Resilient Microservices with Kubernetes

Mete Atamel
Developer Advocate at Google

@meteatamel
atamel@google.com

Google Cloud Platform

In the good old days

Image Source: Wikipedia

A lot happened since then

Internet

Databases

Web Servers

The Monolith

Mobile

App Servers

Microservices

Cloud Computing

Firewalls

Version Control

Caching

Machine Learning

Object Oriented Programming

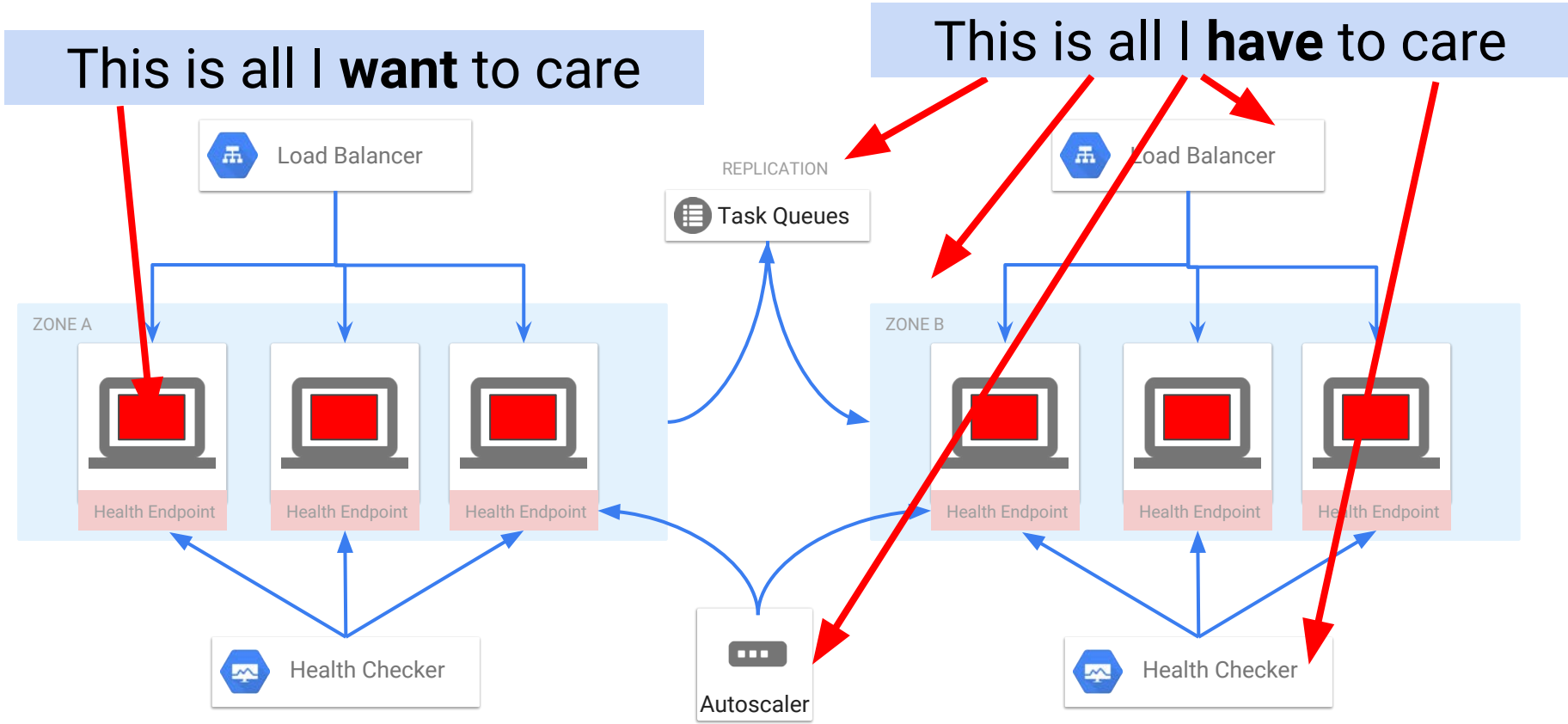
DevOps

Virtual Machines

Big Data

IoT

Nowadays


This is all I **want** to care
This is all I **have** to care
Load Balancer
REPLICATION
Task Queues
Load Balancer
ZONE A
ZONE B
Health Endpoint
Health Endpoint
Health Endpoint
Health Endpoint
Health Endpoint
Health Endpoint
Health Checker
Autoscaler
Health Checker

Write your code in any language and run anywhere exactly the same way

Your app is optimally deployed and managed

All resources are provisioned on demand

Containers + Orcestrators + Cloud

Write your code in any language and run anywhere exactly the same way
⇒ Containers (eg. Docker, Rkt)

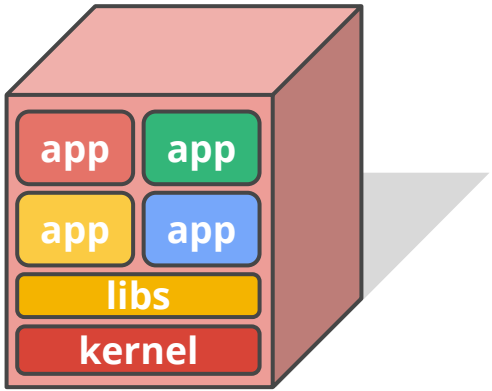
Your app is optimally deployed and managed
⇒ Orchestrators (eg. Kubernetes, Docker Swarm, Mesos)

All resources are provisioned on demand
⇒ Cloud Providers (eg. Google Cloud, AWS, Azure)

Containers

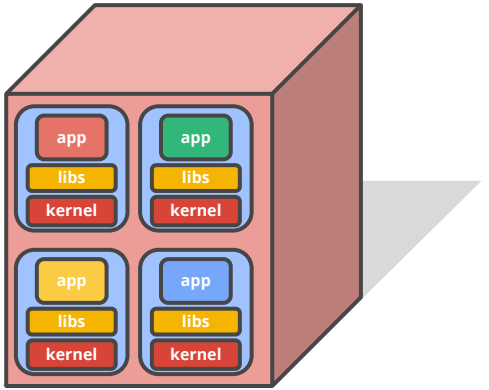
Why containers?

@meteatamel



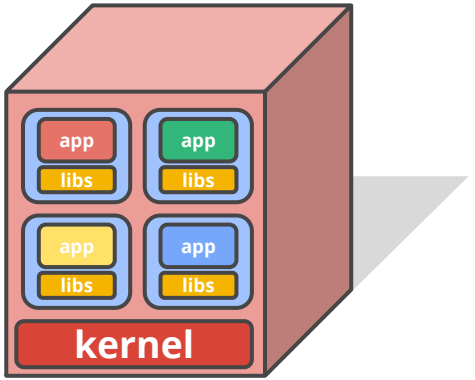
Physical Machine

- ✗ No isolation
- ✗ Common libs
- ✗ Highly coupled Apps & OS



Virtual Machines

- ✓ Isolation
- ✓ No Common Libs
- ✗ Expensive and Inefficient
- ✗ Hard to manage



Containers

- ✓ Isolation
- ✓ No Common Libs
- ✓ Less overhead
- ✗ Less Dependency on Host OS

What is a container?

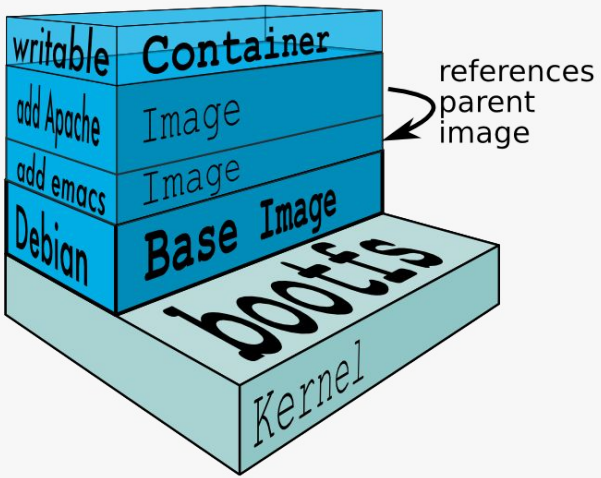
Docker

rkt

A **lightweight** way to virtualize applications

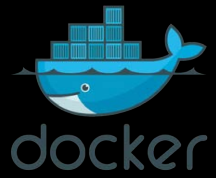
Linux (or Windows) processes

Lightweight	Easily deployable
Hermetically sealed	Introspectable
Isolated	Composable



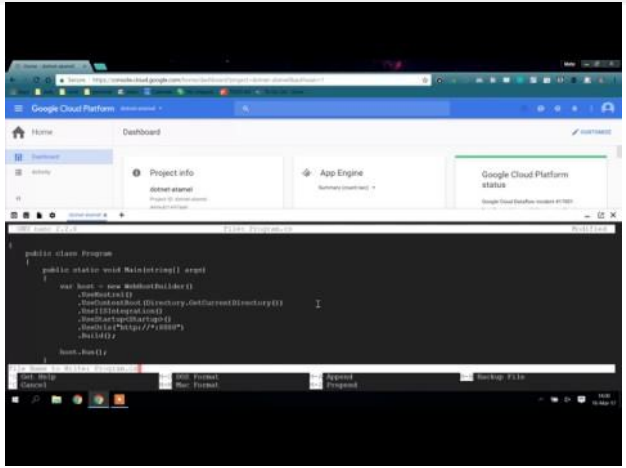
Docker: tooling for the masses

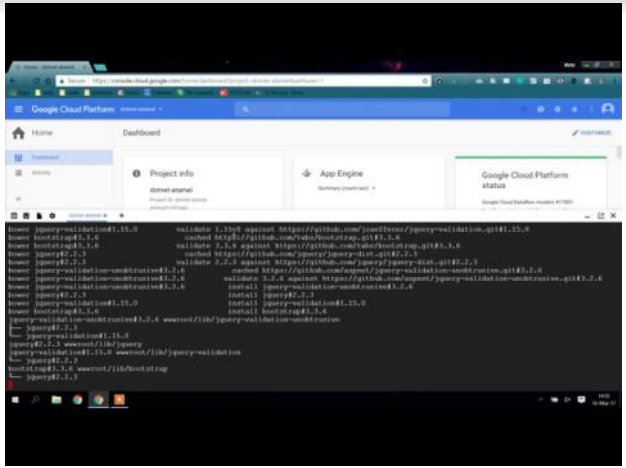
- Docker is the most popular (but not only) container runtime and image format
  - A Dockerfile is a manifest defining what the container does and how it communicates
  - A container is an invocation of a Dockerfile
- A container looks and feels just like a regular operating system – the application doesn't know it's running inside a container
- A container keeps track of (usually) one process, and exposes it to the outside world



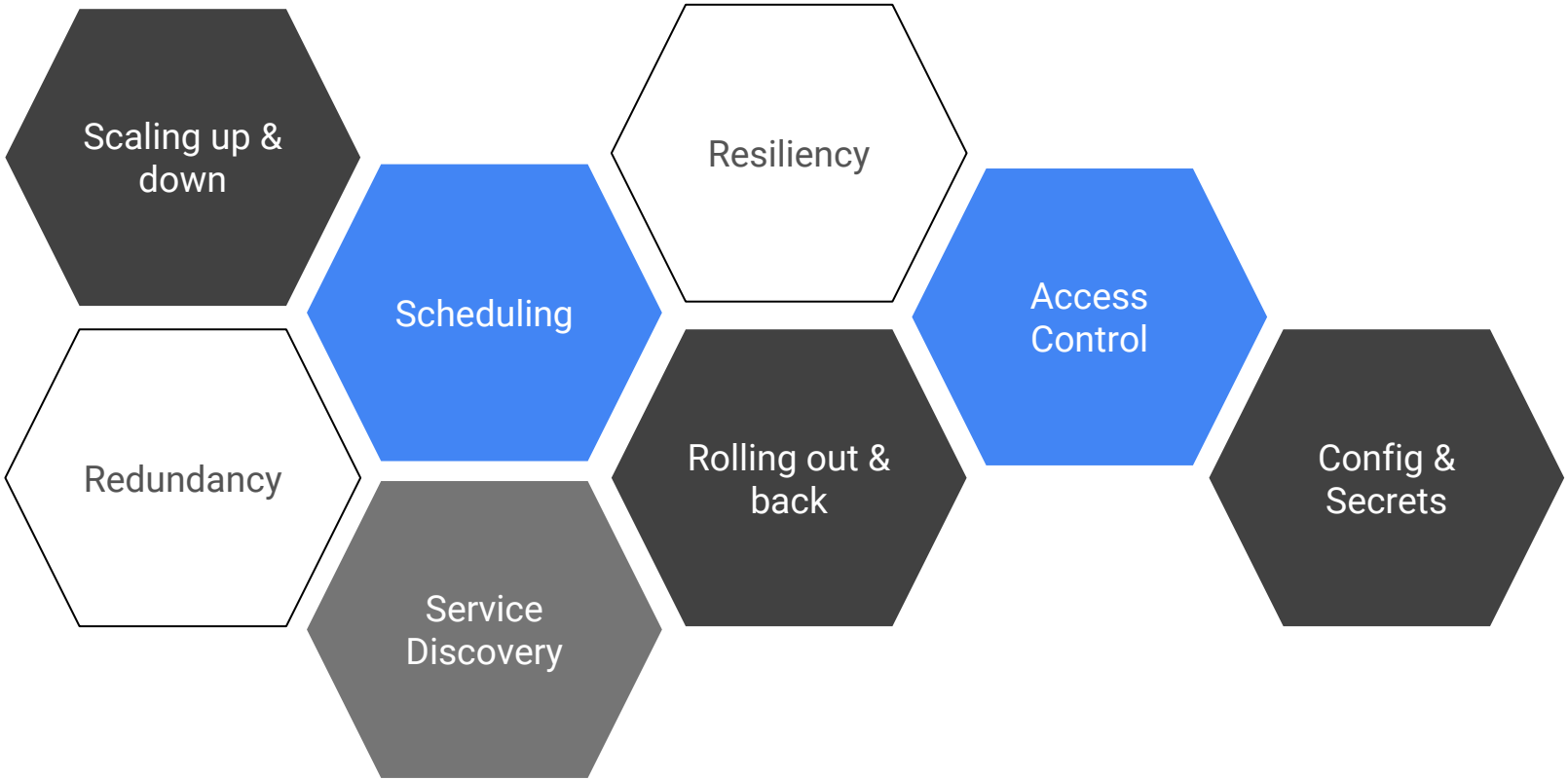

```
FROM debian:latest

RUN apt-get update
RUN apt-get install -y nginx

CMD [“nginx”,”-g”,”daemon off;”]

EXPOSE 80
```

Demo: Containerised App





Containers are not enough



Kubernetes

Kubernetes

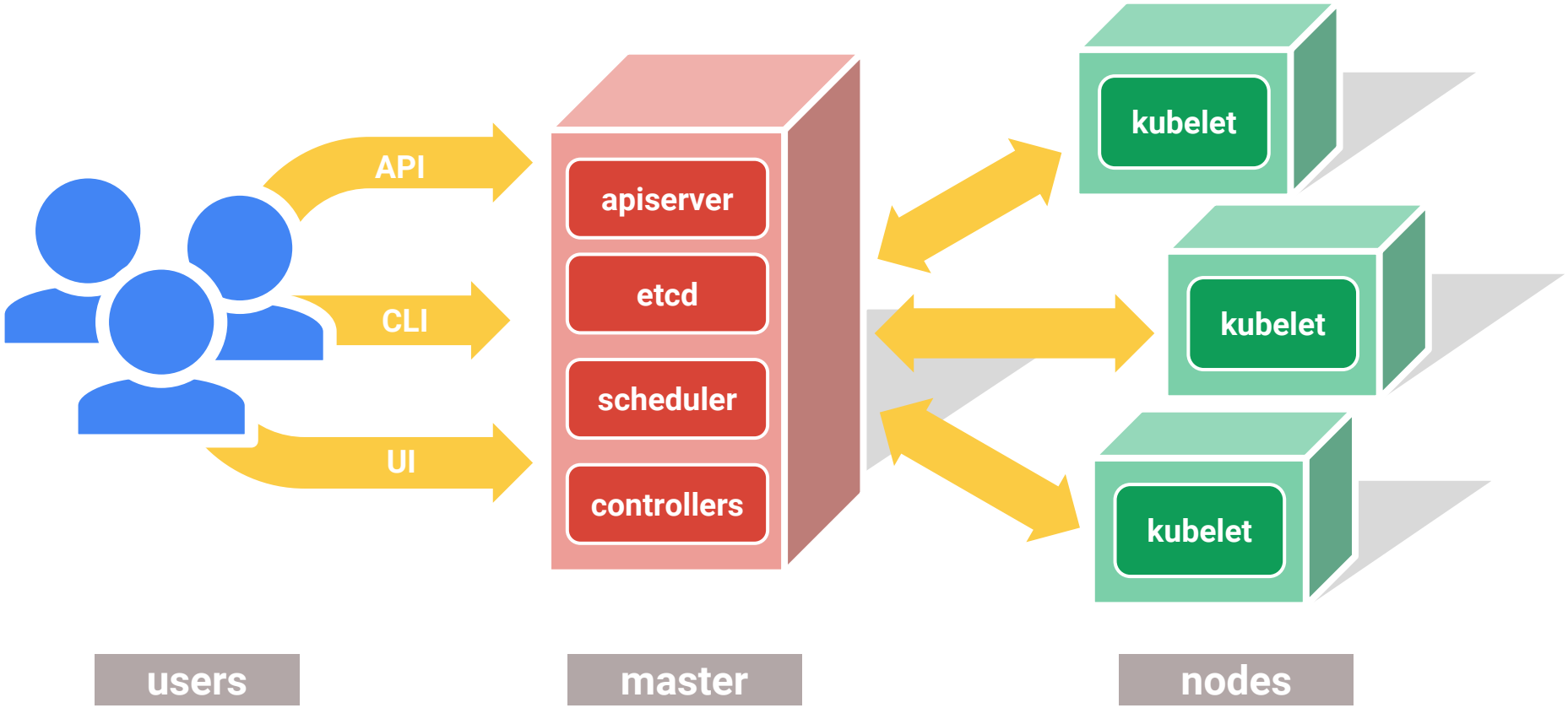
Greek for *“Helmsman”*; also the root of the words *“governor”* and *“cybernetic”*

- Manages container clusters
- Inspired and informed by Google’s experiences and internal systems (borg)
- Supports multiple cloud and bare-metal environments
- Supports multiple container runtimes
- **100% Open source**, written in Go

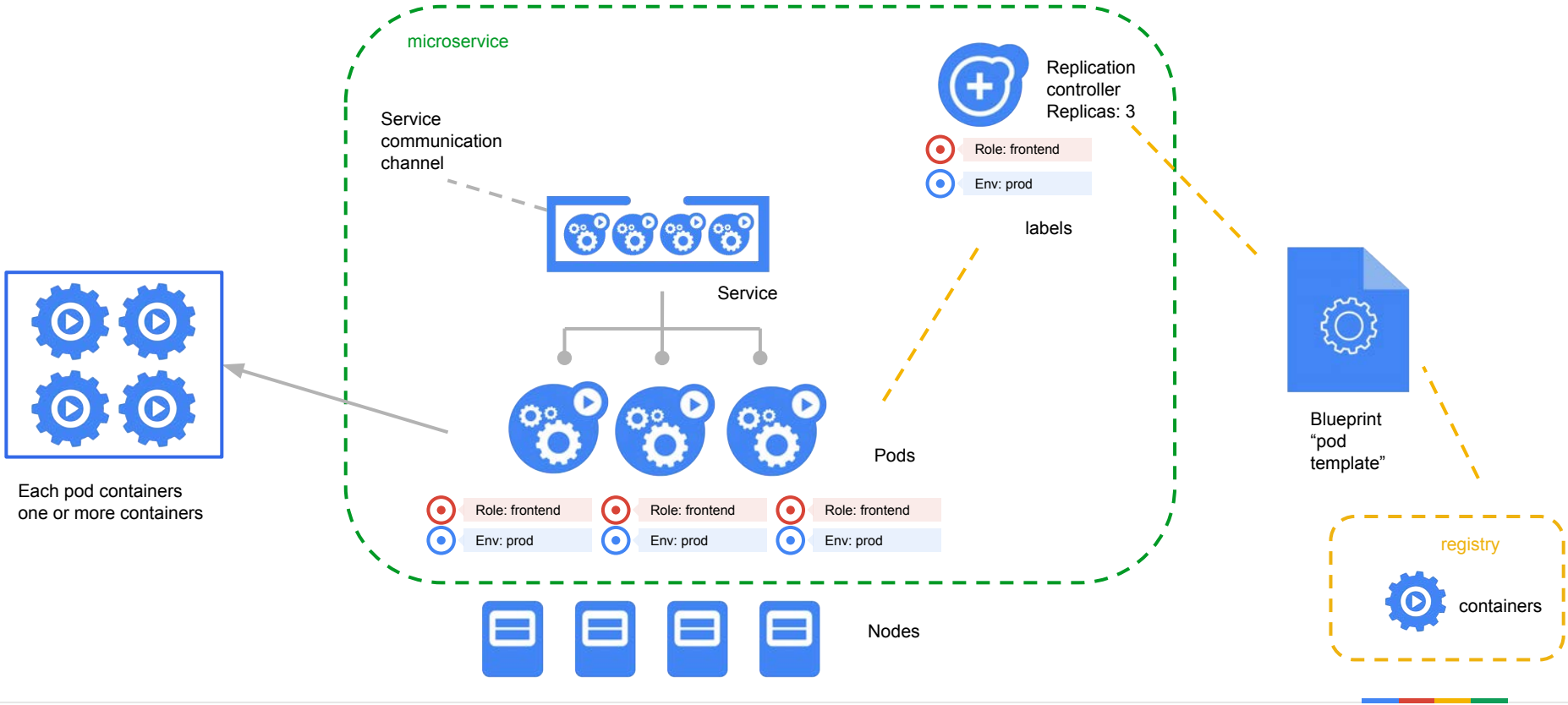
Manage applications, not machines

The 10000 foot view

@meteatamel



Kubernetes cluster



Google Container Engine (GKE)

Cloud Functions
A serverless platform for event-based microservices

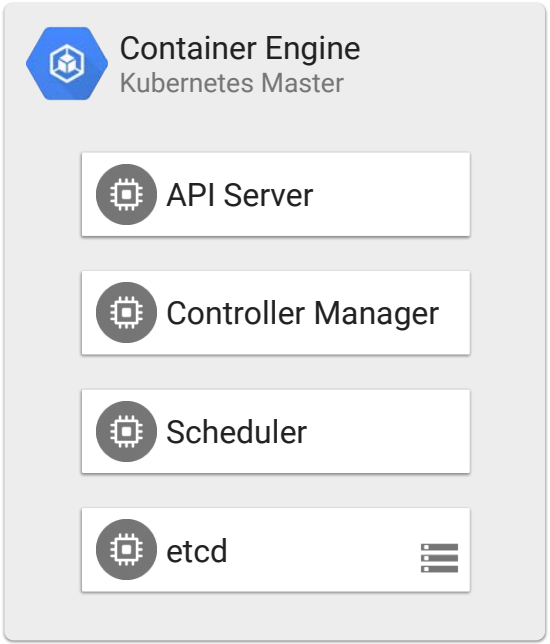
App Engine
Deploy your code and we scale it for you

Google Container Engine (GKE)
Kubernetes-as-a-service

Compute Engine
Full control: VMs for Linux and Windows Server

```
$ gcloud container clusters create cluster-1
Creating cluster cluster-1...done.
Created [https://container.googleapis.com/v1/projects/sandbox/zones/europe-west1-c/clusters/cluster-1].
kubeconfig entry generated for cluster-1.
NAME       ZONE            MASTER_VERSION  MASTER_IP       MACHINE_TYPE   NODE_VERSION  NUM_NODES  STATUS
cluster-1  europe-west1-c  1.4.6           104.199.87.107  n1-standard-1  1.4.6         3          RUNNING

$ gcloud container clusters get-credentials cluster-1
Fetching cluster endpoint and auth data.
kubeconfig entry generated for cluster-1.

$ kubectl get nodes
NAME                                        STATUS    AGE
gke-cluster-1-default-pool-6c50430d-chjm    Ready     2m
gke-cluster-1-default-pool-6c50430d-esqq    Ready     2m
gke-cluster-1-default-pool-6c50430d-zfm9    Ready     2m

$ kubectl get pods
$

$ gcloud container clusters resize cluster-1 --size 5
Pool [default-pool] for [cluster-1] will be resized to 5.
Resizing cluster-1...done.
Updated [https://container.googleapis.com/v1/projects/sandbox/zones/europe-west1-c/clusters/cluster-1].
```

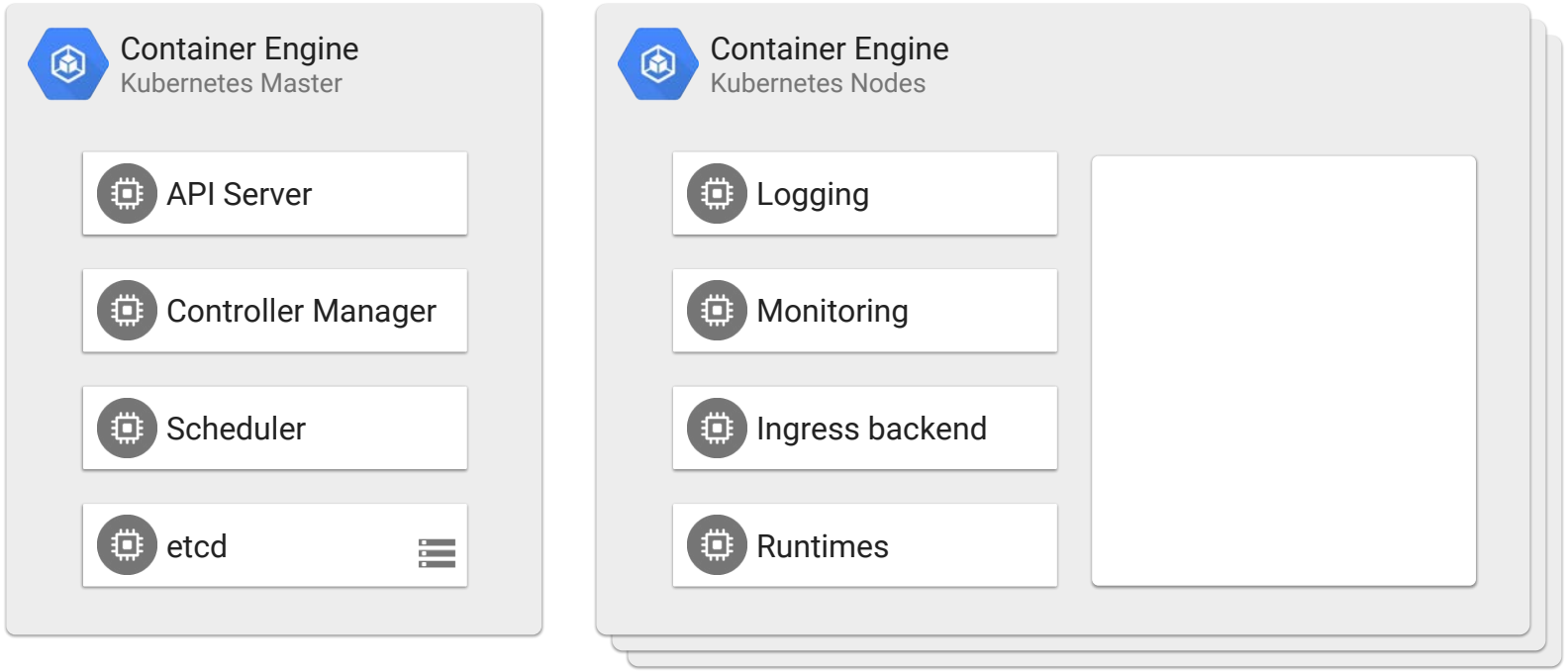
Google manages your control plane



- Backups
- Monitoring
- Restarts
- Resizing for larger clusters

- **99.5% SLA**

...and components on your nodes



Google Container Registry

- Private Docker Container Images
  - Secure, fast, cheap
  - Continuous Delivery integrations
    - CircleCI, Codeship, Drone, Jenkins, Solano CI, Spinnaker, Shippable, Wercker

```
docker tag user/image gcr.io/project-id/image

gcloud docker -- push gcr.io/project-id/image

gcloud docker -- pull gcr.io/project-id/image
```

Google Container Registry

Google Cloud Container Builder

- Fully managed build service
- Build Triggers
  - Integrations with Github and Bitbucket
- Custom builds
  - Custom build scripts
  - Custom container images
- Integrated with
  - Container Registry integration
  - Stackdriver Logging

Google Cloud
Container Builder

Kubernetes Building Blocks

Kubernetes Terminology

Deployment

Pod

Volume

Label

Selector

ReplicaSet

Liveness Probe

Readiness Probe

Service

DaemonSet

Job

StatefulSet

ConfigMap

Secret

Deployment

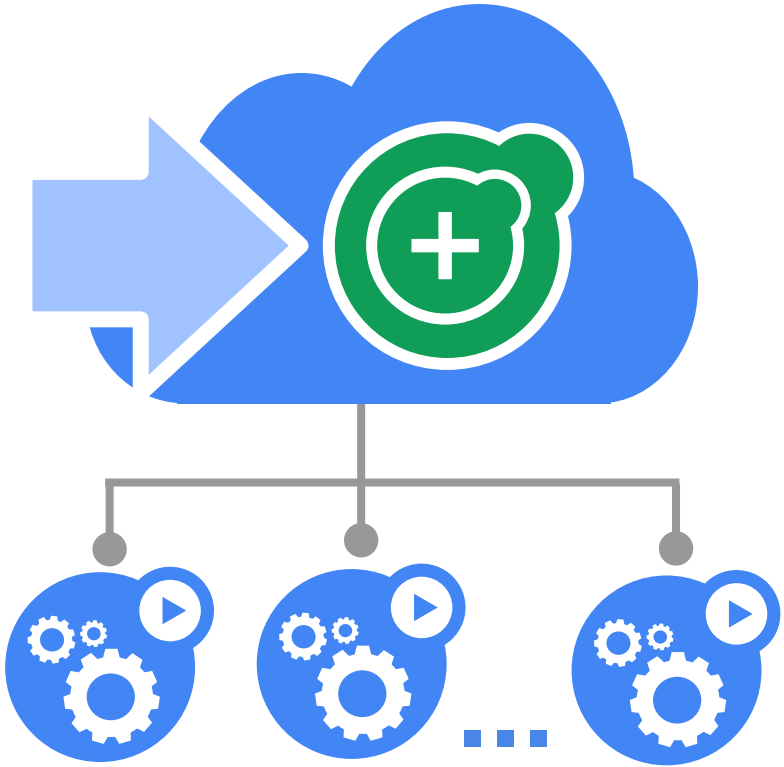
@meteatamel

A Deployment provides declarative updates for Pods and Replica Sets

Describe the desired state and the Deployment controller will change the actual state to the desired state at a controlled rate for you.

Deployment manages replica changes for you

- stable object name
- updates are configurable, done server-side
- `kubectl edit` or `kubectl apply`



Pod

Small group of containers & volumes

Tightly coupled

The atom of scheduling & placement

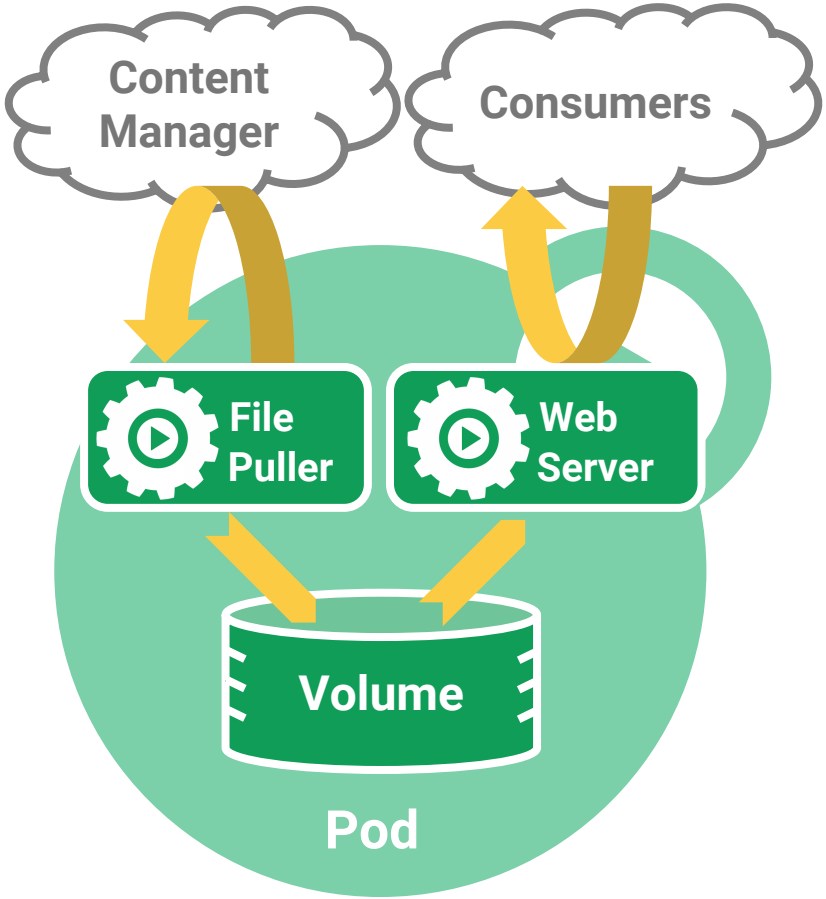
Shared namespace

- share IP address & localhost
- share IPC, etc.

Managed lifecycle

- bound to a node, restart in place
- can die, cannot be reborn with same ID

Example: data puller & web server



Volume

Pod-scoped storage

Support many types of volume plugins

- Empty dir (and tmpfs)
- Host path
- Git repository
- GCE Persistent Disk
- AWS Elastic Block Store
- Azure File Storage
- iSCSI
- Flocker
- NFS
- vSphere
- GlusterFS
- Ceph File and RBD
- Cinder
- FibreChannel
- Secret, ConfigMap, DownwardAPI
- Flex (exec a binary)
- ...

Label

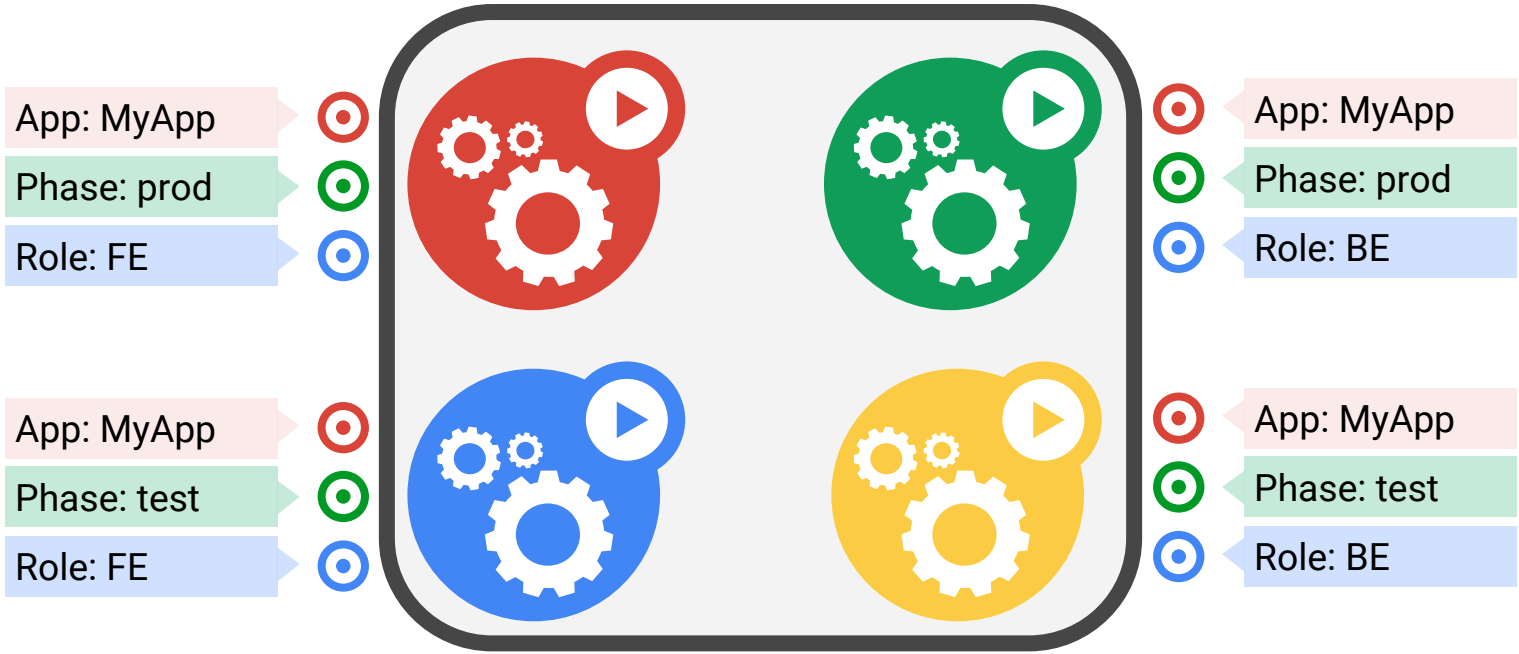
App: MyApp
Phase: prod
Role: FE

App: MyApp
Phase: prod
Role: BE

App: MyApp
Phase: test
Role: FE

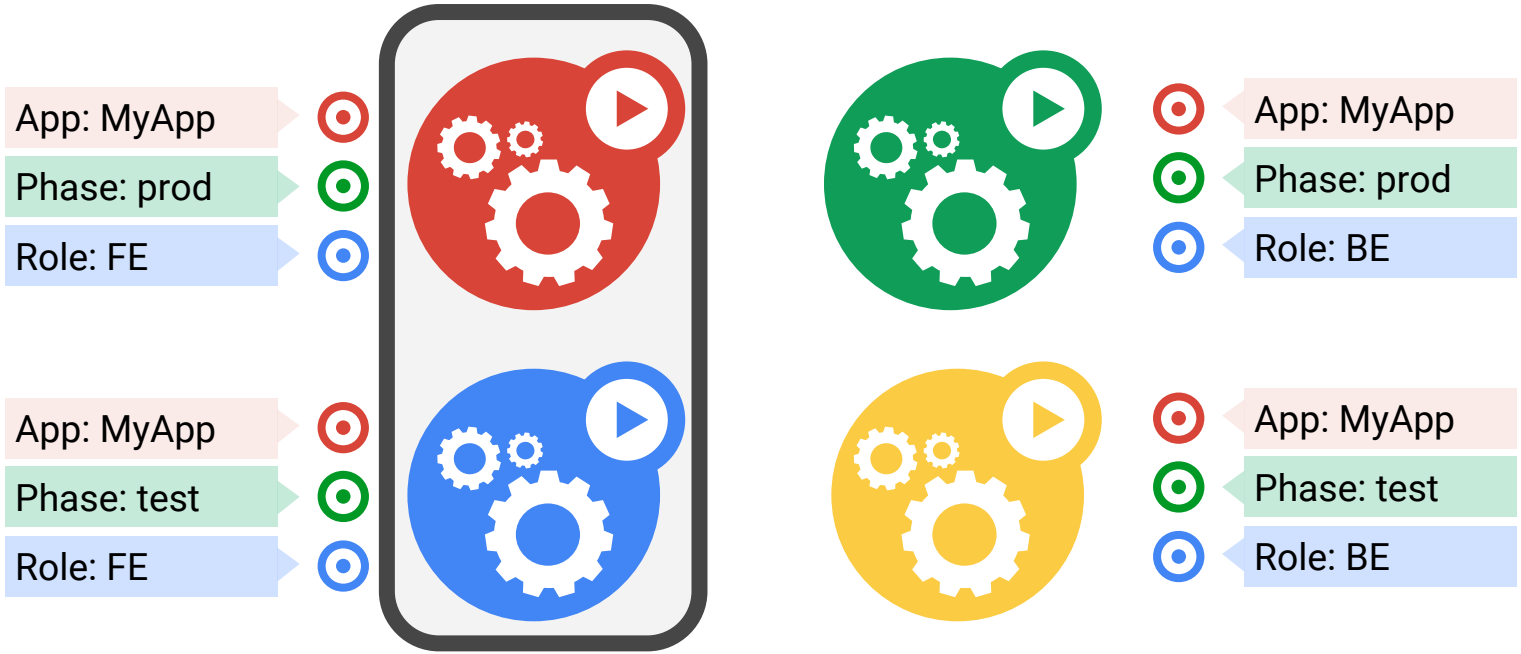
App: MyApp
Phase: test
Role: BE

Selector



App = MyApp

Selector



App = MyApp, Role = FE

Service

A logical grouping of pods that perform the same function (the Service's endpoints)

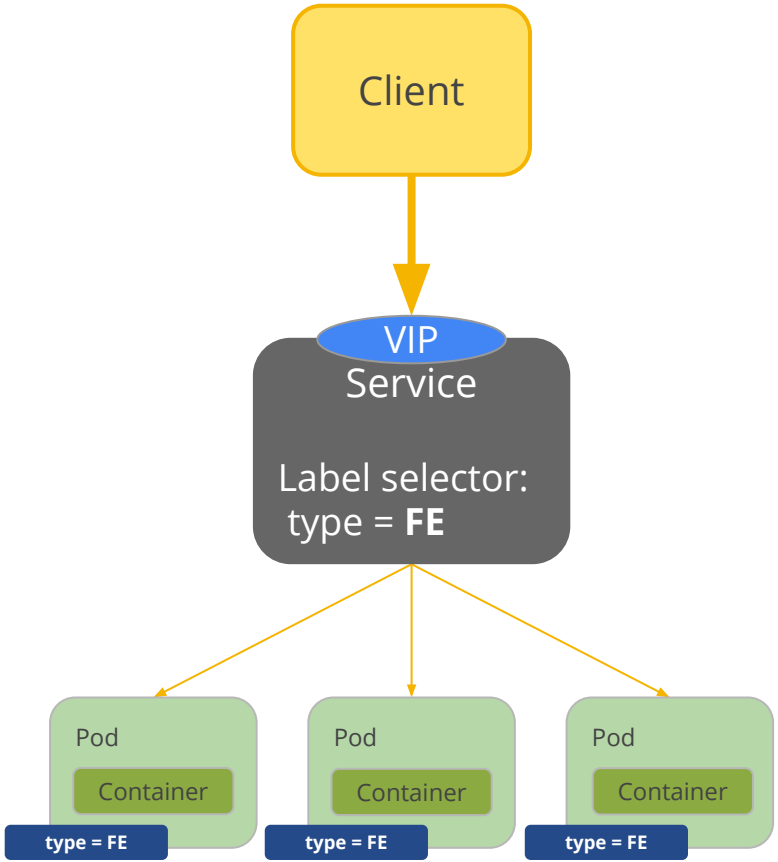
- grouped by label selector

Load balances incoming requests across constituent pods

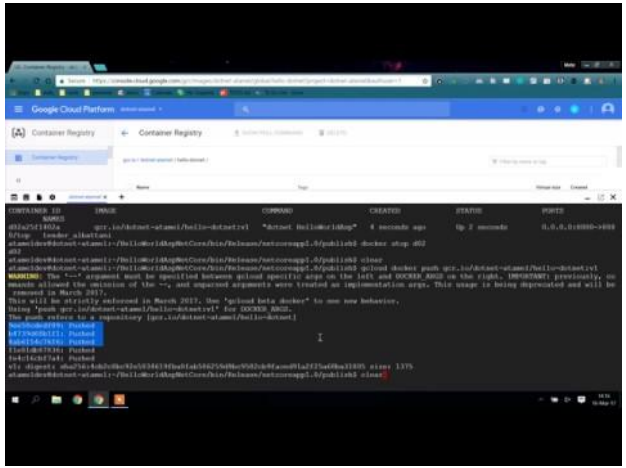
Choice of pod is random but supports session affinity (ClientIP)

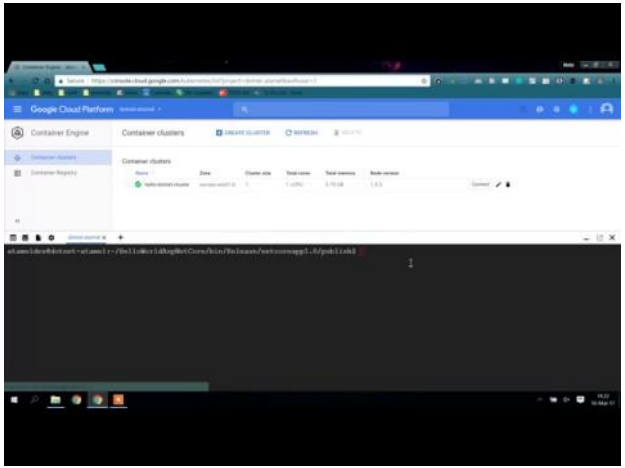
Gets a **stable** virtual IP and port

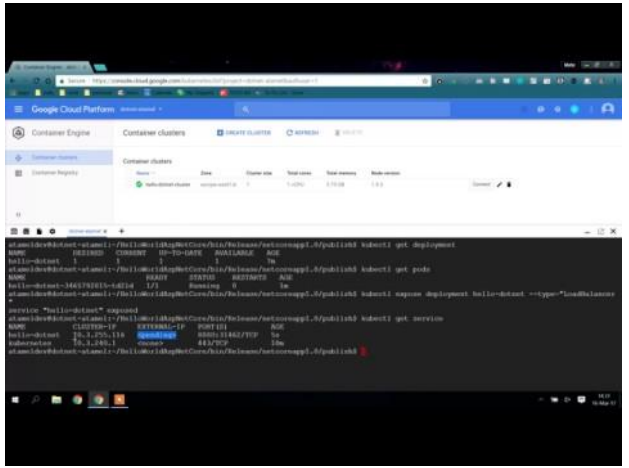
- also a DNS name



Demo: Kubernetes Cluster







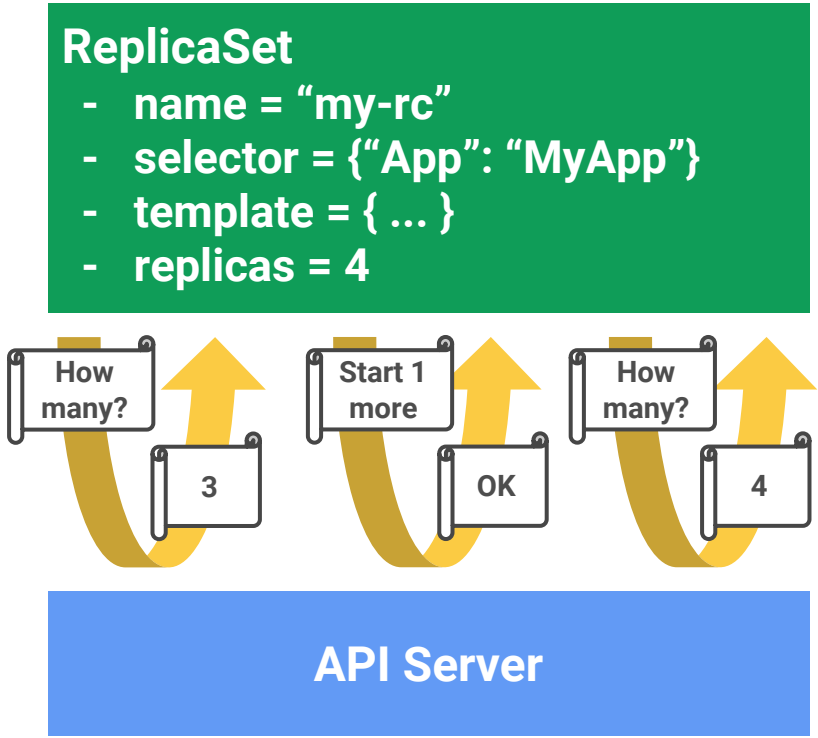
Resiliency & Redundancy

ReplicaSet*

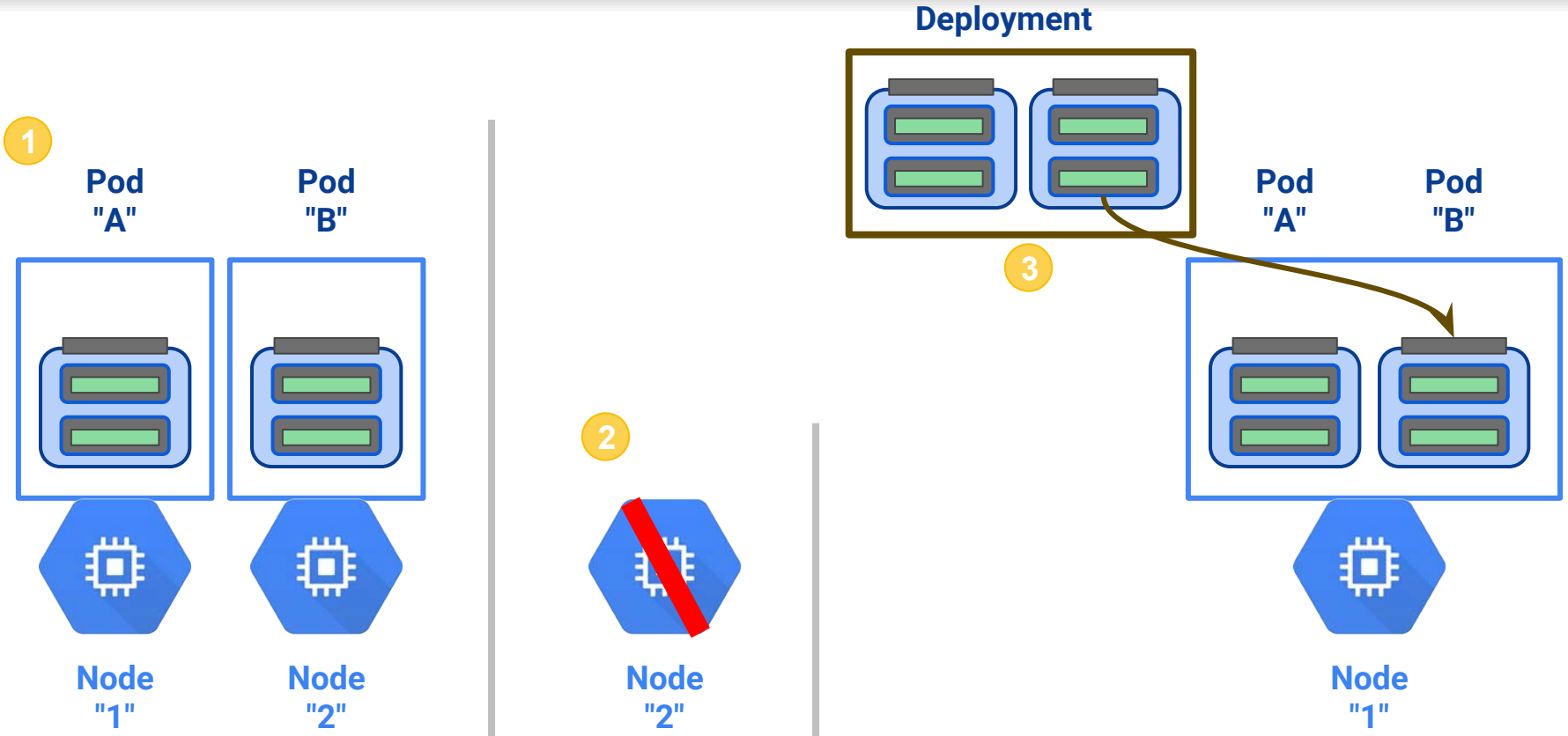
One job: ensure N copies of a pod

- grouped by a selector
- too few? start some
- too many? kill some

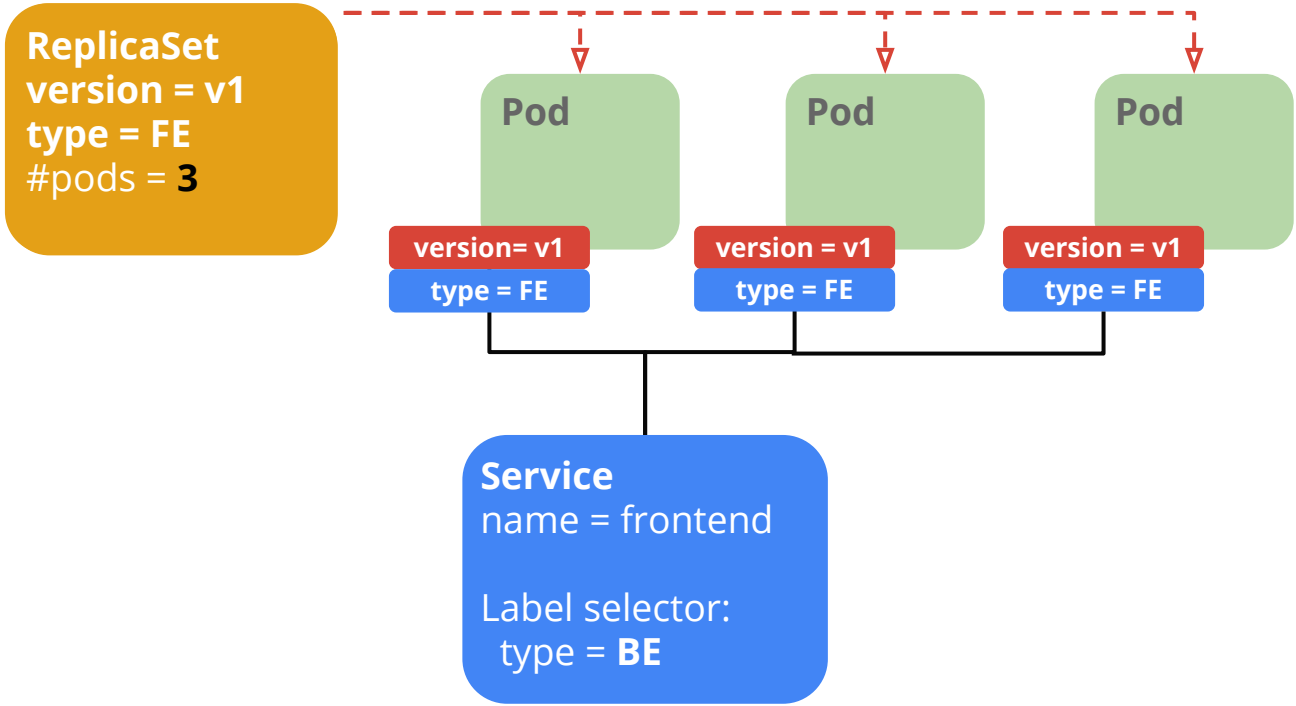
* The evolution of ReplicationControllers



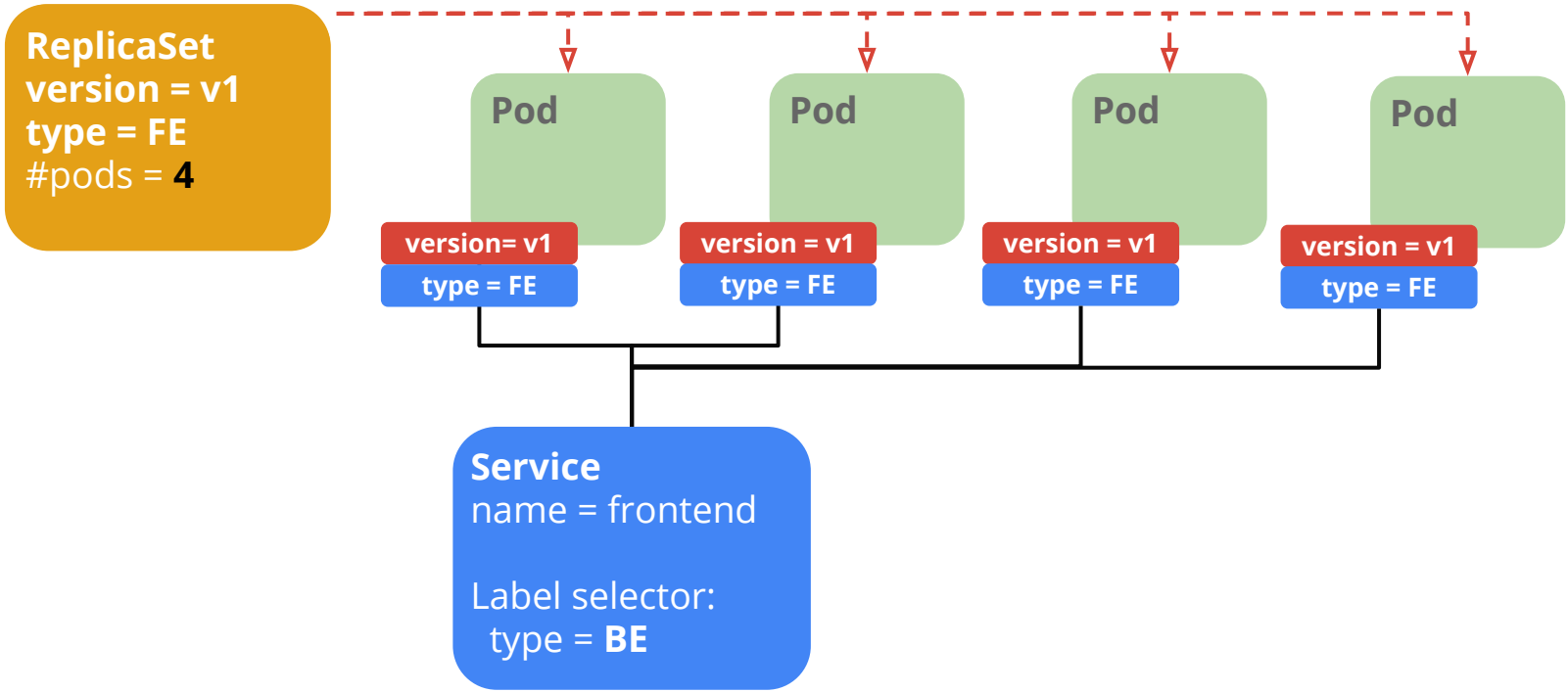
Resiliency



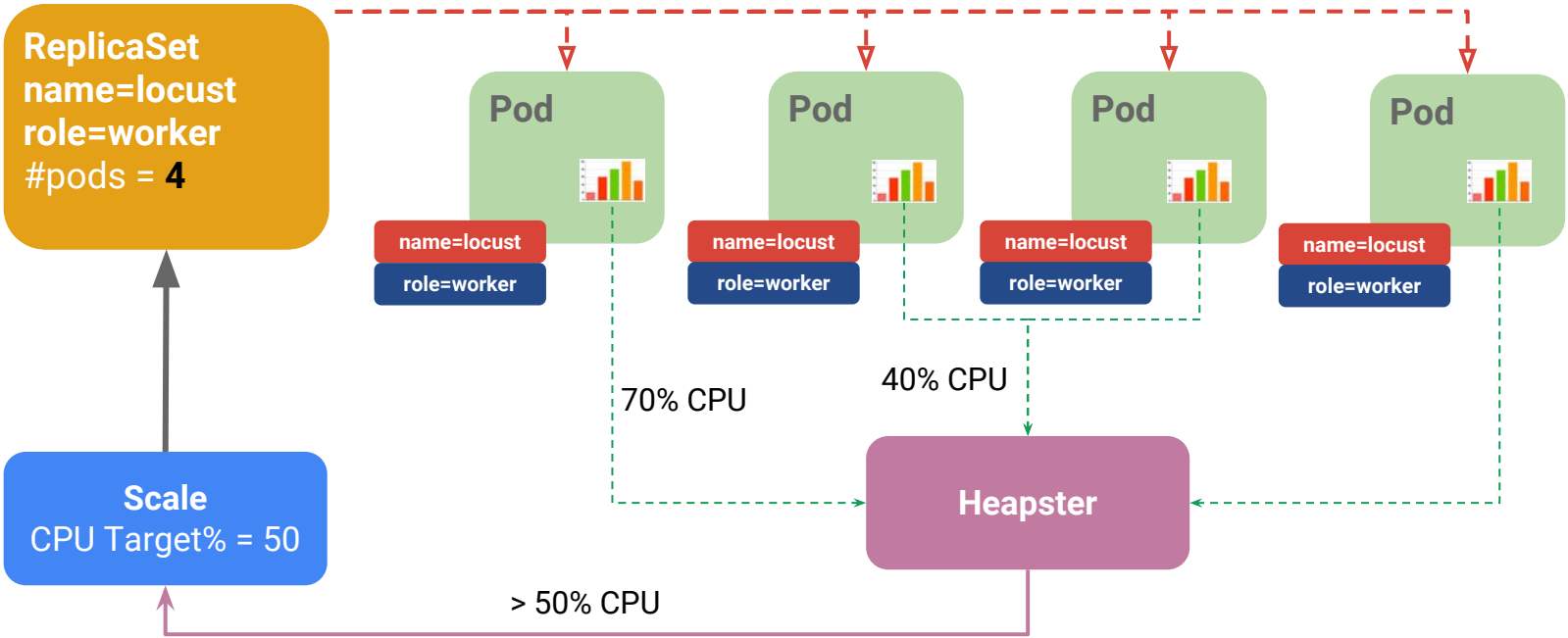
Scaling


ReplicaSet
version = v1
type = FE
#pods = 3
Pod
Pod
Pod
version= v1
type = FE
version = v1
type = FE
version = v1
type = FE
Service
name = frontend
Label selector:
type = BE

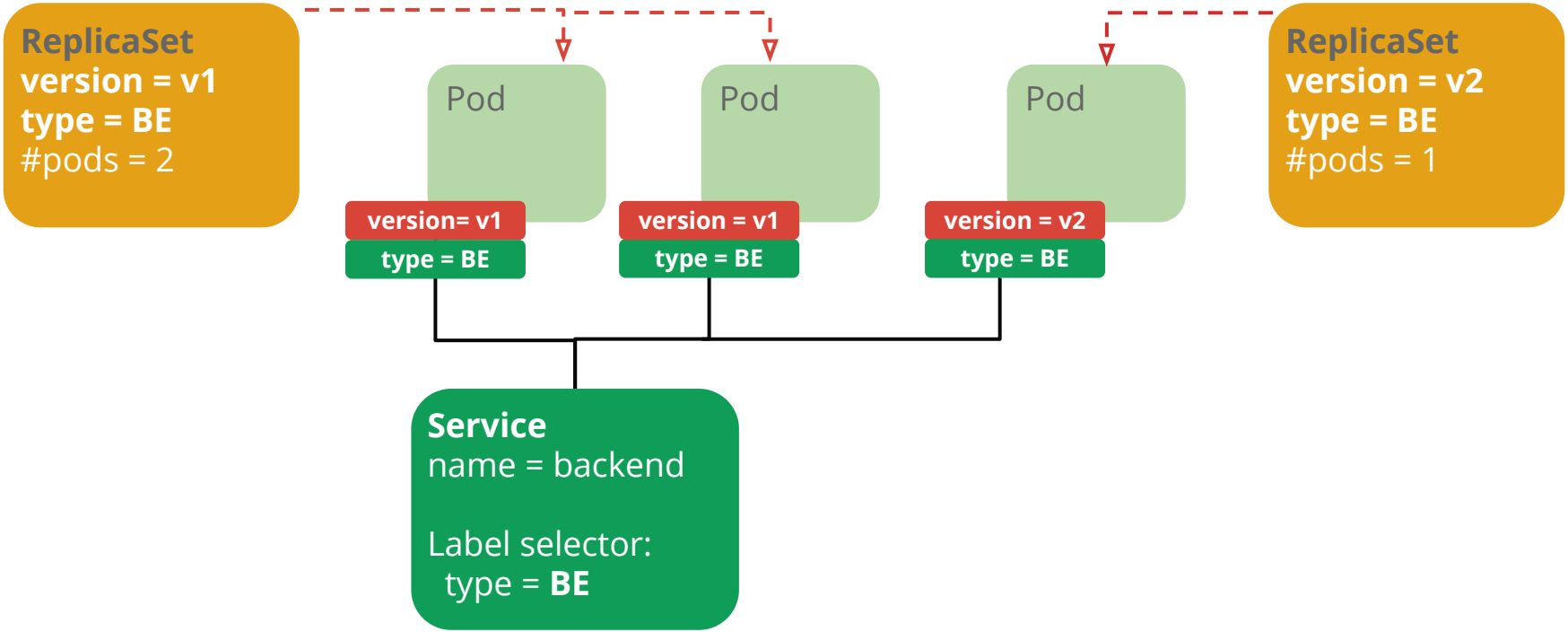
Scaling


ReplicaSet
version = v1
type = FE
#pods = 4
Pod
Pod
Pod
Pod
version= v1
type = FE
version = v1
type = FE
version = v1
type = FE
version = v1
type = FE
Service
name = frontend
Label selector:
type = BE

Autoscaling



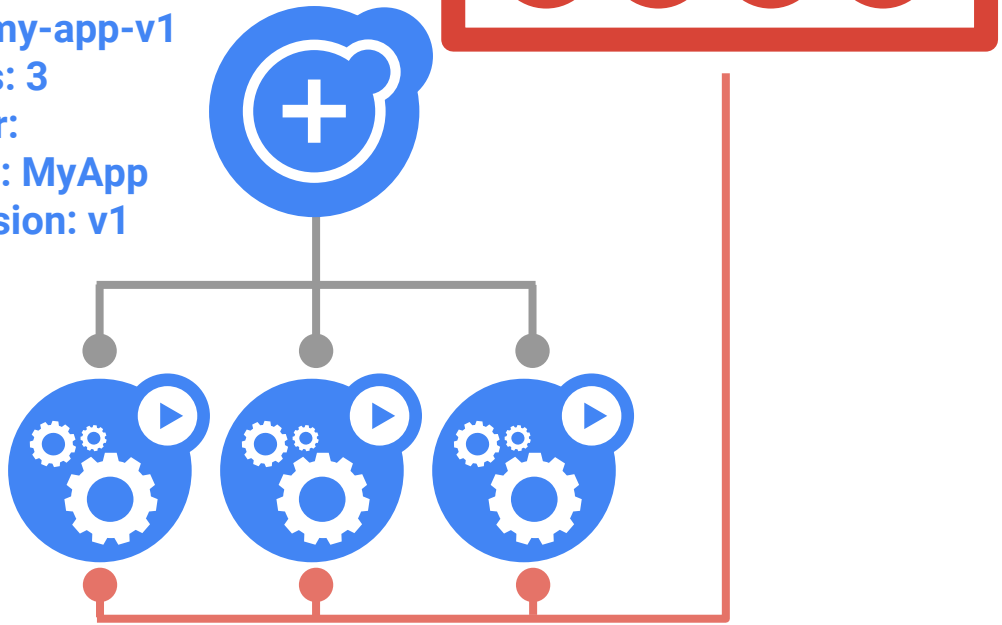
Canary Deployments



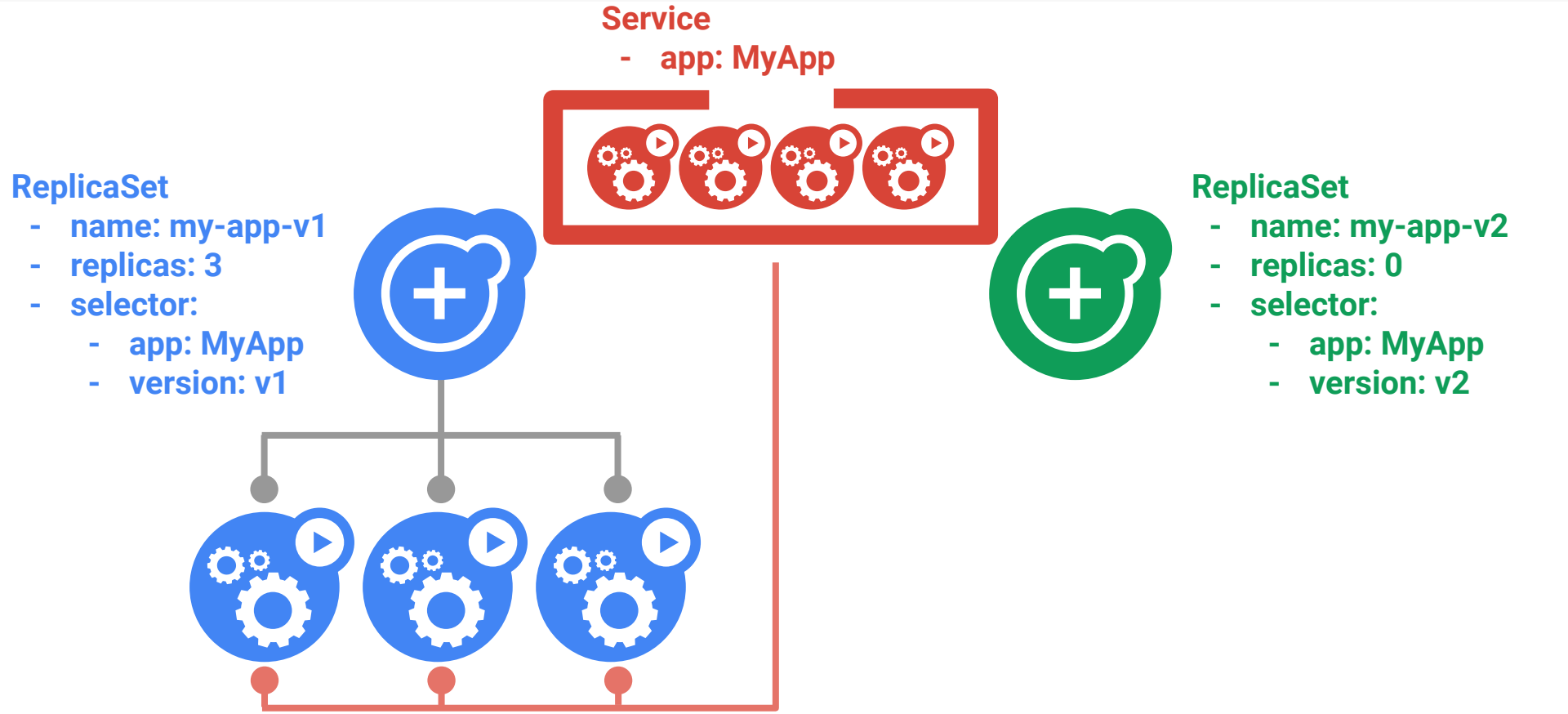
Rolling Update

Service
- app: MyApp

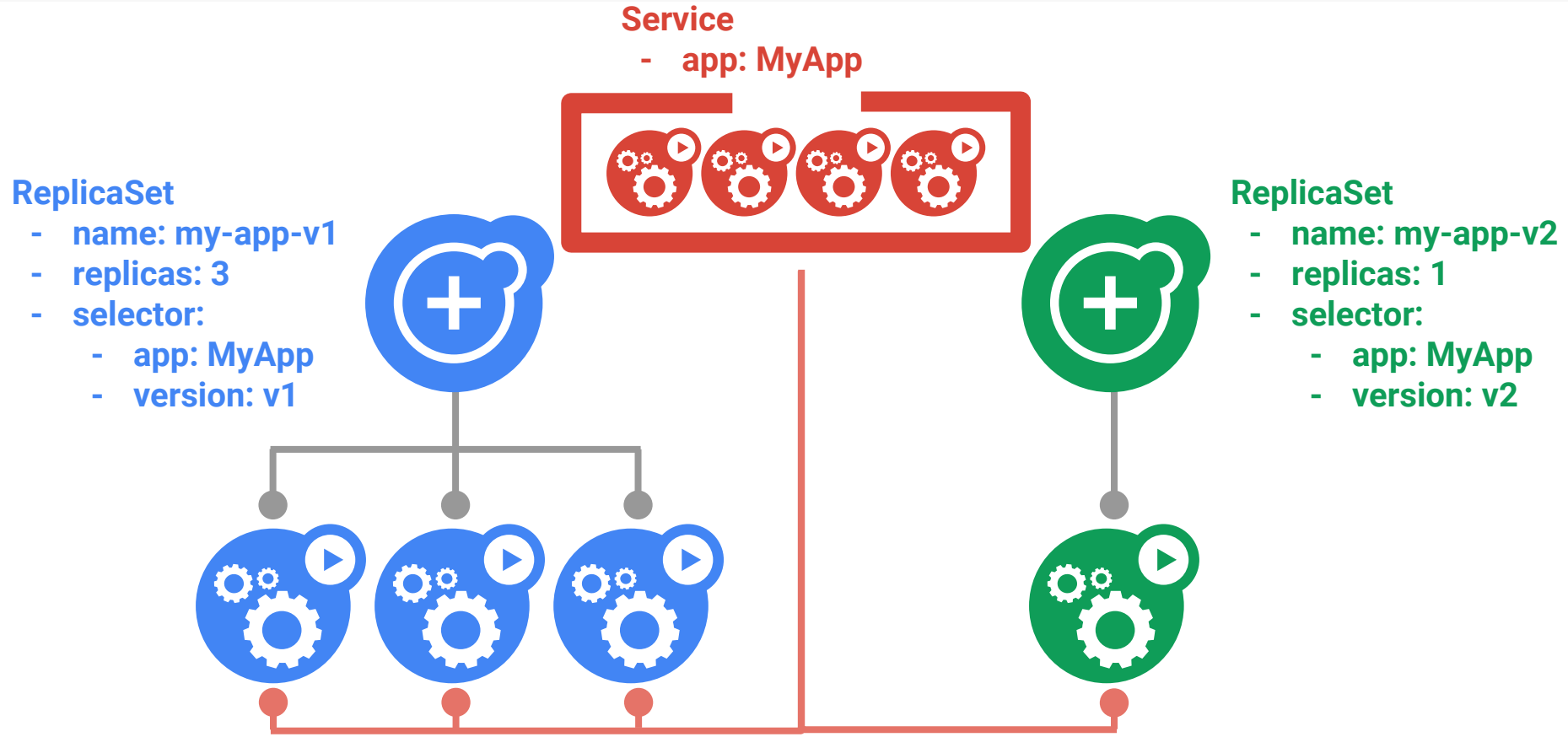
ReplicaSet
- name: my-app-v1
- replicas: 3
- selector:
  - app: MyApp
  - version: v1



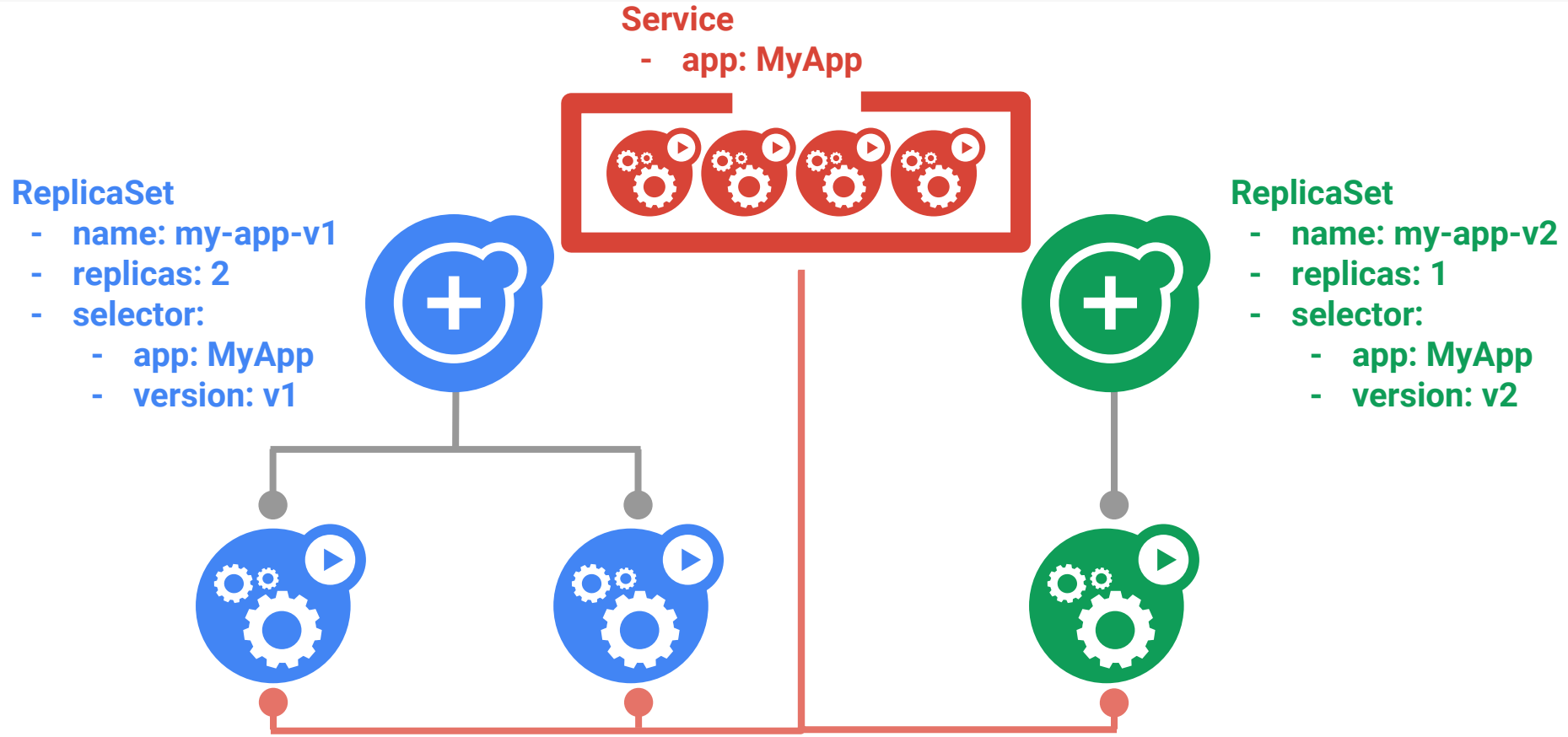
Rolling Update



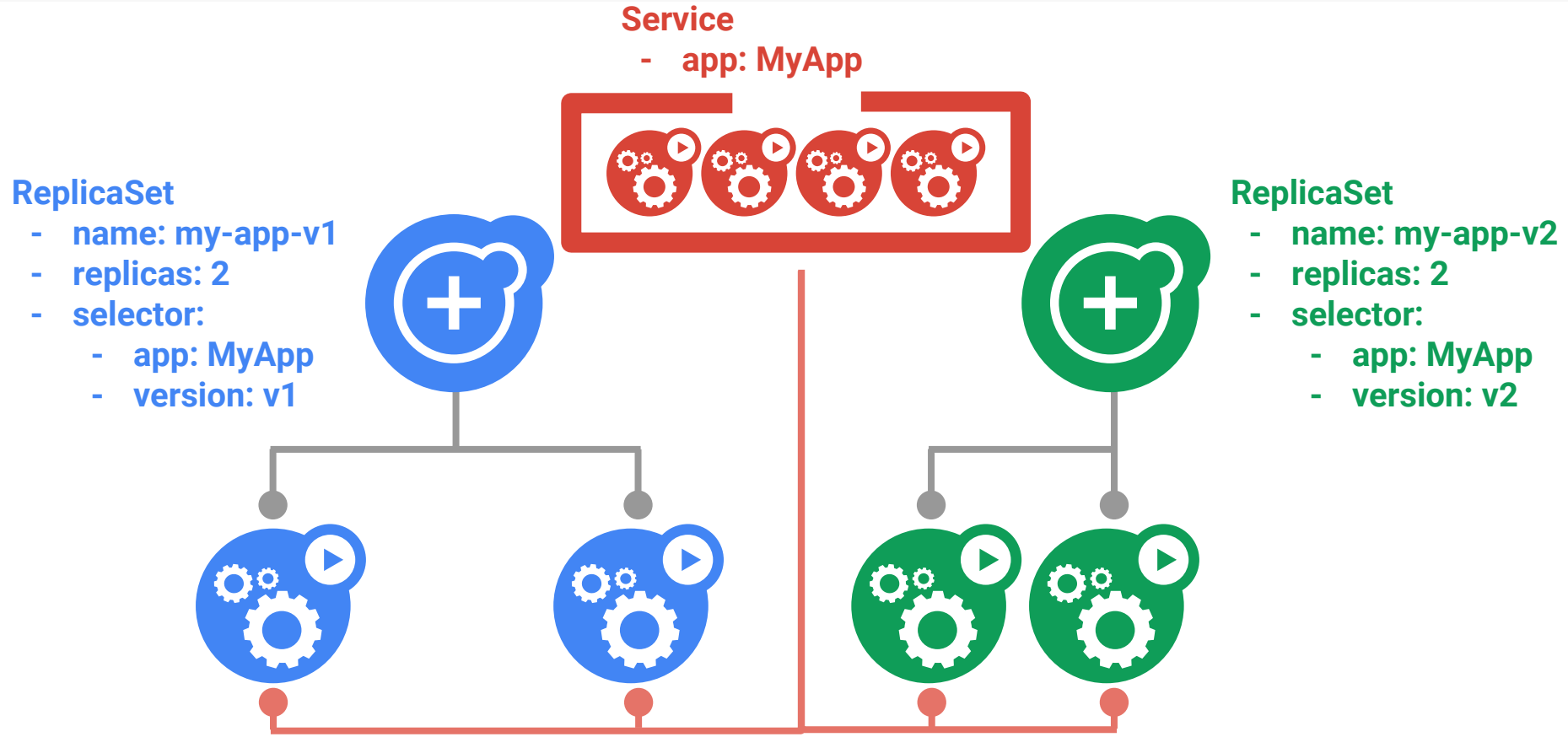
Rolling Update



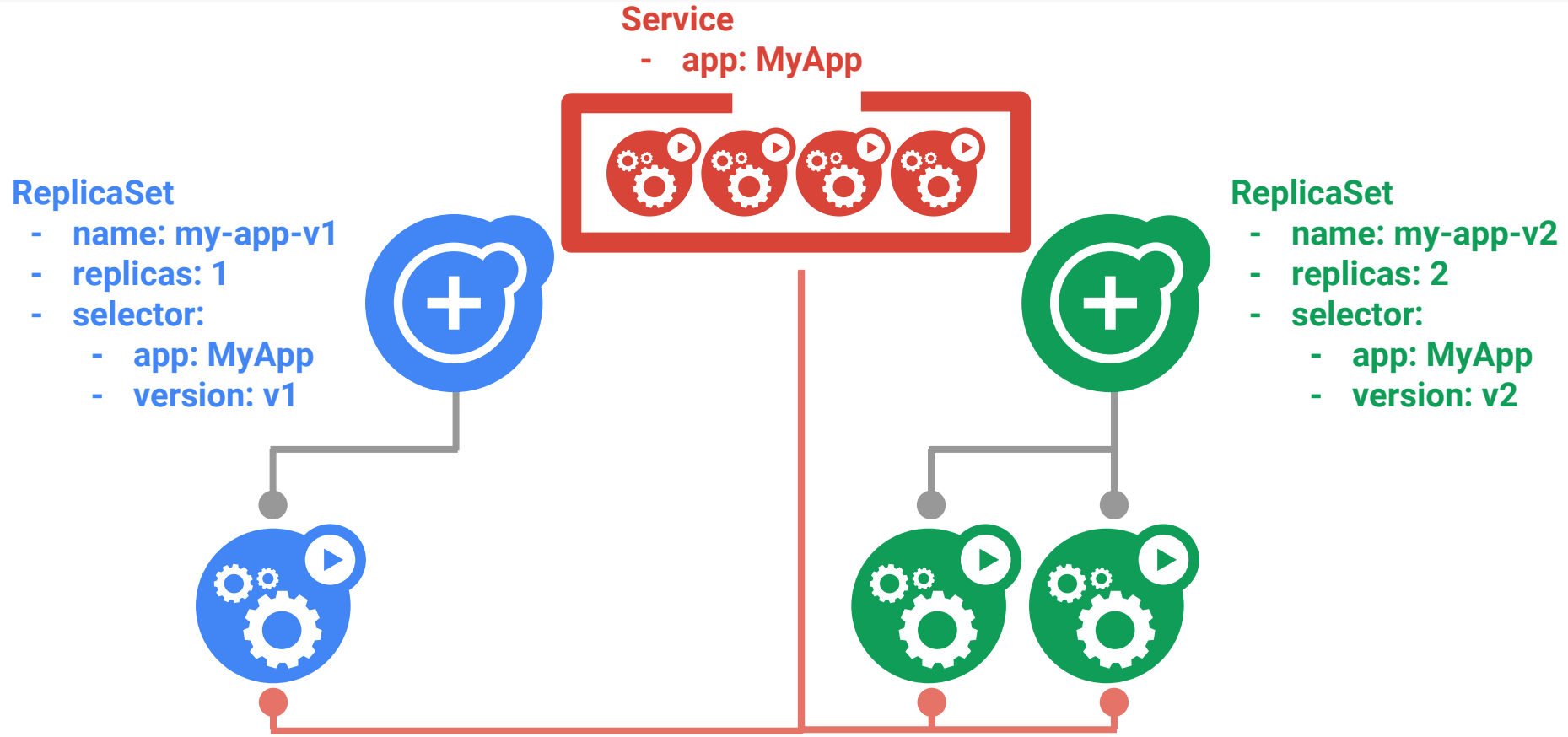
Rolling Update



Rolling Update

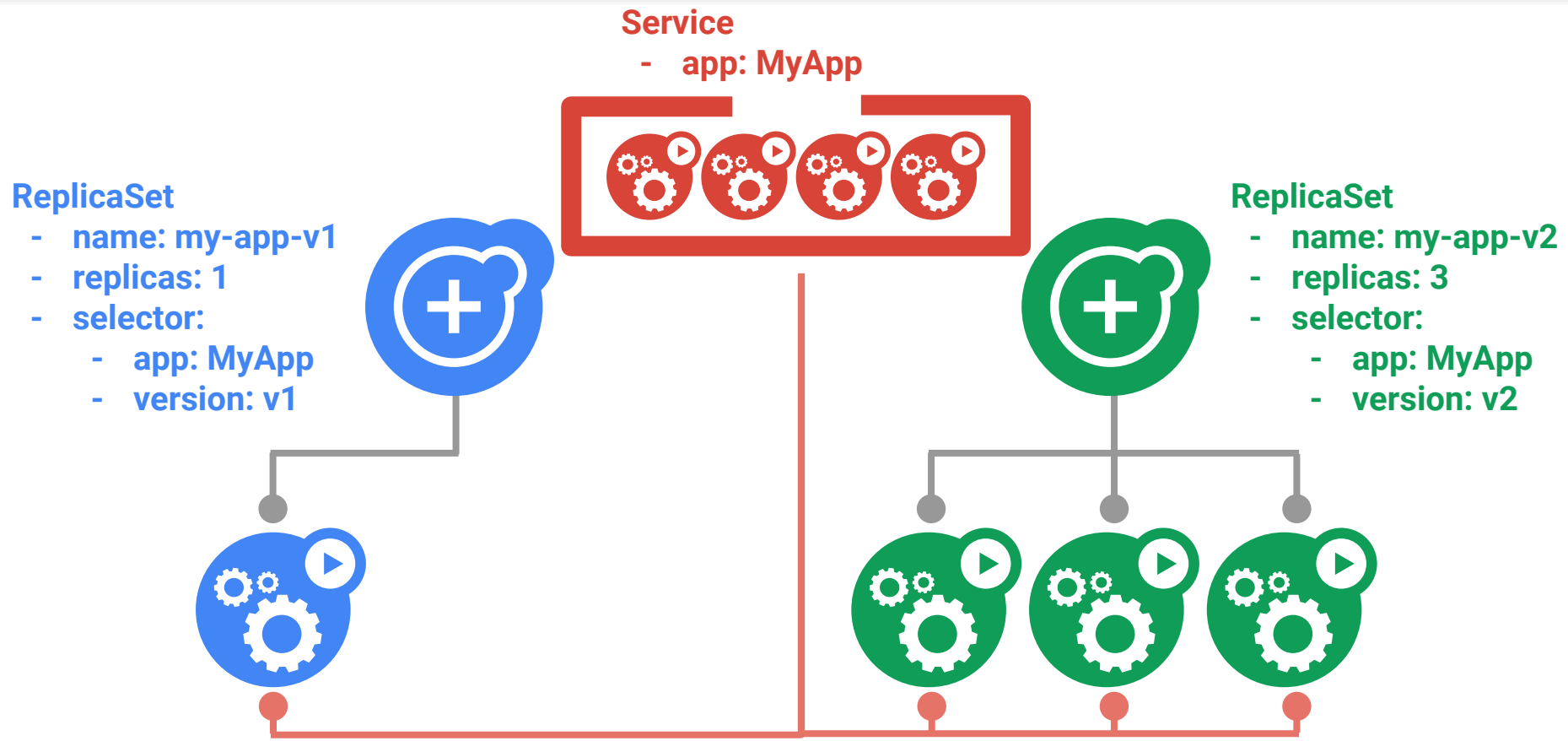


Rolling Update



Rolling Update

@meteatamel

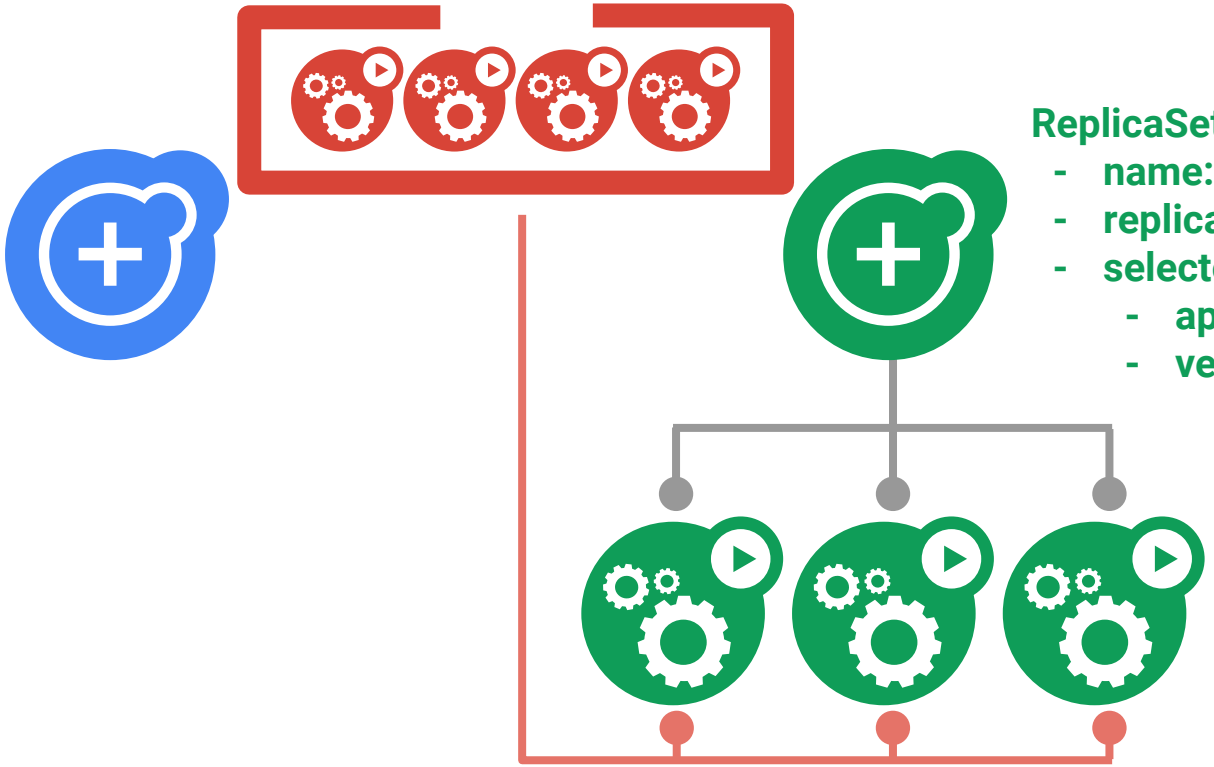


Rolling Update

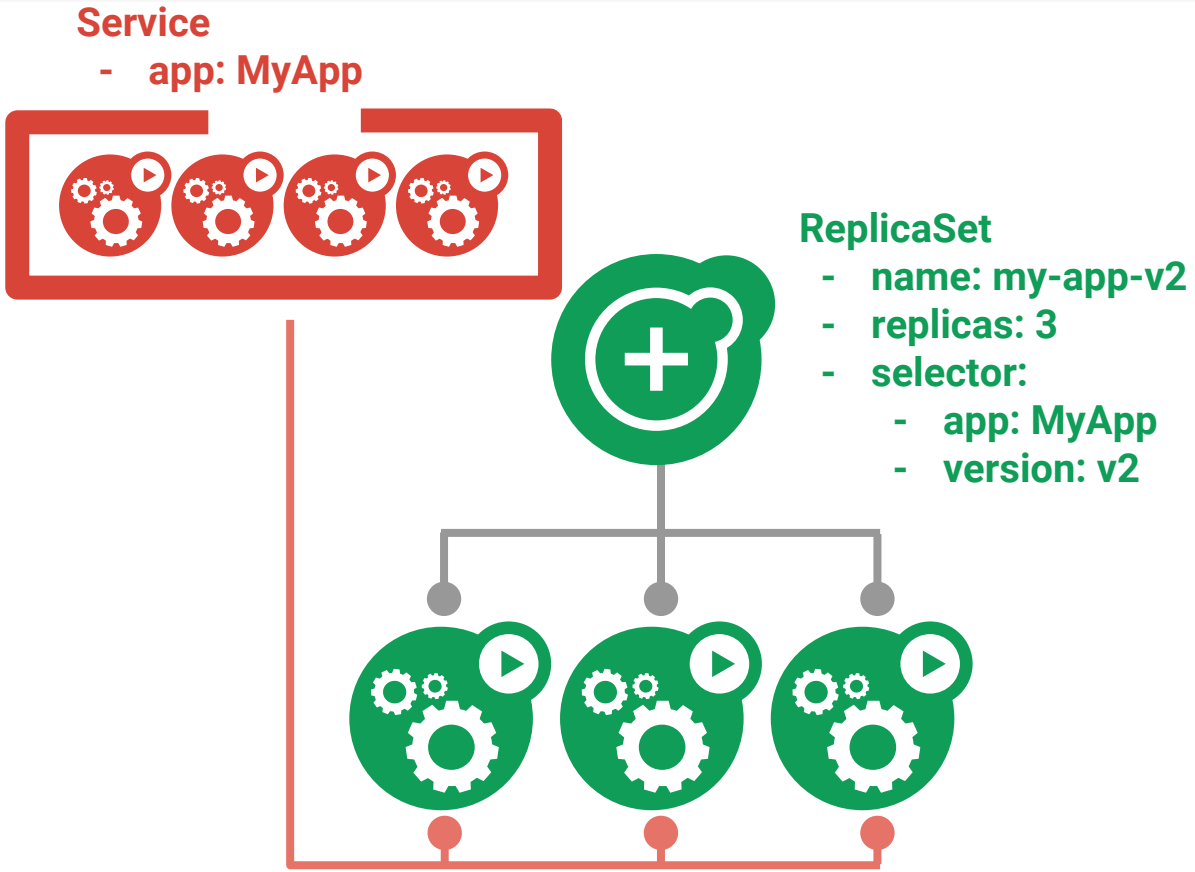
Service
- app: MyApp

ReplicaSet
- name: my-app-v1
- replicas: 0
- selector:
  - app: MyApp
  - version: v1

ReplicaSet
- name: my-app-v2
- replicas: 3
- selector:
  - app: MyApp
  - version: v2



Rolling Update



Health Check: Liveness Probes

Liveness Probes make sure your application is running

```
livenessProbe:
  # an http probe
  httpGet:
    path: /healthz
    port: 8080
  initialDelaySeconds: 15   # wait 15 seconds after pod is started to check for health
  timeoutSeconds: 1         # wait 1 second for a response to health check
```

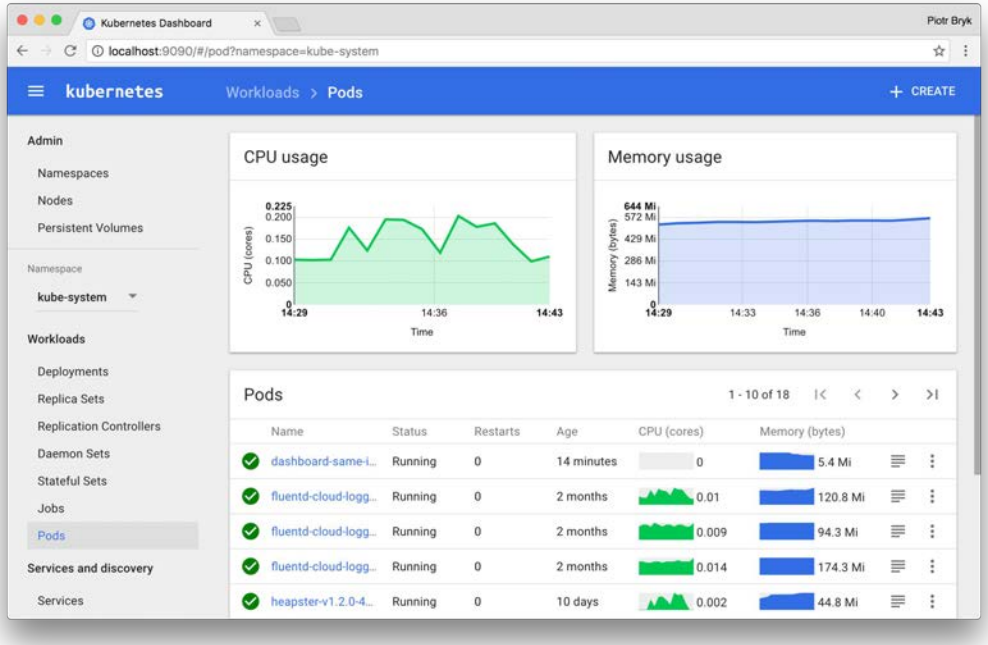
Health Check: Readiness Probes

Readiness probes make sure your application is ready to serve traffic

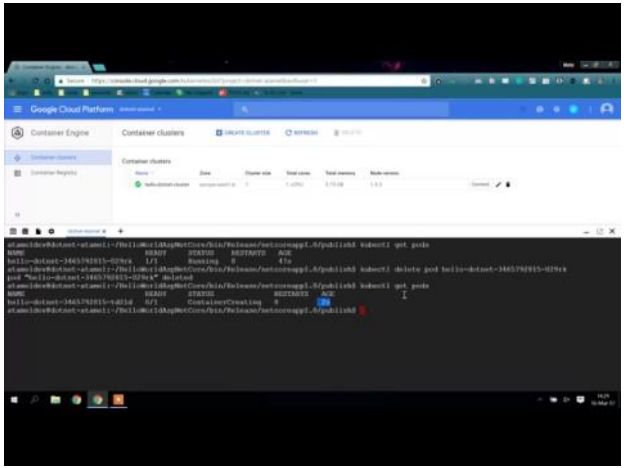
```
readinessProbe:
  # an http probe
  httpGet:
    path: /readiness
    port: 8080
  initialDelaySeconds: 20  # wait 20 seconds after pod is started to check for health
  timeoutSeconds: 5        # wait 5 second for a response to health check
```

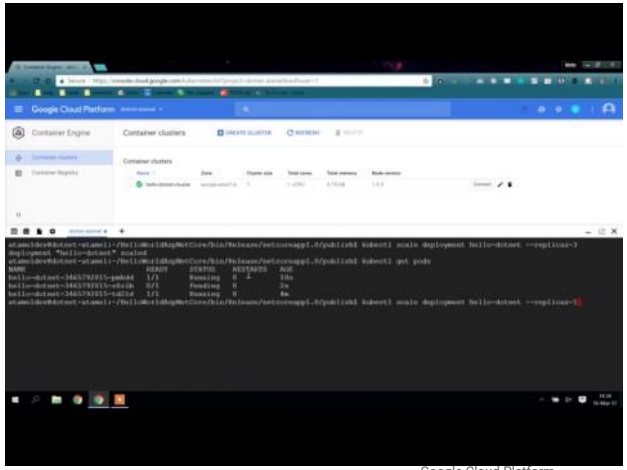
Kubernetes Dashboard

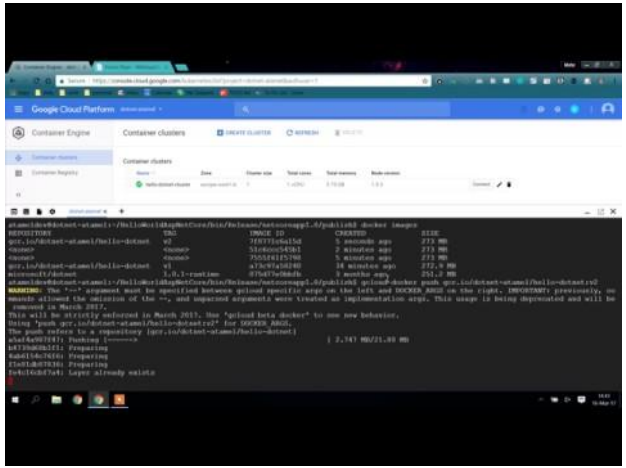
A general purpose, web-based UI to view/manage Kubernetes clusters



Demo: Resiliency







There is more!

DaemonSets

Problem: how to run a Pod on every node?

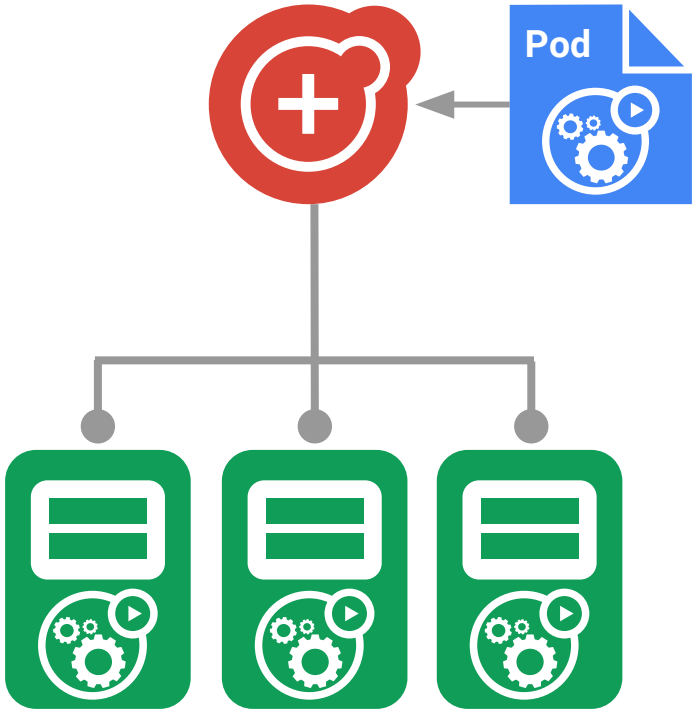
- or a subset of nodes

Similar to ReplicaSet

- principle: do one thing, don't overload

"Which nodes?" is a selector

Use familiar tools and patterns



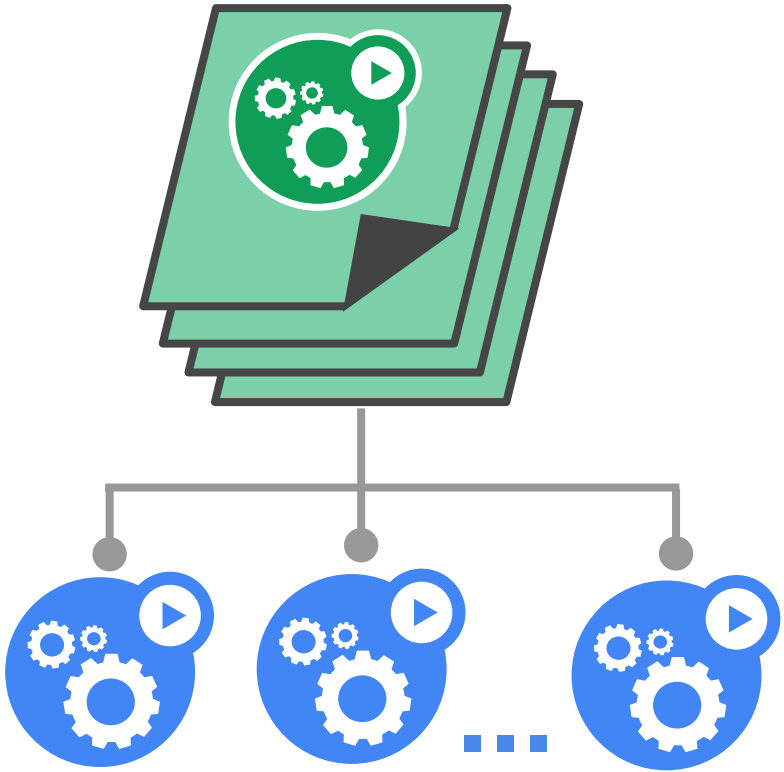
Jobs

Run-to-completion, as opposed to run-forever

- Express parallelism vs. required completions
- Workflow: restart on failure
- Build/test: don't restart on failure

Aggregates success/failure counts

Built for batch and big-data work



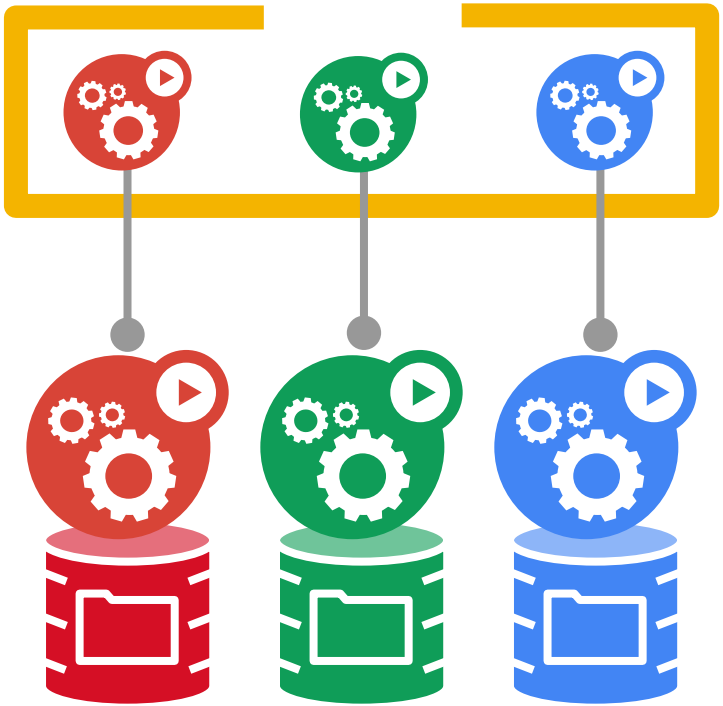
StatefulSets

Goal: enable clustered software on Kubernetes

- mysql, redis, zookeeper, ...

Clustered apps need “identity” and sequencing guarantees

- stable hostname, available in DNS
- an ordinal index
- stable storage: linked to the ordinal & hostname
- discovery of peers for quorum
- startup/teardown ordering



ConfigMaps

Goal: manage app configuration

- ...without making overly-brittle container images

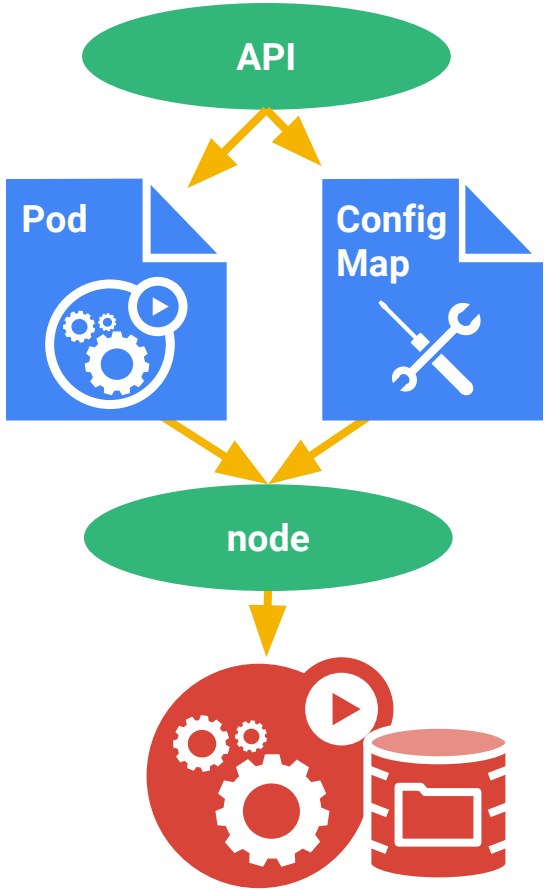
12-factor says config comes from the environment

- Kubernetes is the environment

Manage config via the Kubernetes API

Inject config as a virtual volume into your Pods

- late-binding, live-updated (atomic)
- also available as env vars



Secrets

Goal: grant a pod access to a secured *something*

- don't put secrets in the container image!

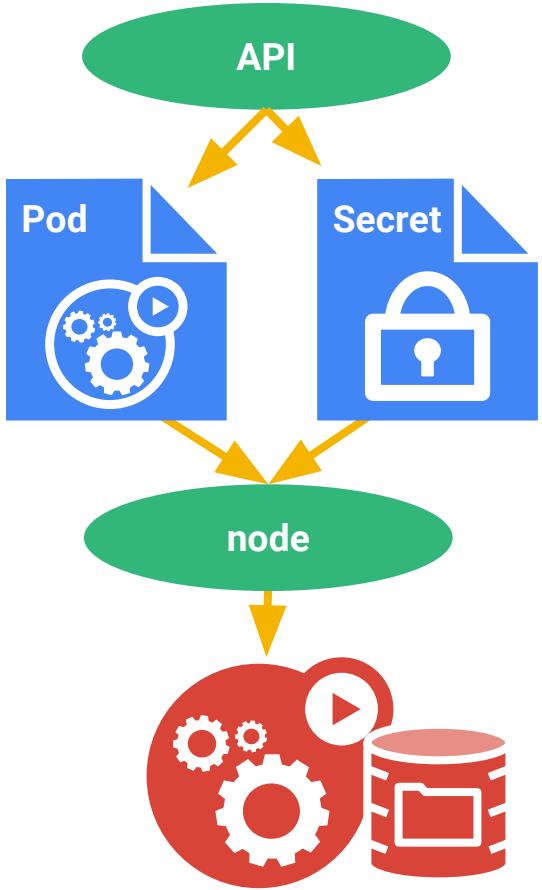
12-factor says config comes from the environment

- Kubernetes is the environment

Manage secrets via the Kubernetes API

Inject secrets as virtual volumes into your Pods

- late-binding, tmpfs - never touches disk
- also available as env vars



Thank You

@meteatamel
atamel@google.com
meteatamel.wordpress.com