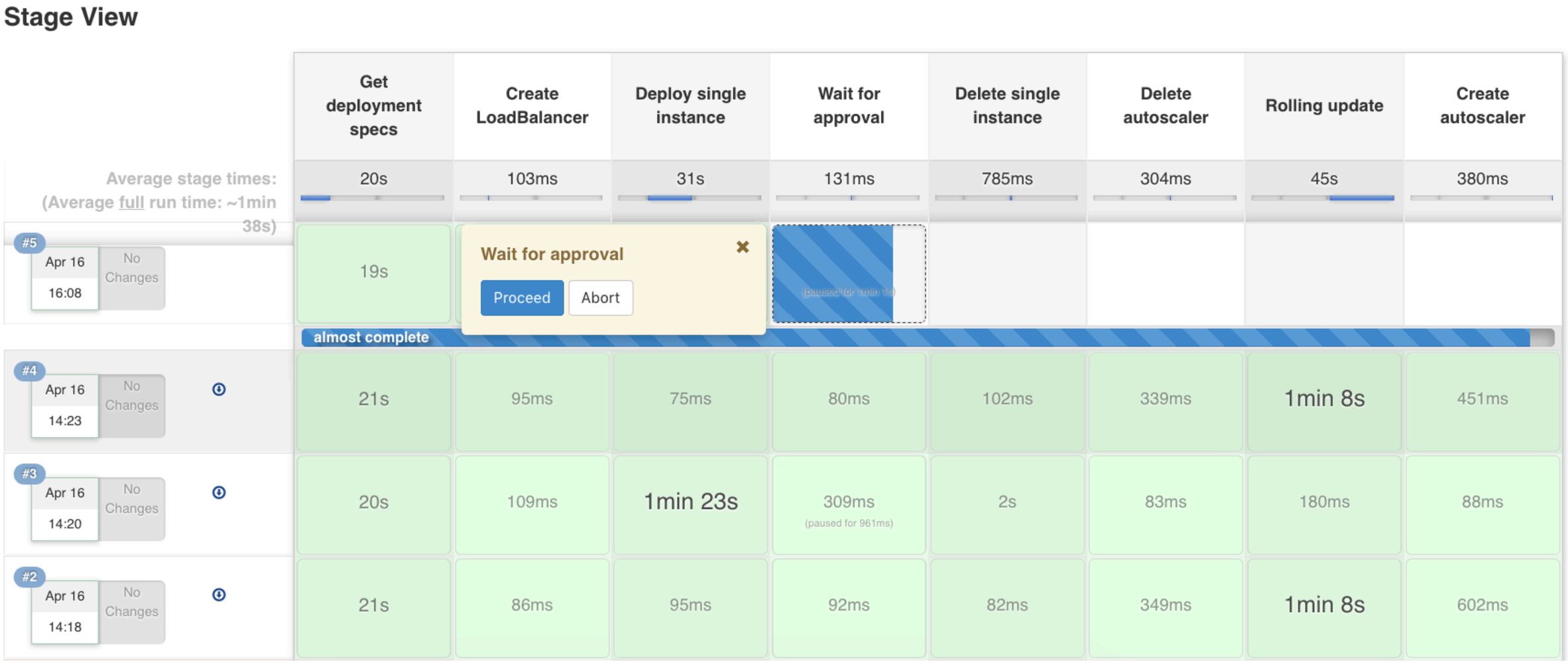
Jenkins job DSL

CreateDeploymentAutoscalerJob.groovy

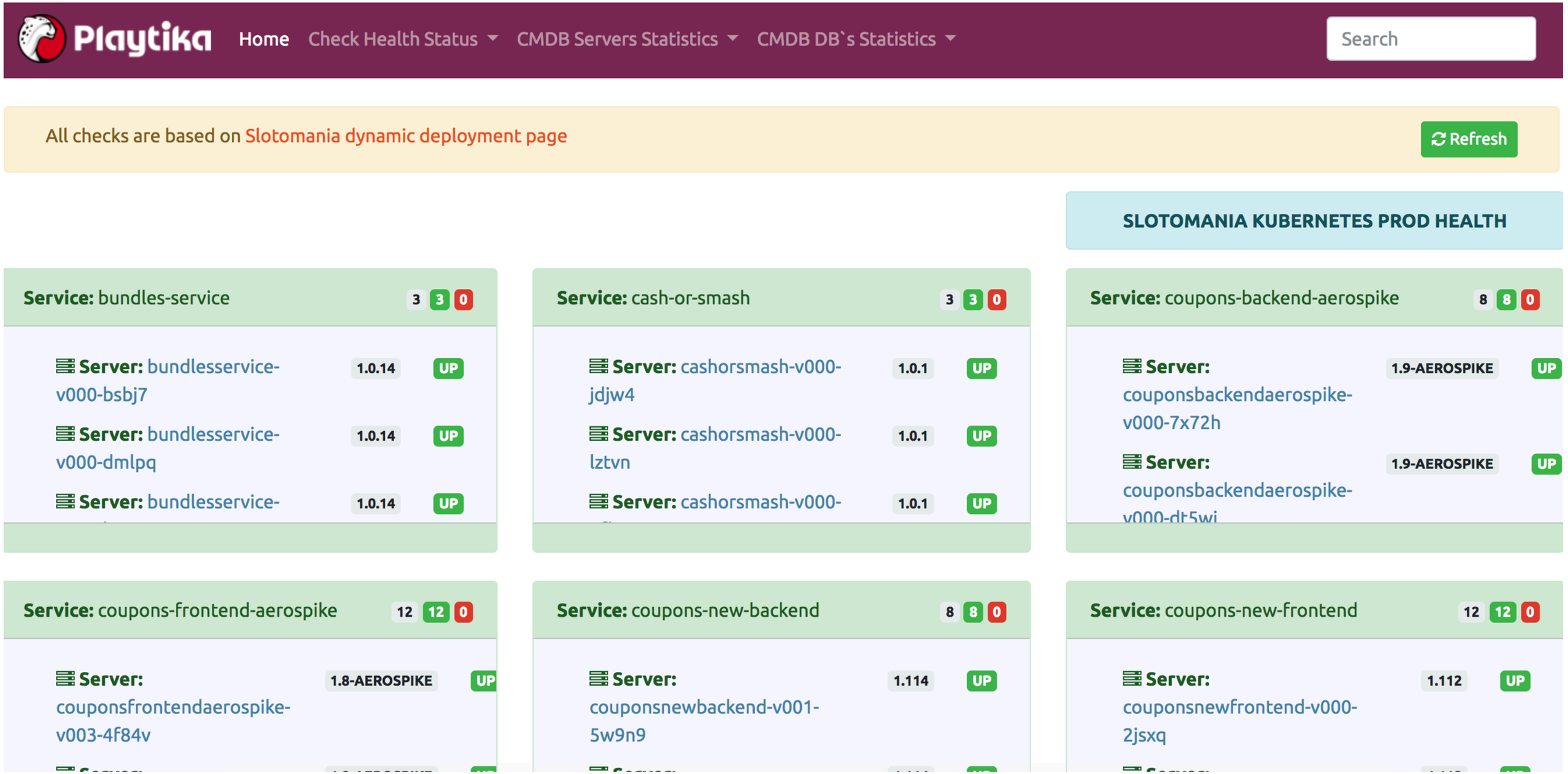
```groovy
package com.playtika.deployment.jenkins.jobs

String parentFolder = 'deployService'

folder(parentFolder)

job( s: "$parentFolder/CreateDeploymentAutoscaler") {
    parameters {...}
    properties {
        buildFailureAnalyzer()
    }
    concurrentBuild( allowConcurrentBuild: true)
    logRotator {...}
    wrappers {...}
    lockableResources {...}
    steps {
        buildDescription regexp: '', description: '[serviceName: $serviceName, namespace: $namespace]'
        shell """
        echo "Job parameter: namespace=\${namespace}"
        echo "Job parameter: serviceName=\${serviceName}"
        echo "Job parameter: teamName=\${teamName}"
        echo "Job parameter: parentJobId=\${parentJobId}"
        echo "Job parameter: deploymentId=\${deploymentId}"

        sanitizedServiceName=\$(echo "\${serviceName}" | sed -e 's/[^A-Za-z0-9]//g' | tr '[:upper:]' '[:lower:]')
        kubectl -n "\$namespace" rollout status "deploy/\$sanitizedServiceName"
        kubectl -n "\$namespace" autoscale "deploy/\$sanitizedServiceName" --min="\$min" --max="\$max" --cpu-percent="\$cpu"
        """.stripIndent().trim()
    }
}
```

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

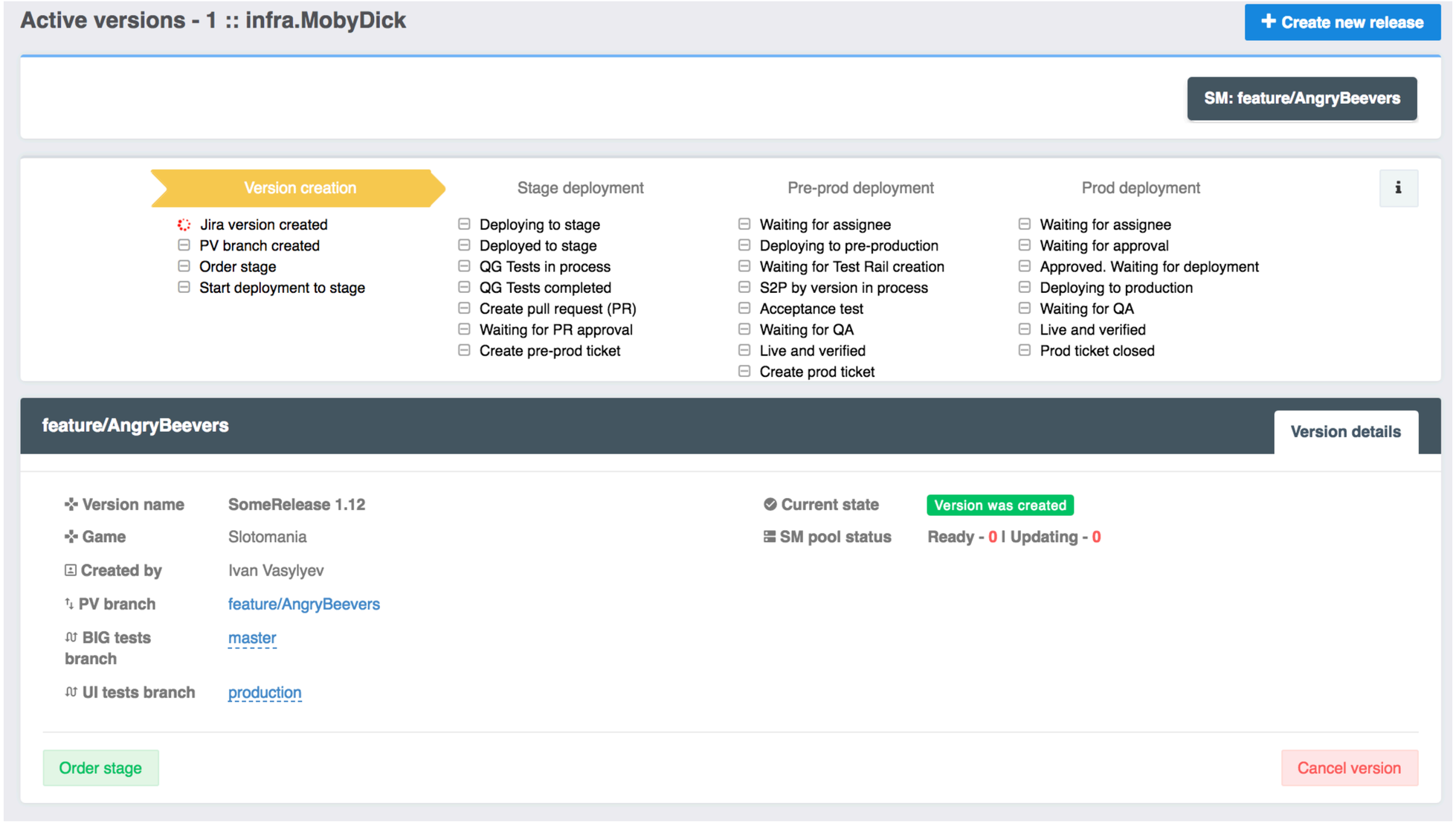
Canary+rolling Jenkins pipeline



Namespace overview page



Version release page



Summary

- Kubernetes is wining 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