Google Cloud

Containers and Kubernetes

Google Cloud Training and Certification

Agenda Slide

01 Introduction to Containers

02 Kubernetes 101

03 Demo

Vic Iglesias

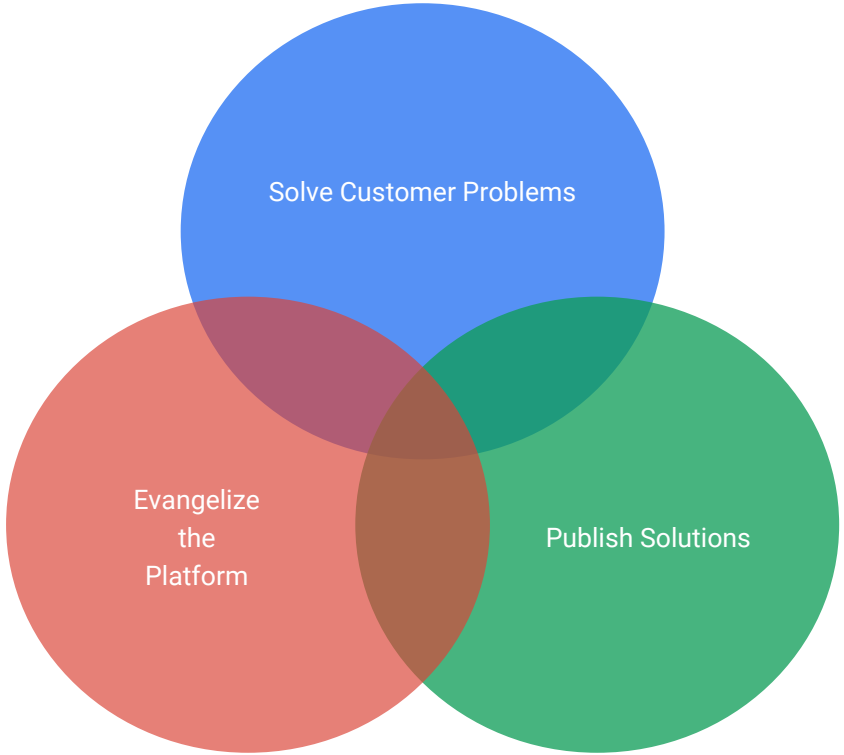
Cloud Solutions Architect

Graduated UCSB, 2008

viglesias@google.com

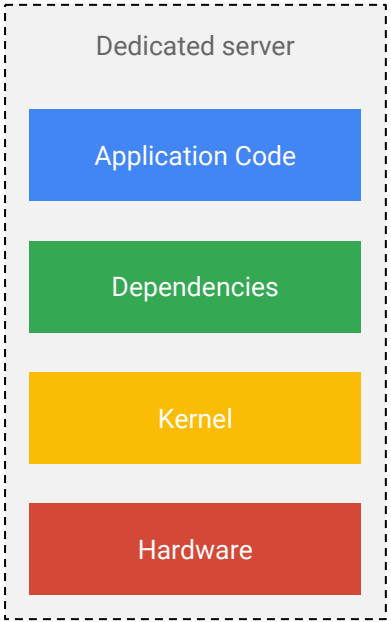
@vicnastea

What is a Cloud Solutions Architect?

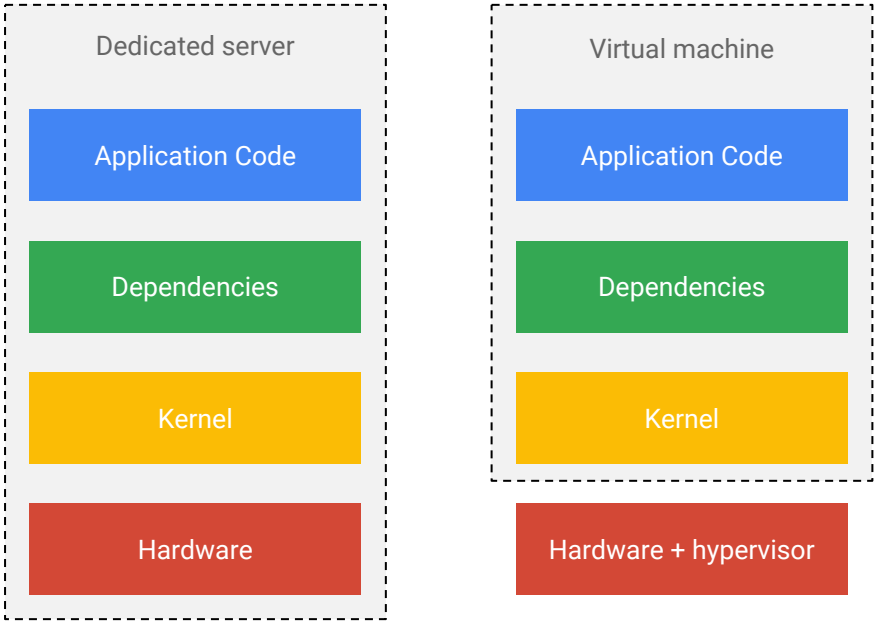


Introduction to Containers

Application Containers are changing the way people deploy and run applications

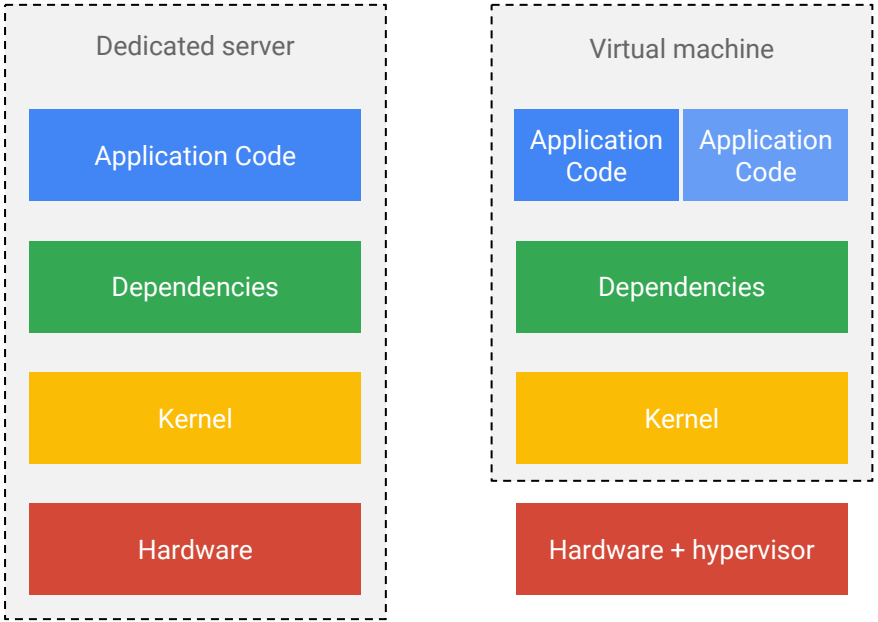


Deployment ~months
Low utilization
Not portable



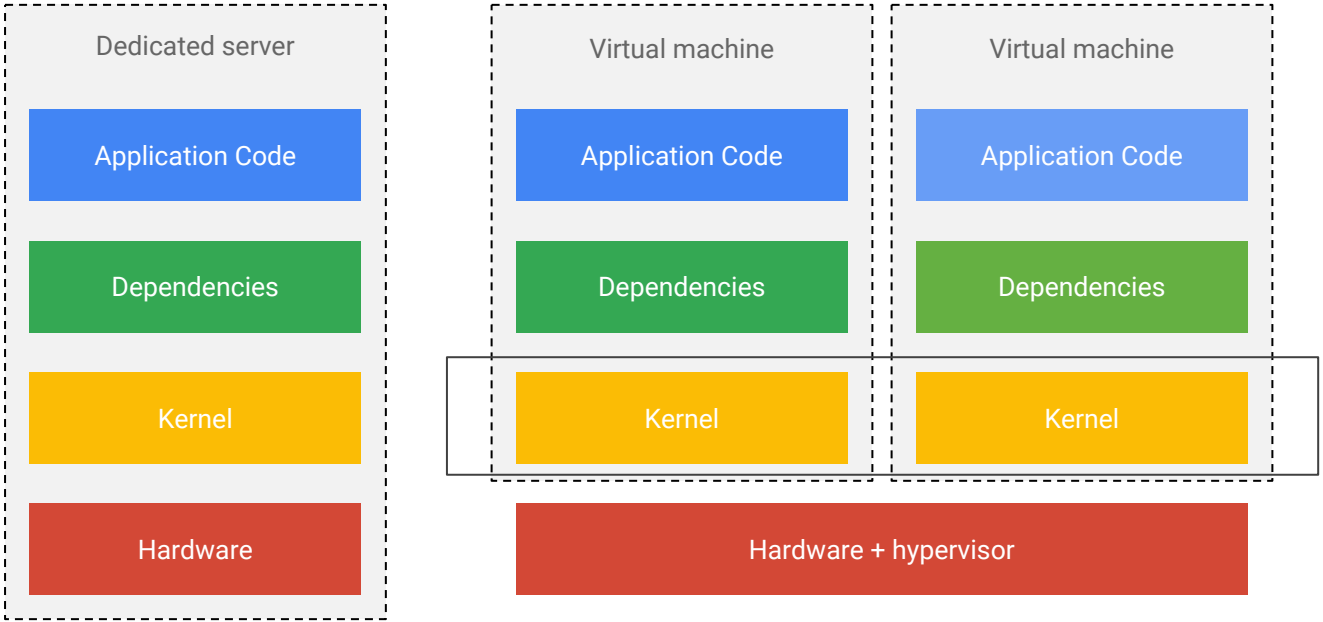
Deployment ~months
Low utilization
Not portable

Deployment ~days (mins)
Improved utilization
Hypervisor specific



Deployment ~months
Low utilization
Not portable

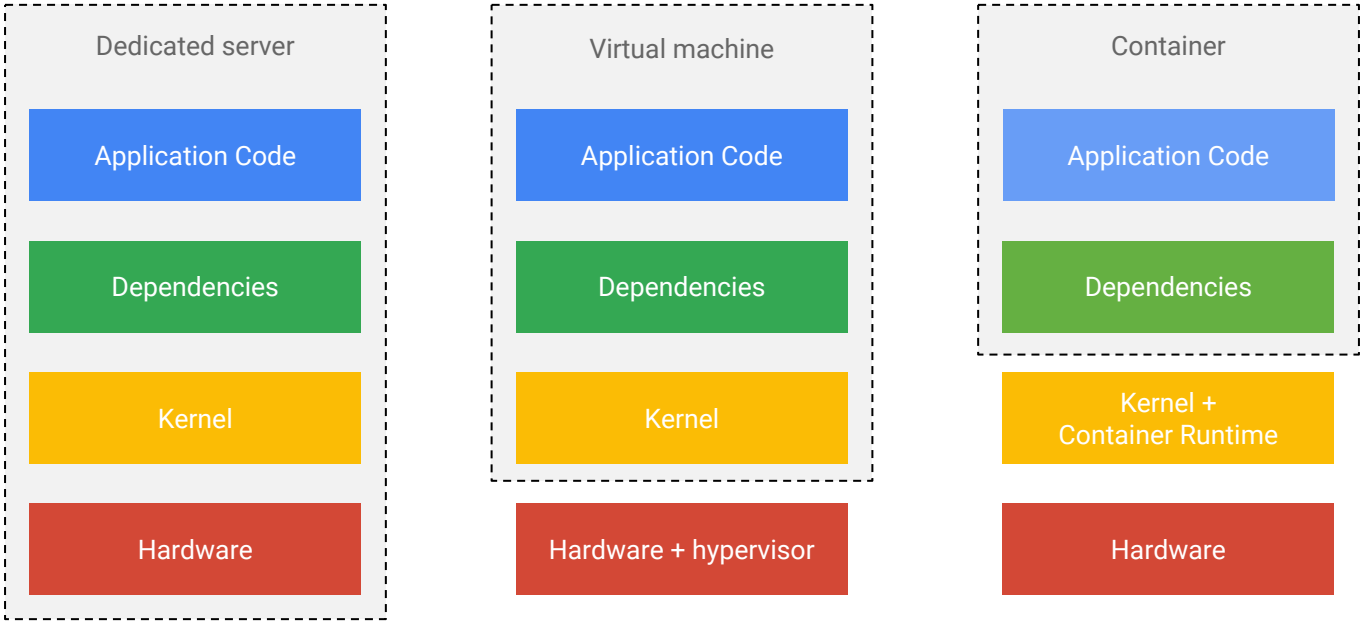
Deployment ~days (mins)
Hypervisor specific
Low isolation, Tied to OS



Deployment ~months
Not portable
Low utilization

Deployment ~days (mins)
Hypervisor specific
Low isolation, Tied to OS

Deployment ~days (mins)
Hypervisor specific
Redundant OS

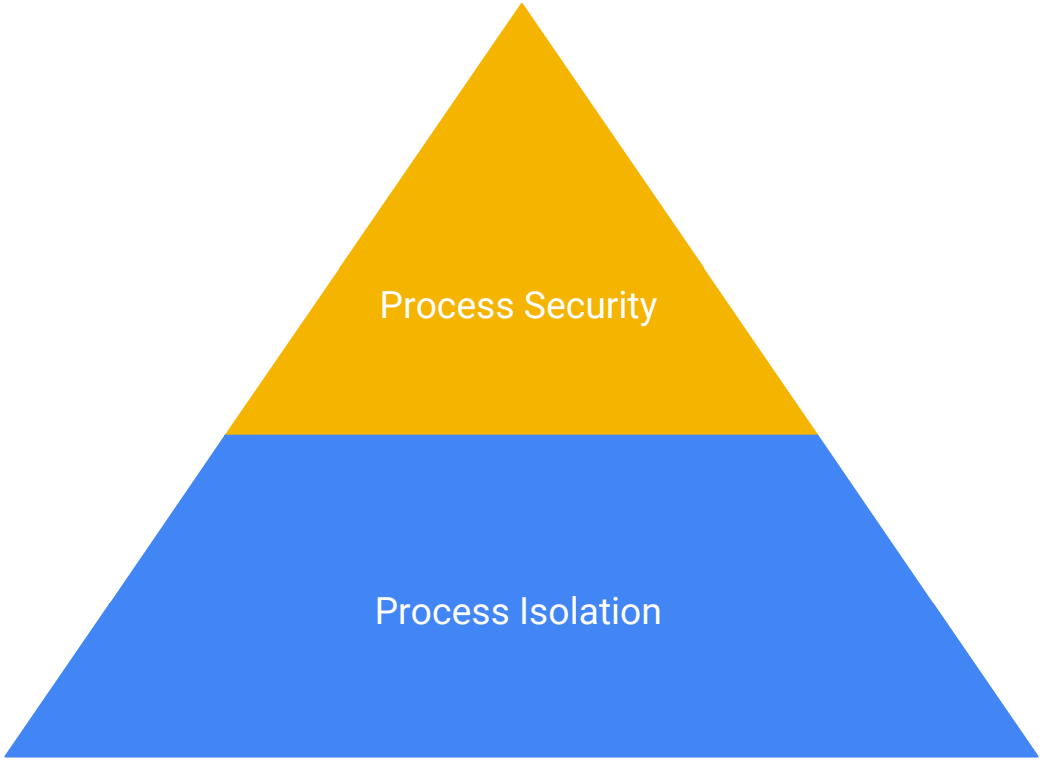


Deployment ~months
Not portable
Low utilization

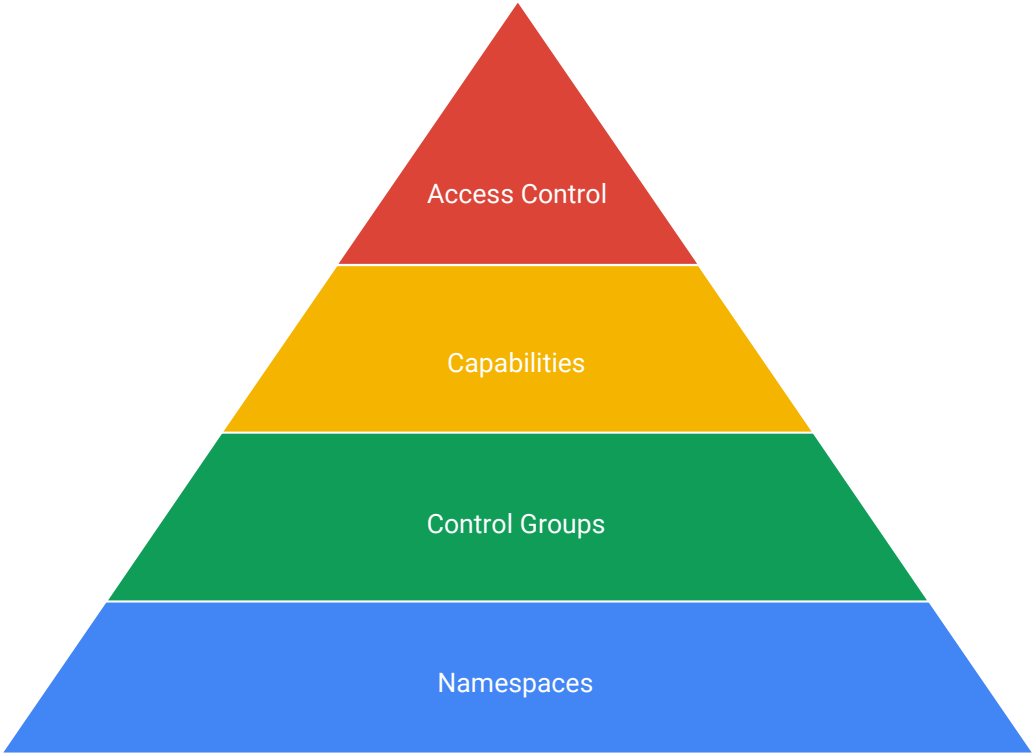
Deployment ~days (mins)
Hypervisor specific
Low isolation, Tied to OS

Deployment ~mins (sec)
Portable
Very Efficient

OS Virtualization Building Blocks

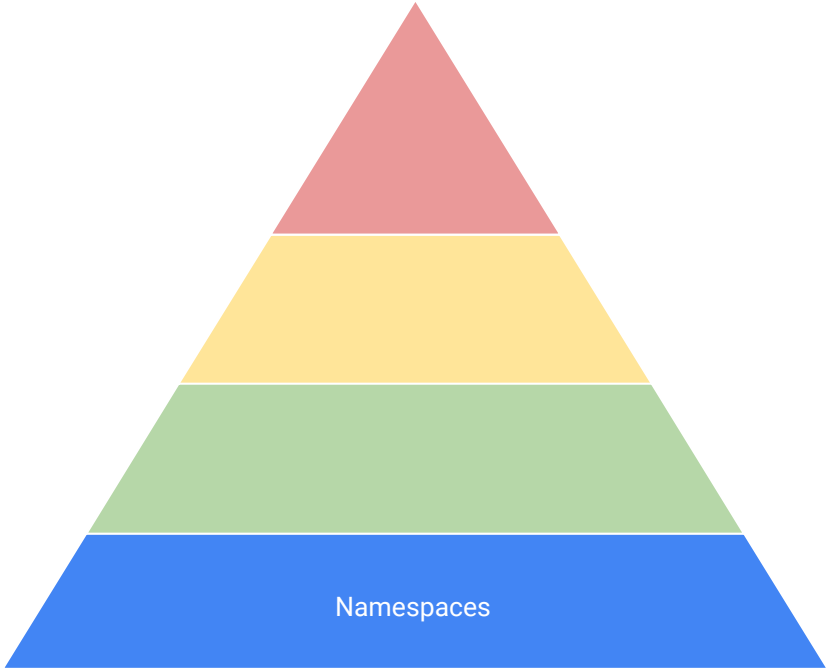


OS Virtualization Building Blocks



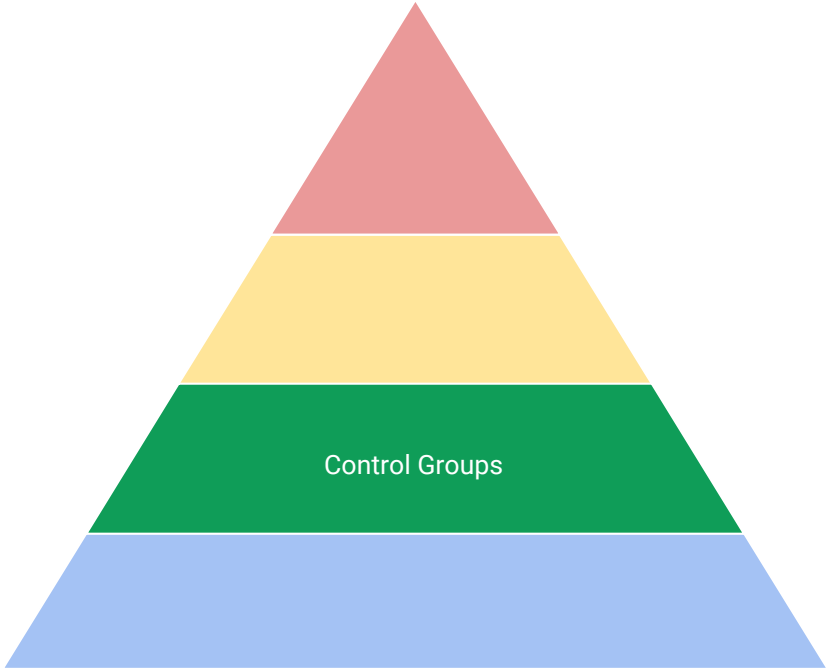
Namespaces

- Limit what a process can see
- Types
  - Net
  - Pid
  - IPC
  - Mount
  - and more...



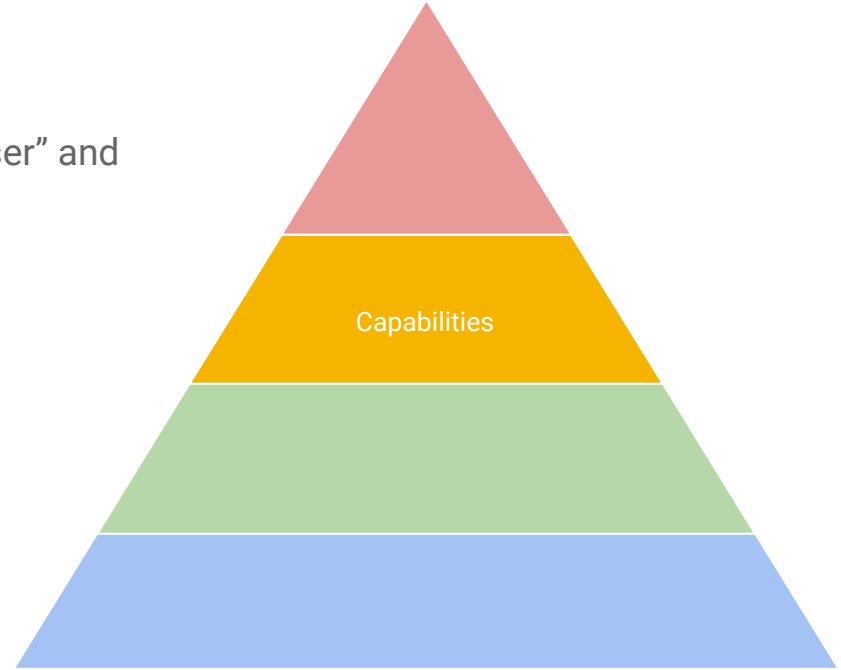
Control Groups

- Limit the resources that a process can use
- Types
  - Memory
  - CPU
  - Network
  - Block I/O
  - and more...



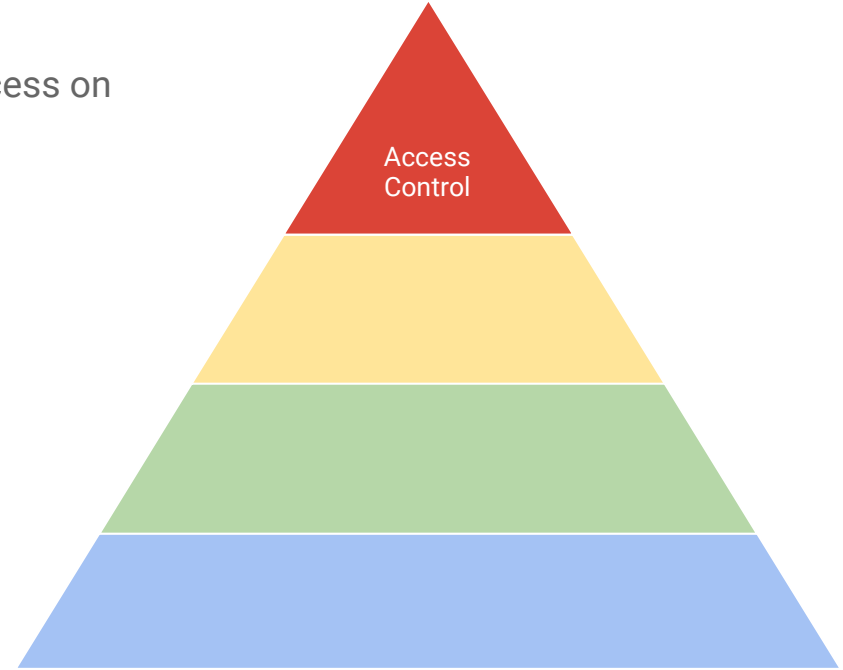
Root Capabilities

- Limits the things a process can do
- Granular permissions between a what a “regular user” and root can do.
- Types
  - CAP_CHOWN
  - CAP_NET_ADMIN
  - CAP_SYS_TIME
  - CAP_SYS_BOOT
  - and more...

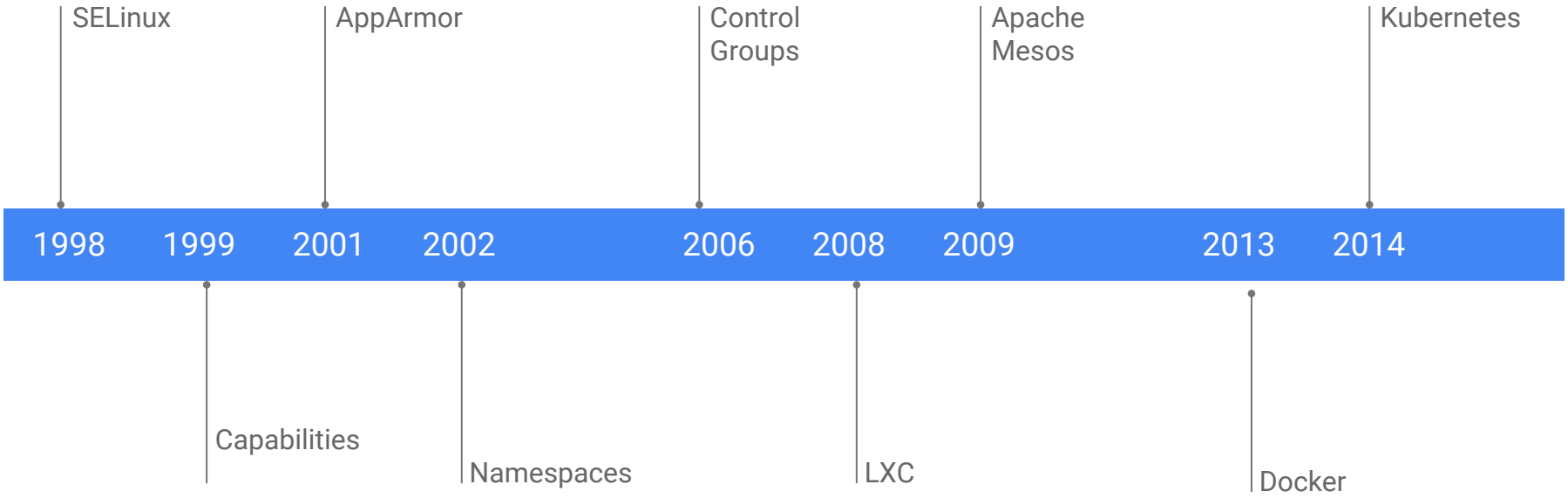


Access Control

- Fine grained controls over what processes can access on a system
- Examples
  - Process X can open a socket on port 8888
  - Process Y can read from file /var/log/syslog
- Implementations
  - SELinux
  - AppArmor
  - and more...



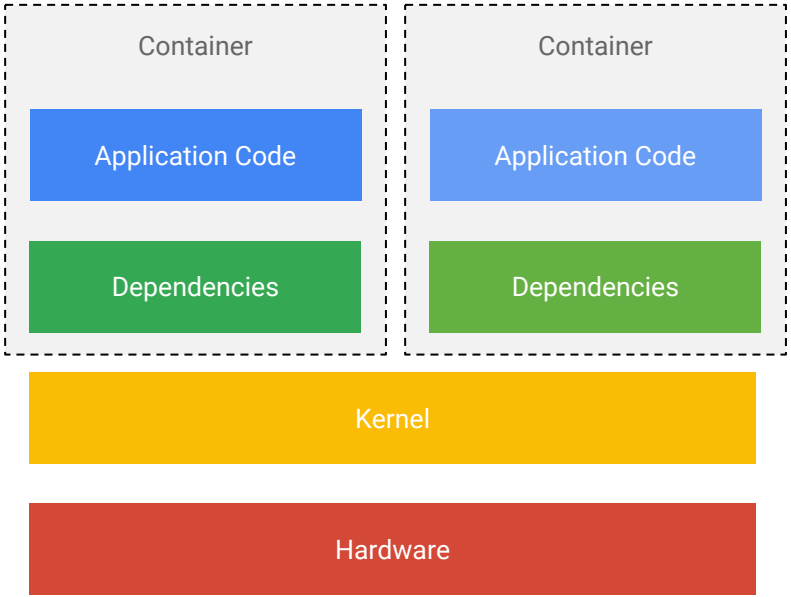
Linux Container Technology Timeline



Source: https://www.nccgroup.trust/globalassets/our-research/us/whitepapers/2016/april/ncc_group_understanding_hardening_linux_containers-10pdf

Why Developers Care?

- “Separation of code and compute”
  - Consistency across dev, test, and production
  - Consistency across bare-metal, VMs, and cloud
  - No more “it worked on my computer”
- Packaged applications
  - Agile application creation and deployment
  - Continuous Integration/Delivery
- A path to microservices
  - Introspectable
  - Isolated/loosely coupled, distributed, and elastic



So… What is Docker?

- Polished user interface to create Linux containers
- Layered image format for portability and speed and deployment
- Configuration file for creating portable images
- API for exposing
  - Container management operations
  - Container metrics
  - Available images

Introduction to Docker

Google Cloud Training and Certification

web-server.py

$> python web-server.py

```
import tornado.ioloop
import tornado.web
import socket
class MainHandler(tornado.web.RequestHandler):
    def get(self):
        self.write("Hostname: " +
socket.gethostname())
def make_app():
    return tornado.web.Application([
        (r"/", MainHandler),
    ])
if __name__ == "__main__":
    app = make_app()
    app.listen(8888)
    tornado.ioloop.IOLoop.current().start()
```

Dockerfile

```
$ docker build -t py-web-server .

$ docker run -d py-web-server
```

You can also do stuff like:

```
$ docker images
$ docker ps
$ docker logs <container id>
$ docker stop py-web-server
```

```
FROM library/python:3.6.0-alpine
RUN pip install tornado
ADD web-server.py /web-server.py
CMD ["python", "/web-server.py"]
```

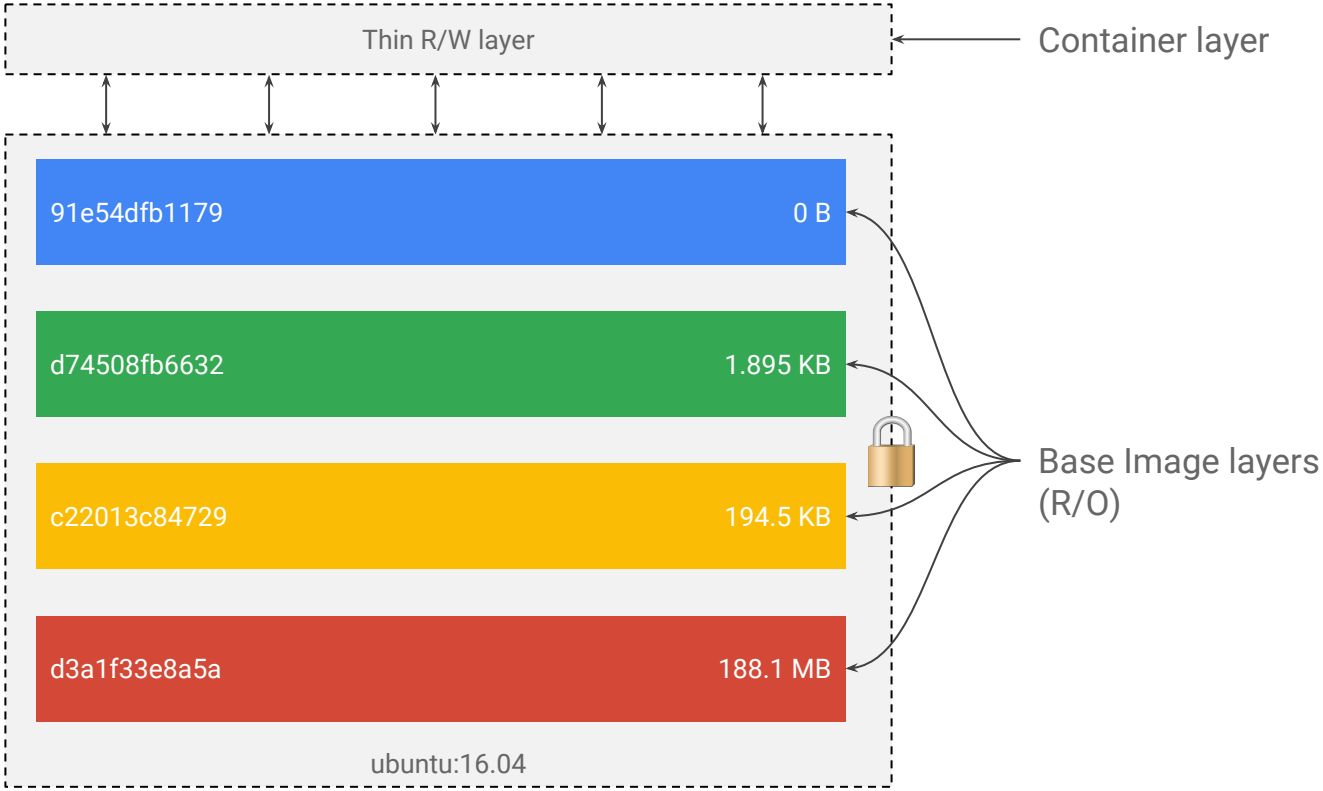
Build Container Image

- Build a container image with docker and push the image up to GCR (Google Container Registry)

```
docker build \
    -t gcr.io/$PROJECT_ID/py-web-server:v1 .

gcloud docker -- push \
    gcr.io/$PROJECT_ID/py-web-server:v1

docker run -d -p 8080:8080 \
    --name py-web-server \
    gcr.io/$PROJECT_ID/py-web-server:v1
```



Example container
(based on ubuntu:16.04 image)

Containers are the new package format

- Bundle your app with its dependencies
- Ship your application reliably
- Only update the parts that change
- Run it across infrastructures
  - Bare metal
  - Public Cloud
  - Private Cloud

Google Cloud

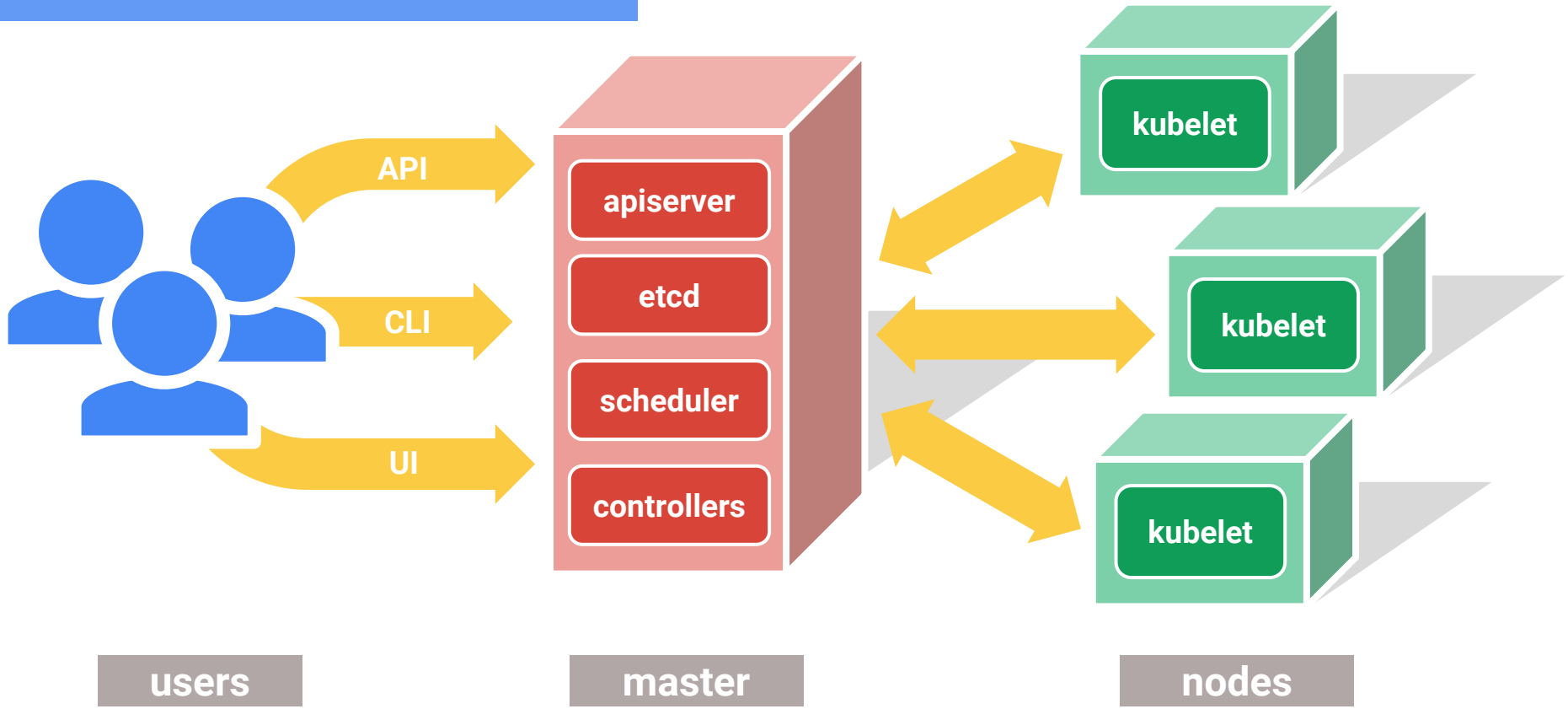
Kubernetes 101

Google Cloud Training and Certification

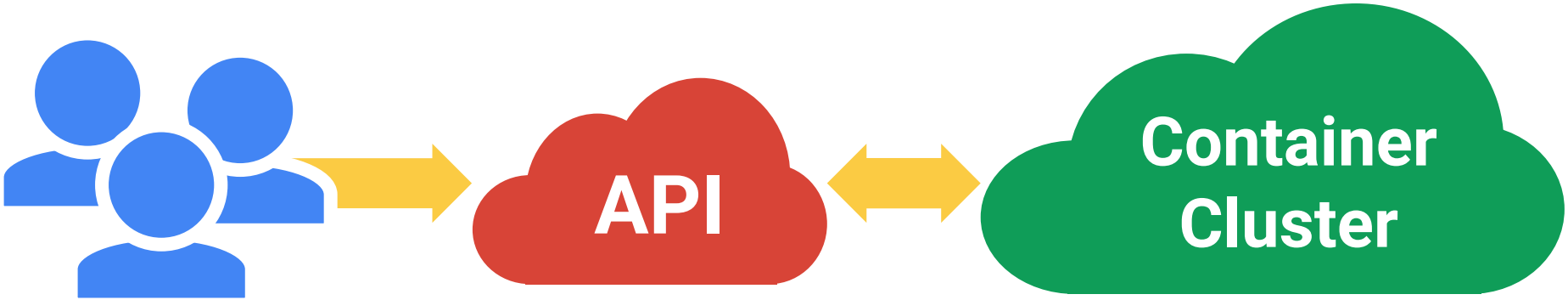
Introduction to Kubernetes

Google Cloud Training and Certification

The 10000 foot view



All users really care about

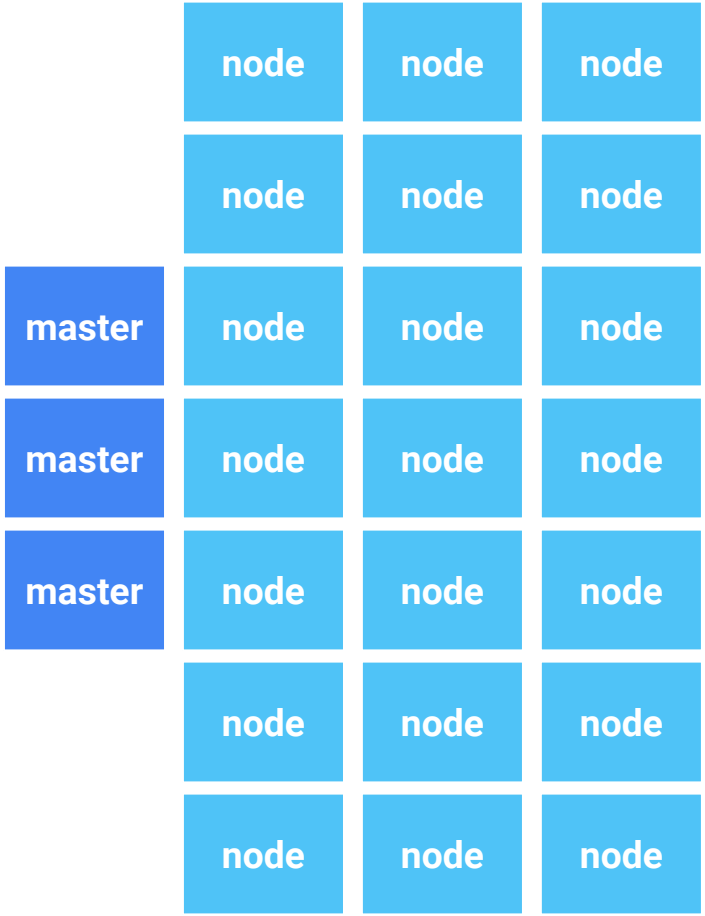


cluster

virtual machines that Kubernetes manages

cluster

cluster

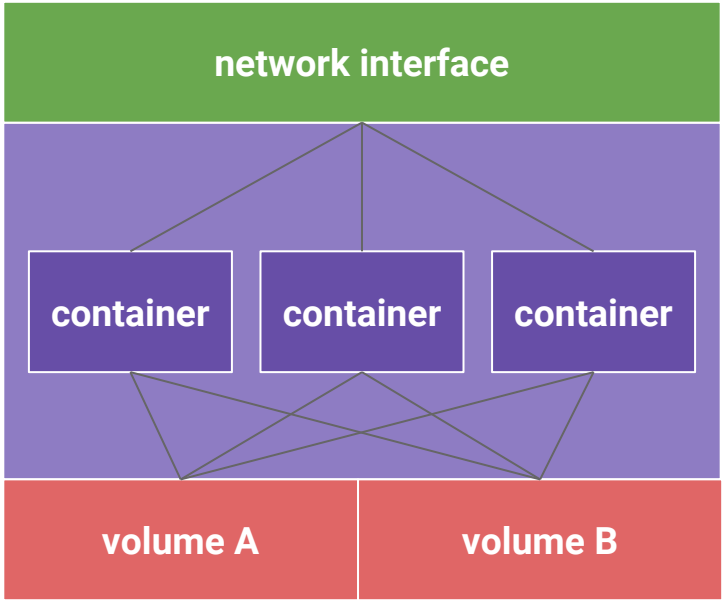


cluster

pod

group of containers sharing storage and network

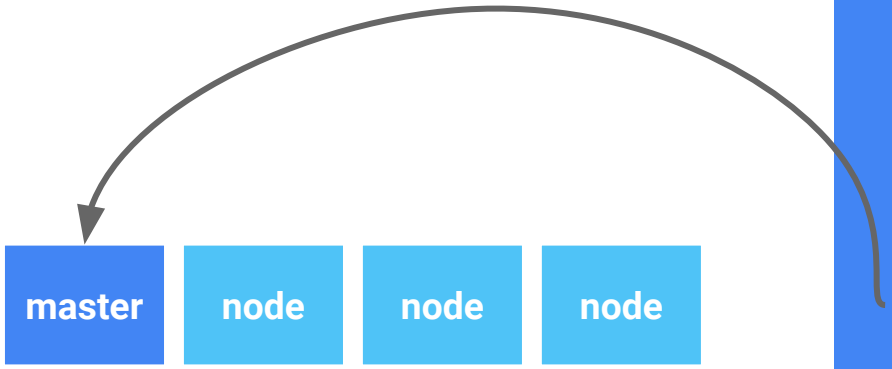
pod


network interface
container
container
container
volume A
volume B

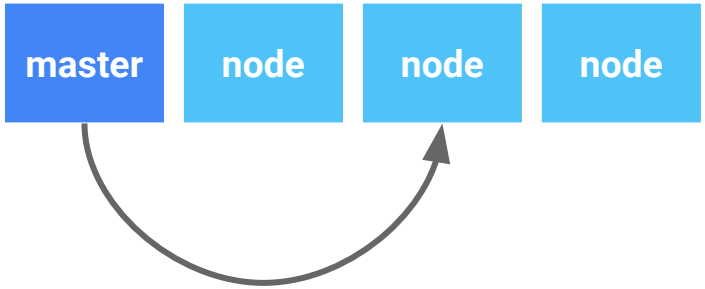
pod

```
apiVersion: v1
kind: Pod
metadata:
  name: my-app
spec:
  containers:
  - name:  my-app
    image: my-app
  - name:  nginx-ssl
    image: nginx
    ports:
    - containerPort: 80
    - containerPort: 443
```

pod.yaml



pod.yaml



pod.yaml

pod.yaml

Deployment

ensure *N* pods are running

Deployment

```
kind: Deployment
apiVersion: v1beta1
metadata:
  name: frontend
spec:
  replicas: 4
  selector:
    role: web
  template:
    metadata:
      name: web
      labels:
        role: web
    spec:
      containers:
      - name:  my-app
        image: my-app
      - name:  nginx-ssl
        image: nginx
        ports:
        - containerPort: 80
        - containerPort: 443
```


app.yaml

```
kind: Deployment
apiVersion: v1beta1
metadata:
  name: frontend
spec:
  replicas: 4
  selector:
    role: web
  template:
    metadata:
      name: web
      labels:
        role: web
    spec:
      containers:
      - name:  my-app
        image: my-app
      - name:  nginx-ssl
        image: nginx
        ports:
        - containerPort: 80
        - containerPort: 443
```

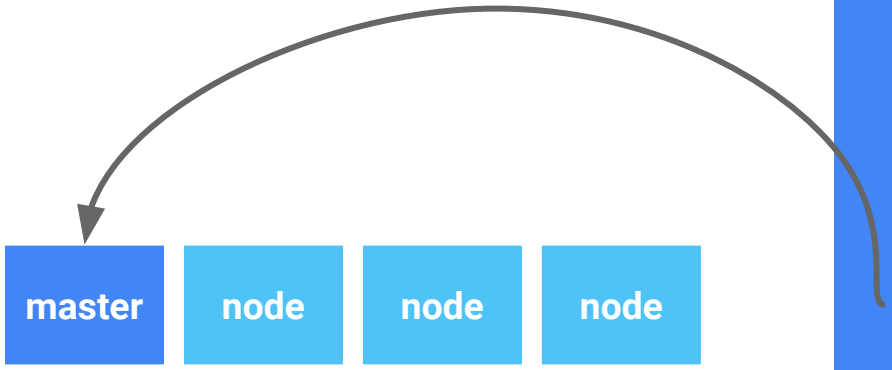
app.yaml

```
kind: Deployment
apiVersion: v1beta1
metadata:
  name: frontend
spec:
  replicas: 4
  selector:
    role: web
  template:
    metadata:
      name: web
      labels:
        role: web
    spec:
      containers:
      - name:  my-app
        image: my-app
      - name:  nginx-ssl
        image: nginx
        ports:
        - containerPort: 80
        - containerPort: 443
```

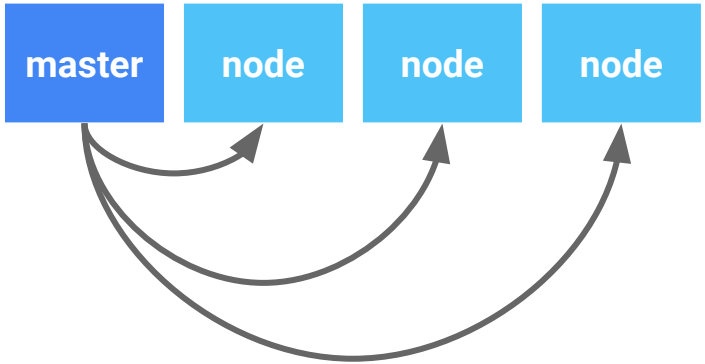
app.yaml

```
kind: Deployment
apiVersion: v1beta1
metadata:
  name: frontend
spec:
  replicas: 4
  selector:
    role: web
  template:
    metadata:
      name: web
      labels:
        role: web
    spec:
      containers:
      - name:  my-app
        image: my-app
      - name:  nginx-ssl
        image: nginx
        ports:
        - containerPort: 80
        - containerPort: 443
```

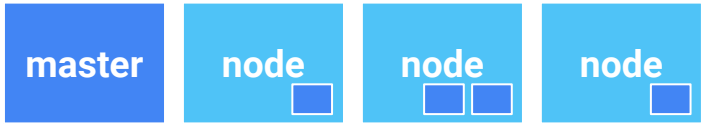

app.yaml



app.yaml



app.yaml



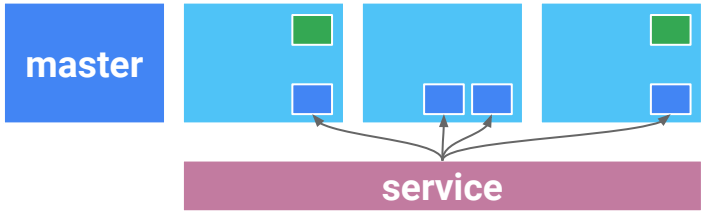
app.yaml

How do pods communicate with each other?

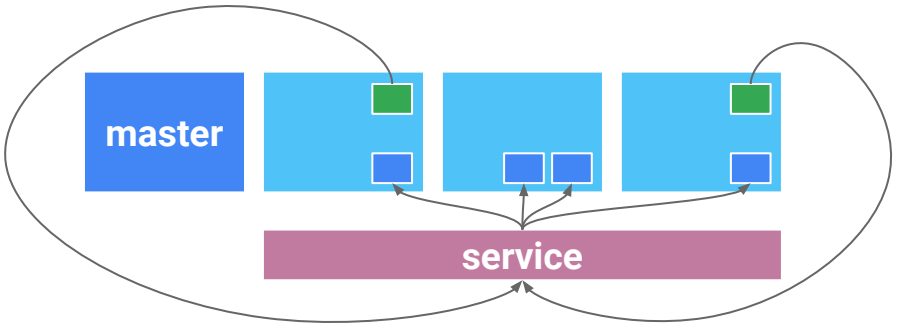
service

abstraction to communicate with pods

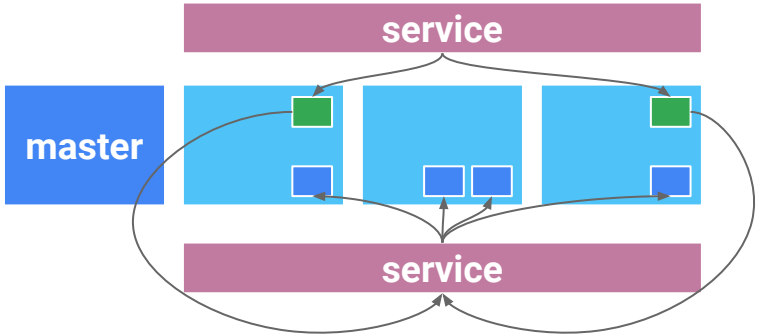
service



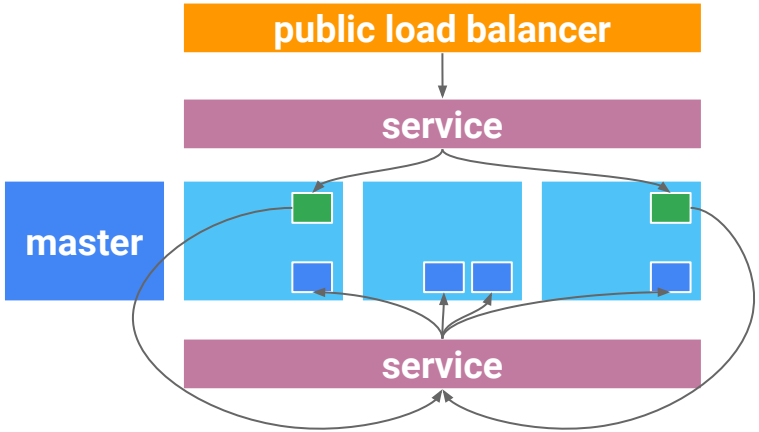
service



service



service



service

```
kind: Service
apiVersion: v1
metadata:
  name: web-frontend
spec:
  ports:
  - name: http
    port: 80
    targetPort: 80
    protocol: TCP
  selector:
    role: web
  type: LoadBalancer
```

Google

svc.yaml

```
kind: Service
apiVersion: v1
metadata:
  name: web-frontend
spec:
  ports:
  - name: http
    port: 80
    targetPort: 80
    protocol: TCP
  selector:
    role: web
  type: LoadBalancer
```

Google

svc.yaml

```
kind: Service
apiVersion: v1
metadata:
  name: web-frontend
spec:
  ports:
  - name: http
    port: 80
    targetPort: 80
    protocol: TCP
  selector:
    role: web
  type: LoadBalancer
```

Google

svc.yaml

Labels & Selectors

Labels

Arbitrary metadata

Attached to any API object

Generally represent **identity**

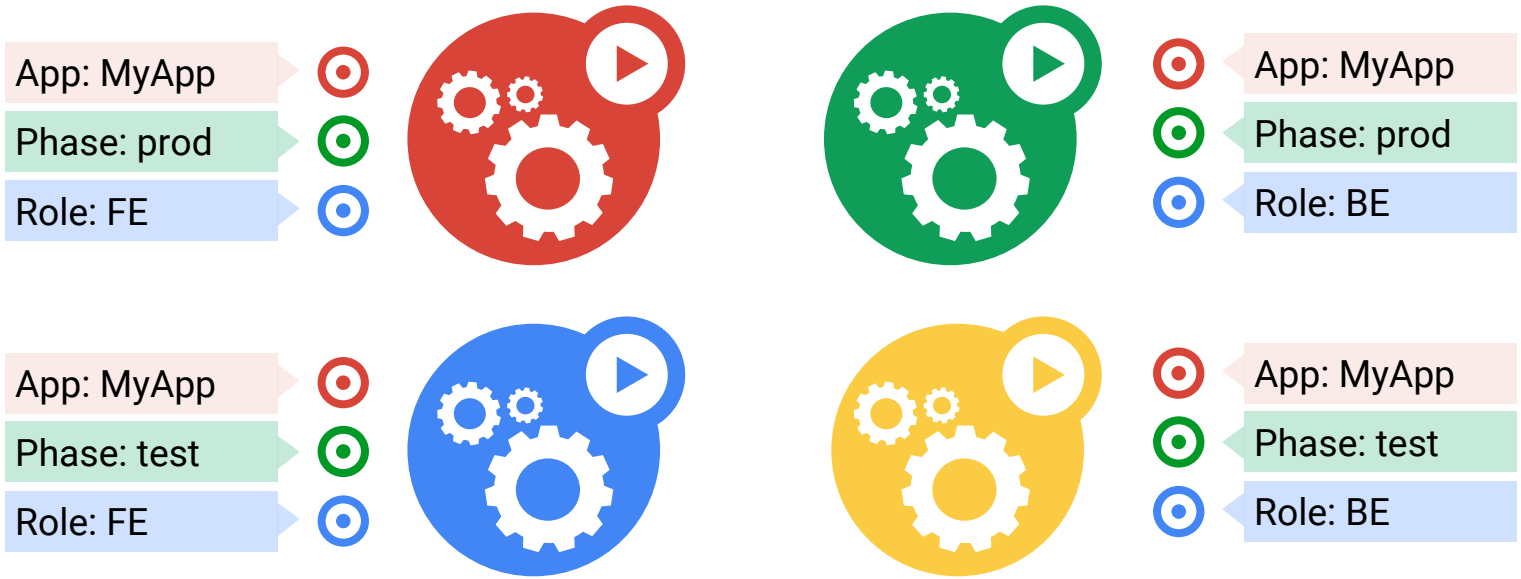
Queryable by **selectors**

- think SQL '*select ... where ...*'

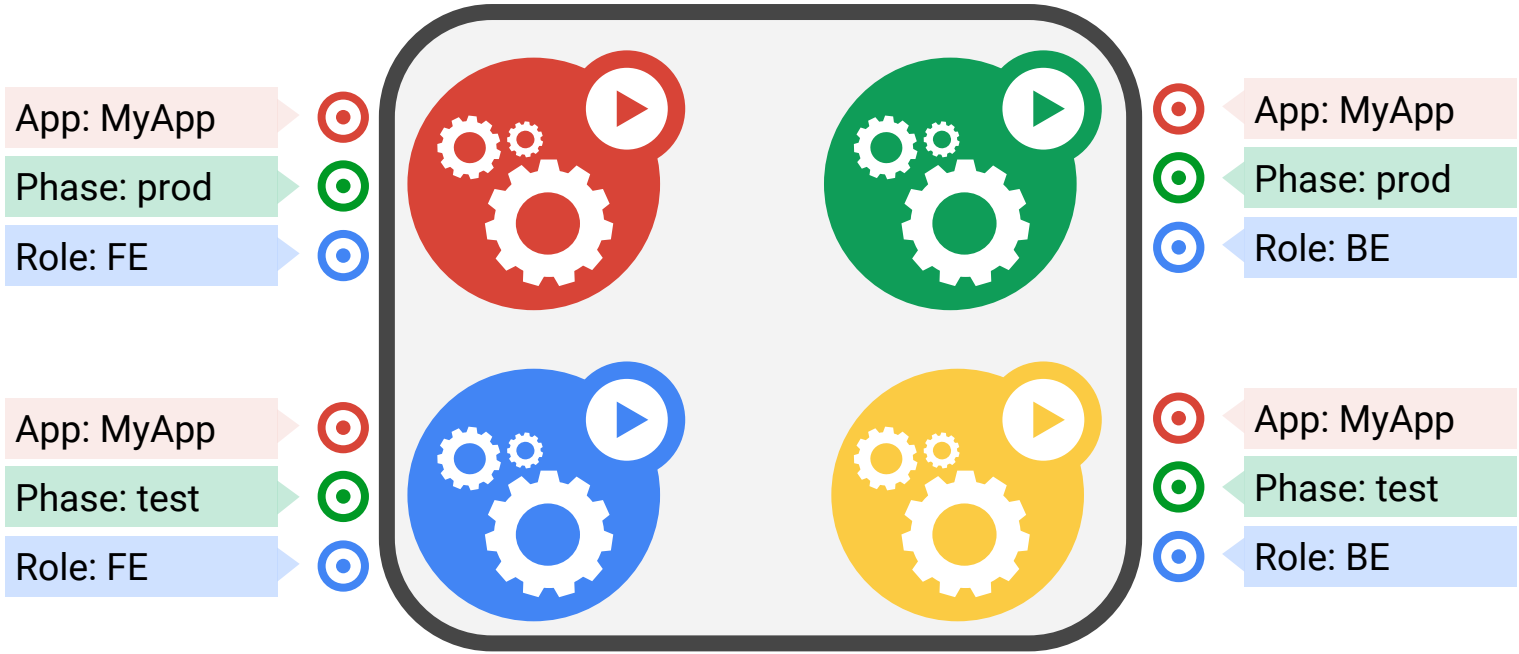
The **only** grouping mechanism

- pods under a ReplicationController
- pods in a Service
- capabilities of a node (constraints)

Selectors

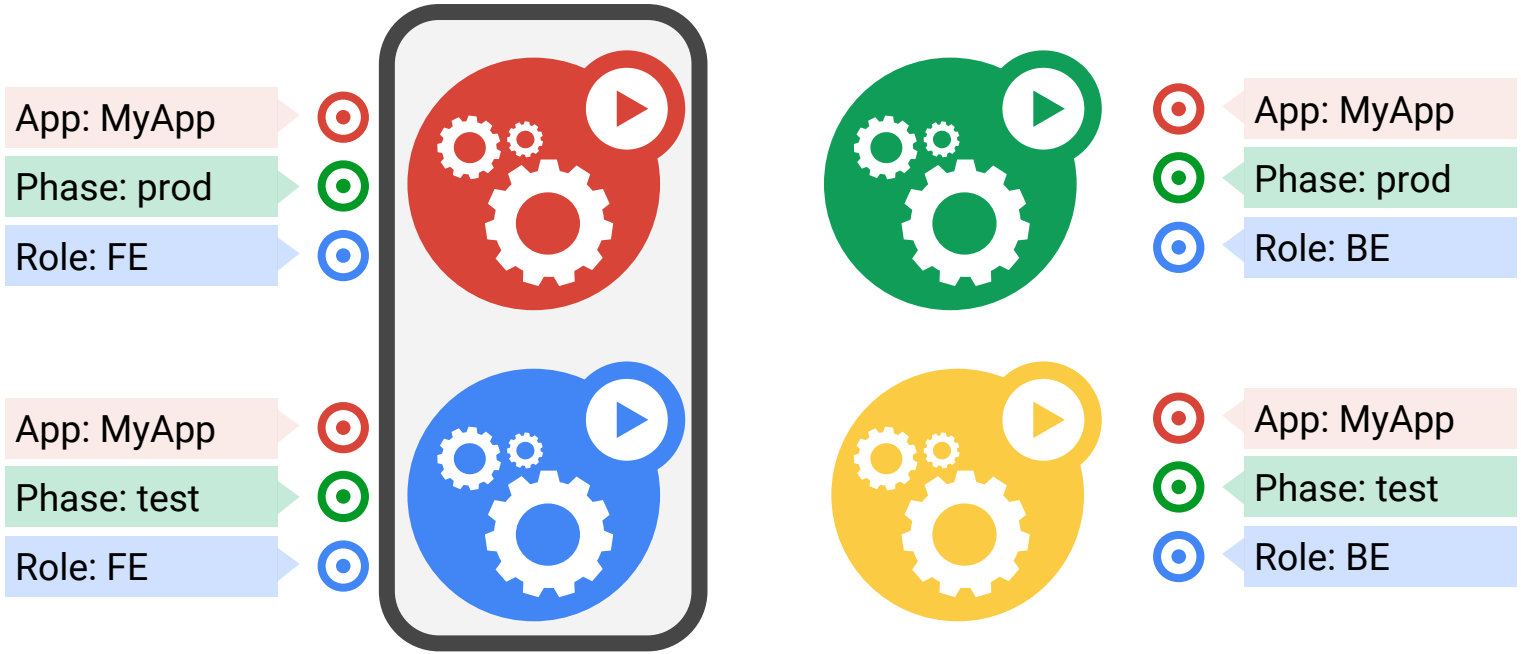

App: MyApp
Phase: prod
Role: FE
App: MyApp
Phase: prod
Role: BE
App: MyApp
Phase: test
Role: FE
App: MyApp
Phase: test
Role: BE

Selectors



App = MyApp

Selectors

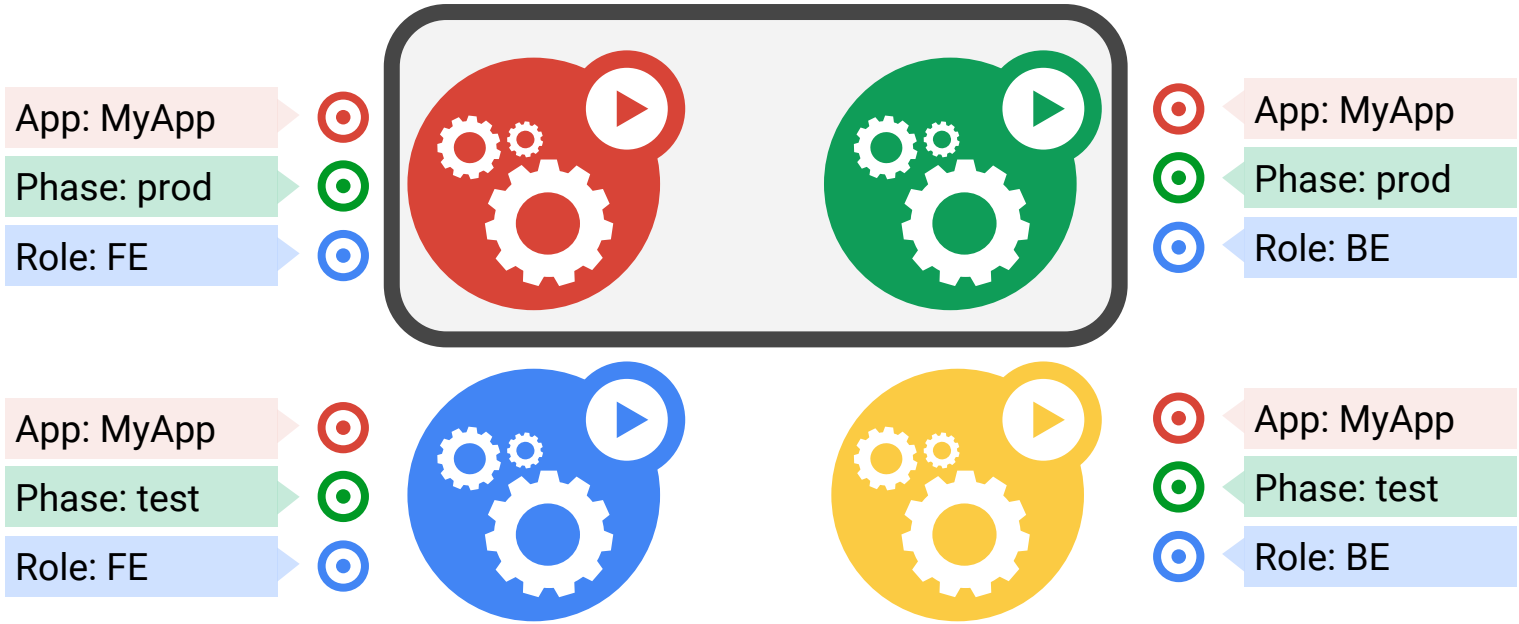


App = MyApp, Role = FE

Selectors

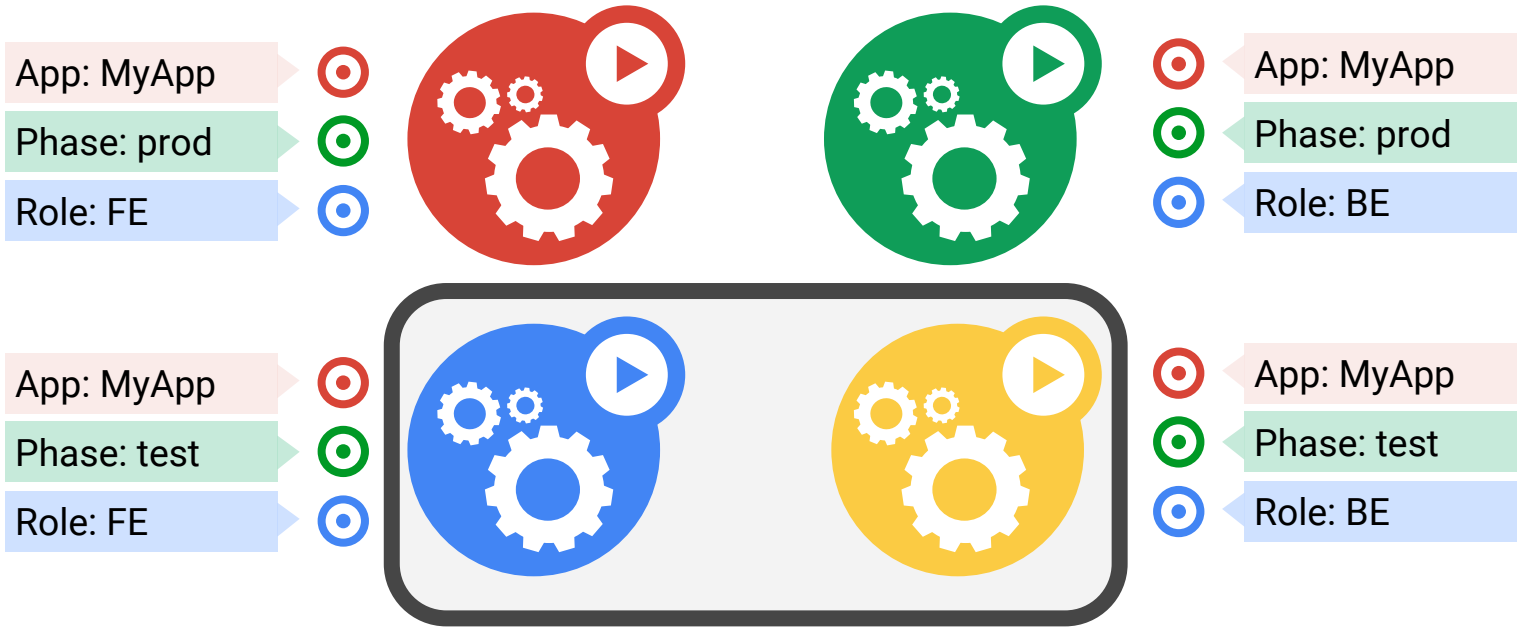
App = MyApp, Role = BE

Selectors



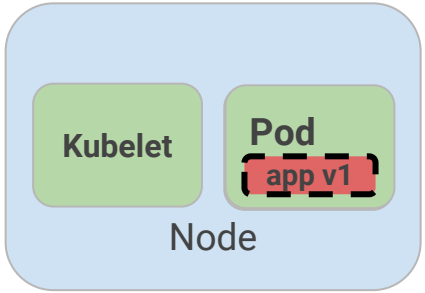
App = MyApp, Phase = prod

Selectors

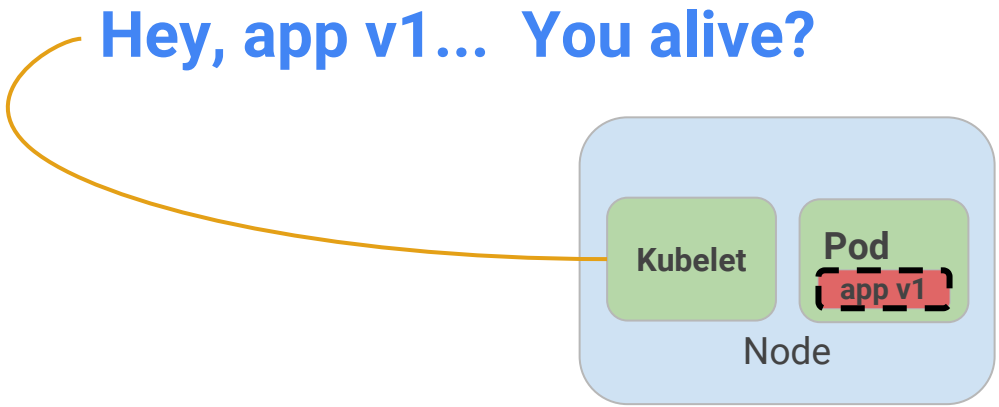

App: MyApp
Phase: prod
Role: FE
App: MyApp
Phase: prod
Role: BE
App: MyApp
Phase: test
Role: FE
App: MyApp
Phase: test
Role: BE
App = MyApp, Phase = test

Monitoring & Health Checking

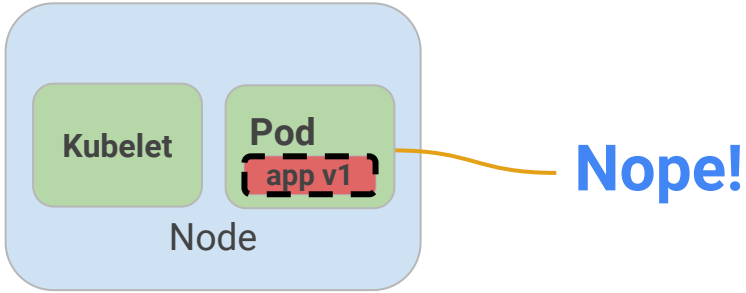
Monitoring & Health Checks (1 of 8)



Monitoring & Health Checks (2 of 8)

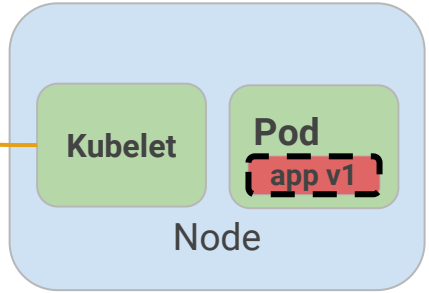


Monitoring & Health Checks (3 of 8)

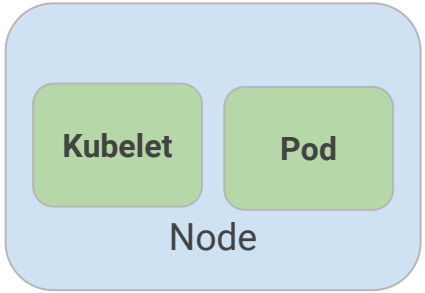


Monitoring & Health Checks (4 of 8)

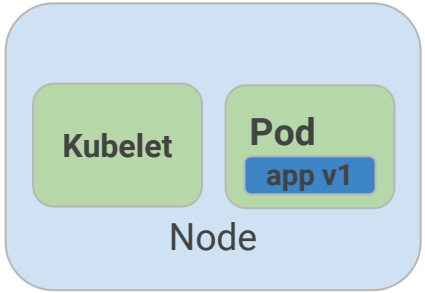
OK, then I'm going to restart you...



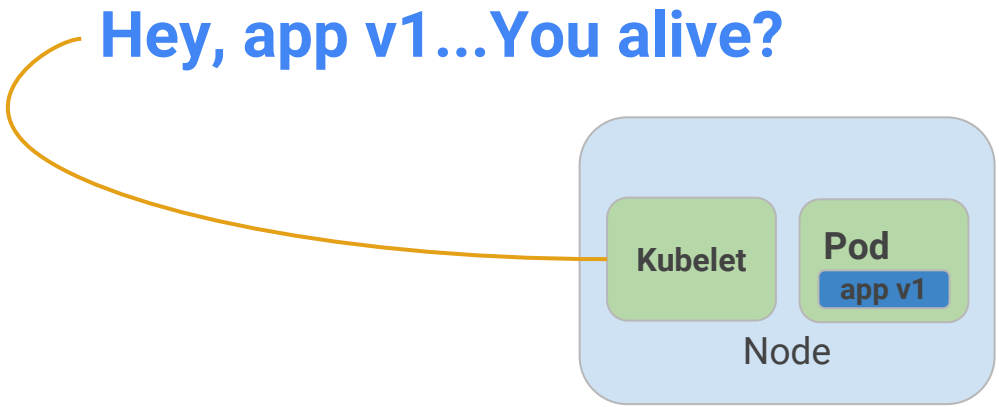
Monitoring & Health Checks (5 of 8)



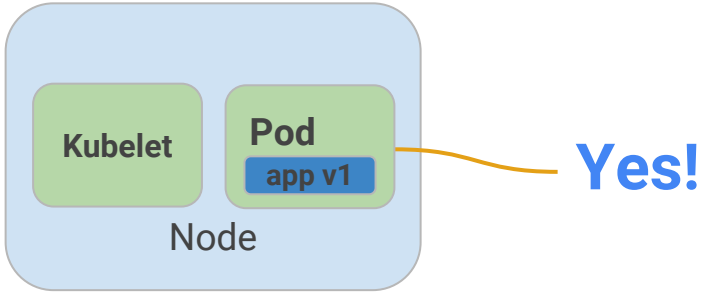
Monitoring & Health Checks (6 of 8)



Monitoring & Health Checks (7 of 8)



Monitoring & Health Checks (8 of 8)

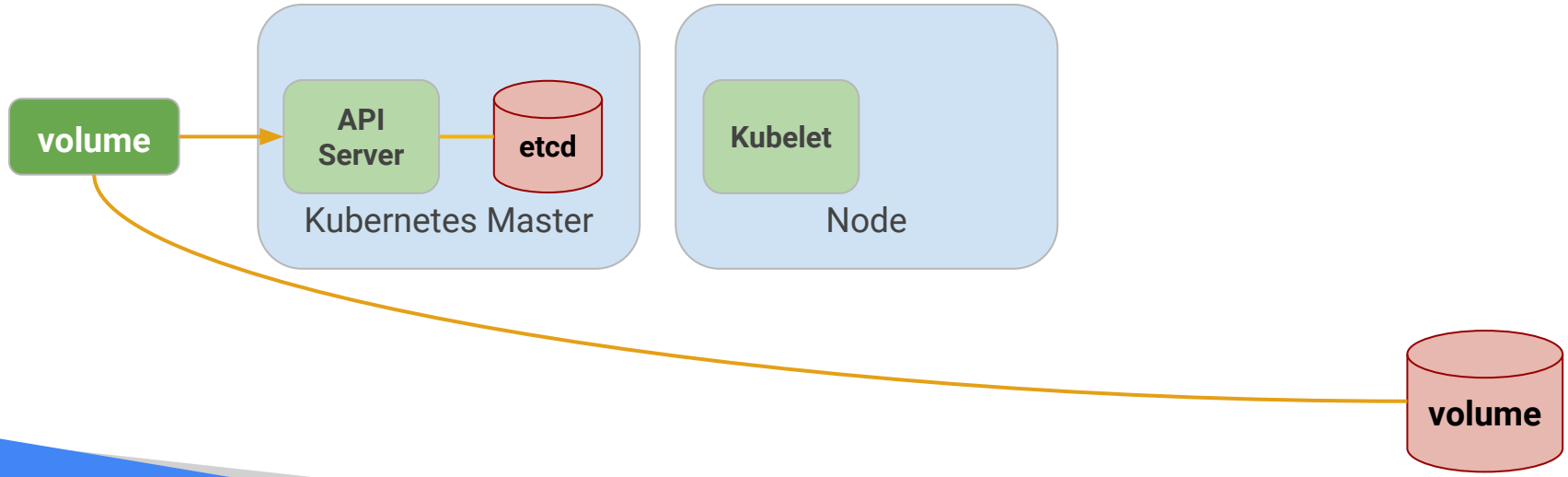


Configuration & Volumes

Volumes

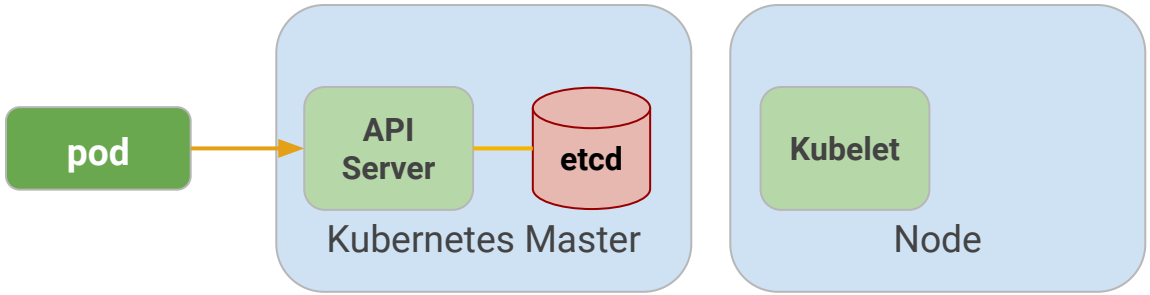
Volumes (1 of 3)

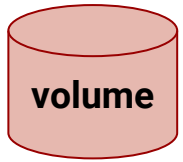
```
$ kubectl create secret generic tls-certs --from-file=tls/
```



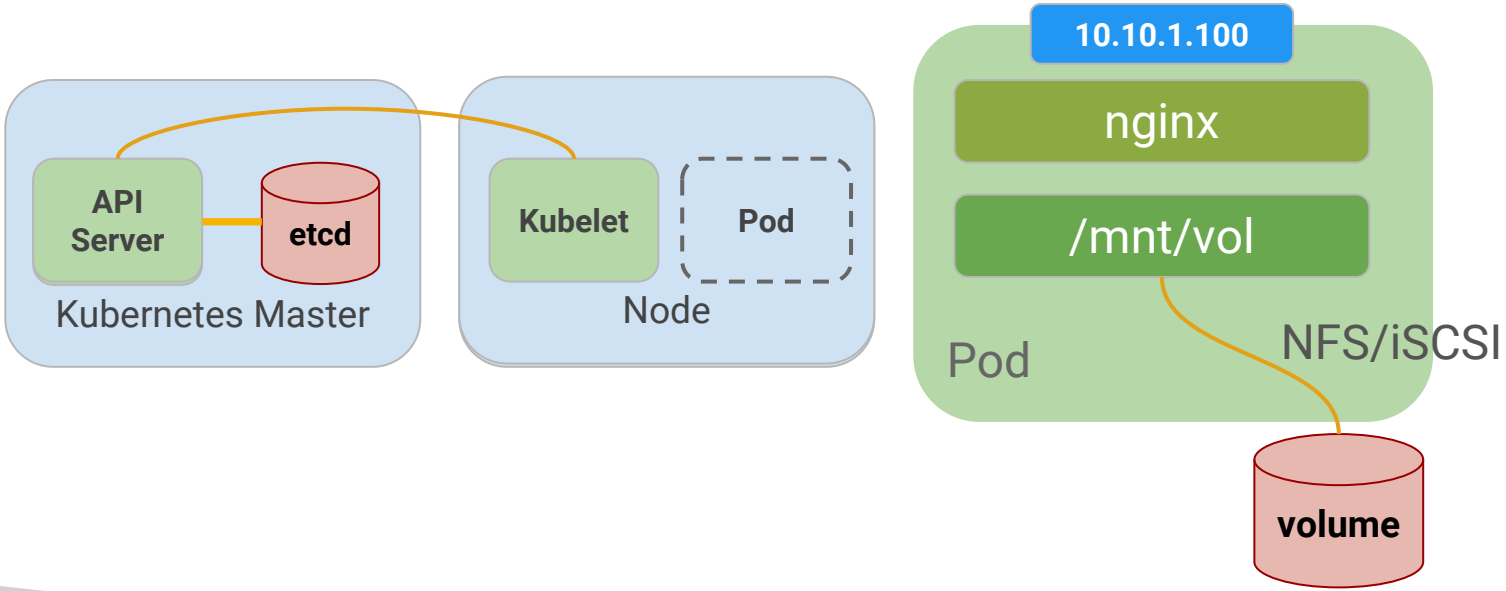
Volumes (2 of 3)

```
$ kubectl create -f pods/secure-monolith.yaml
```





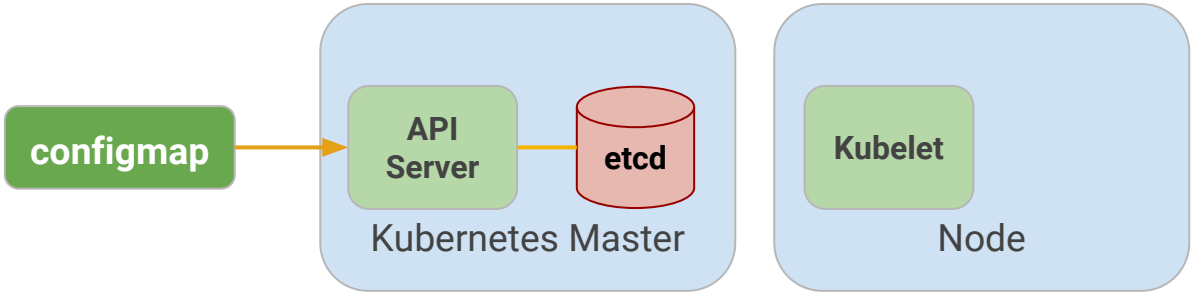
Volumes (3 of 3)



ConfigMaps

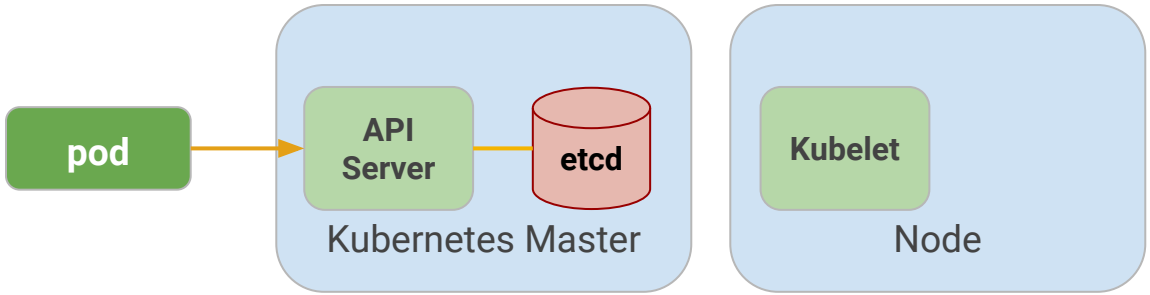
ConfigMaps (1 of 3)

```
$ kubectl create configmap my-configmap --from-file=config.conf
```

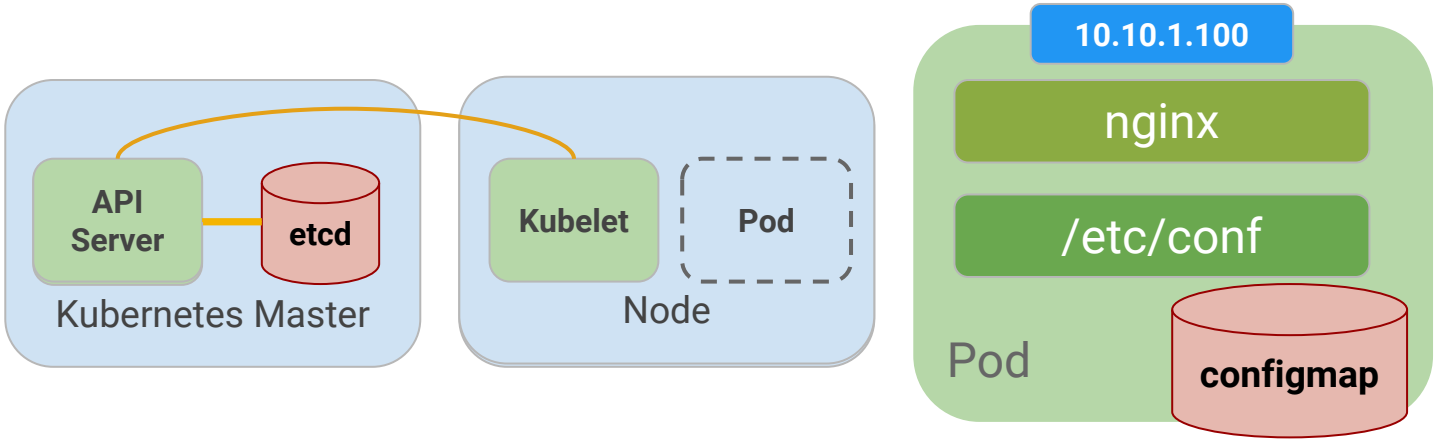


ConfigMaps (2 of 3)

```
$ kubectl create -f pods/mypod.yaml
```



ConfigMaps (3 of 3)



Google Cloud

Deploying to Kubernetes

Google Cloud Training and Certification

Introduction to Deployments

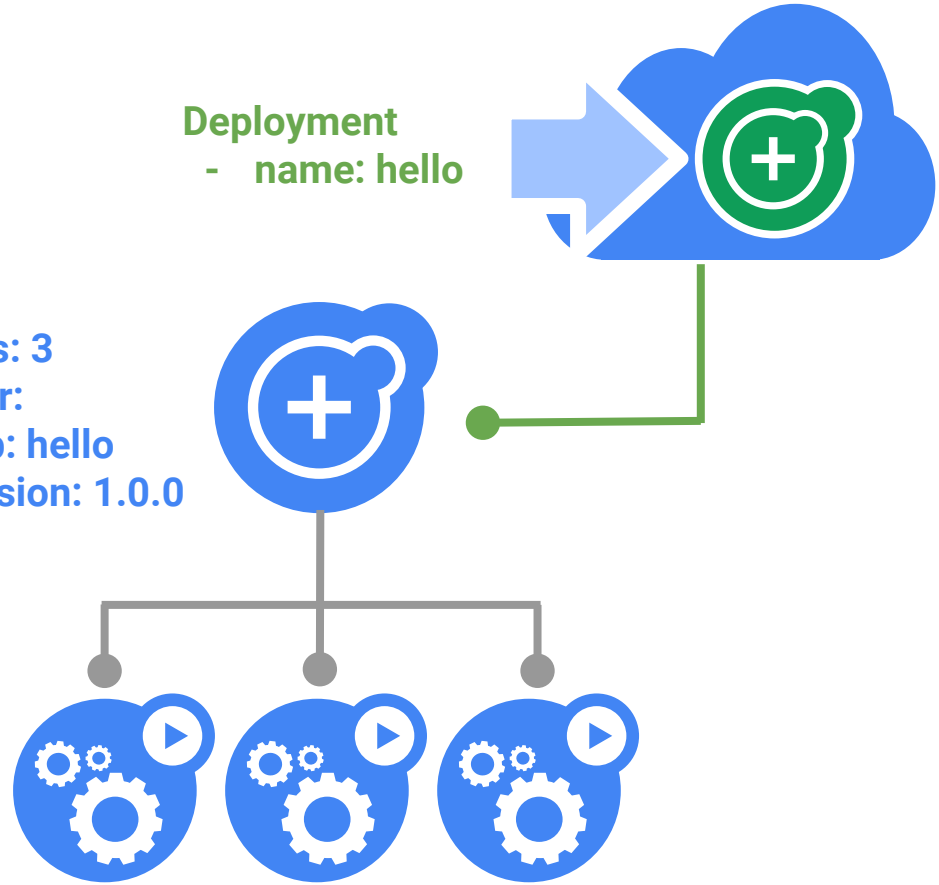
Google Cloud Training and Certification

Deployments

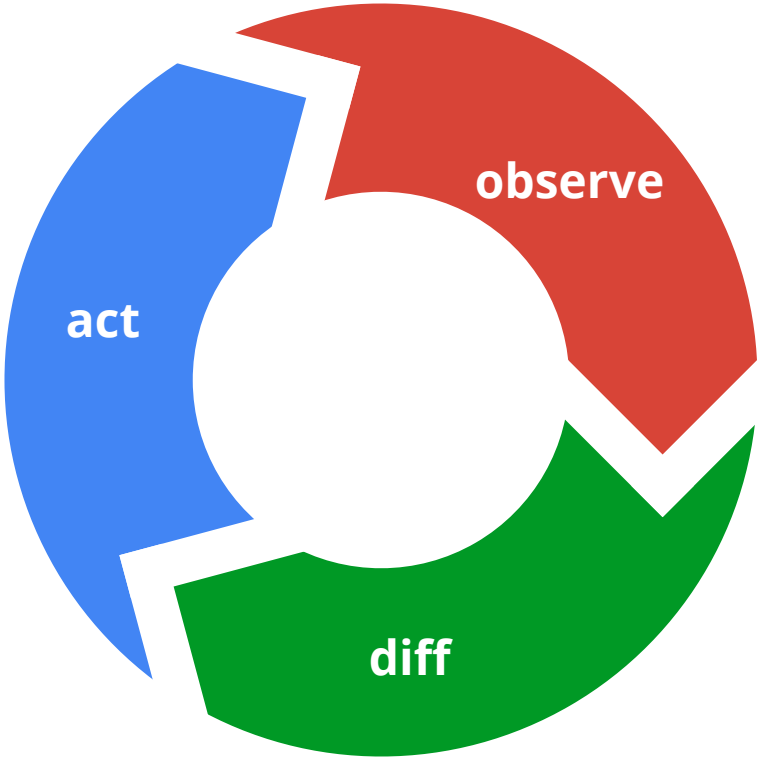
Deployment
- name: hello

ReplicaSet
- replicas: 3
- selector:
  - app: hello
  - version: 1.0.0

Pod
- containers:
  - name: hello
  - image: hello



Deployments

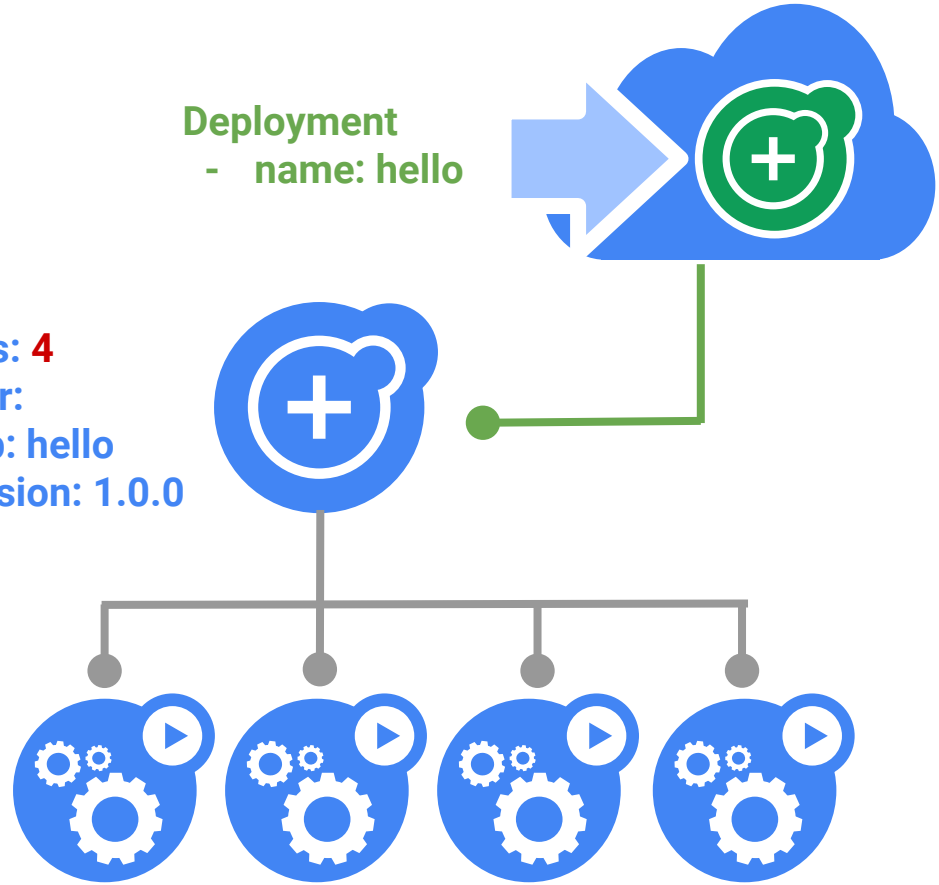


Deployments

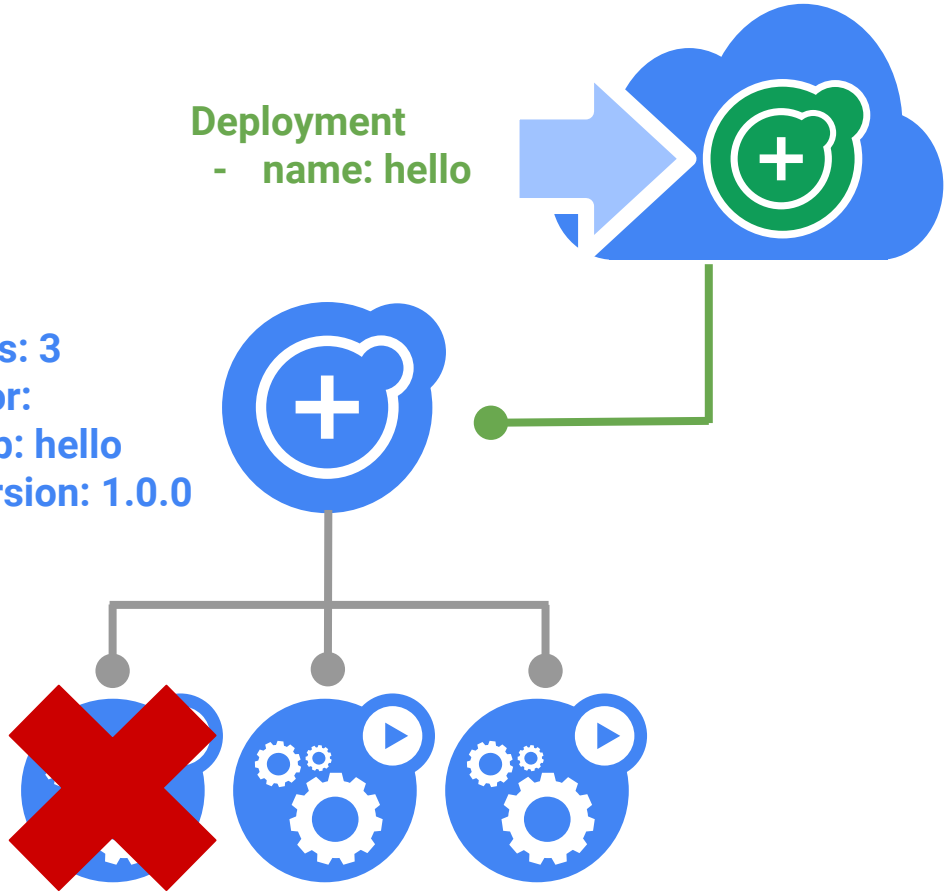
Deployment
- name: hello

ReplicaSet
- replicas: 4
- selector:
  - app: hello
  - version: 1.0.0

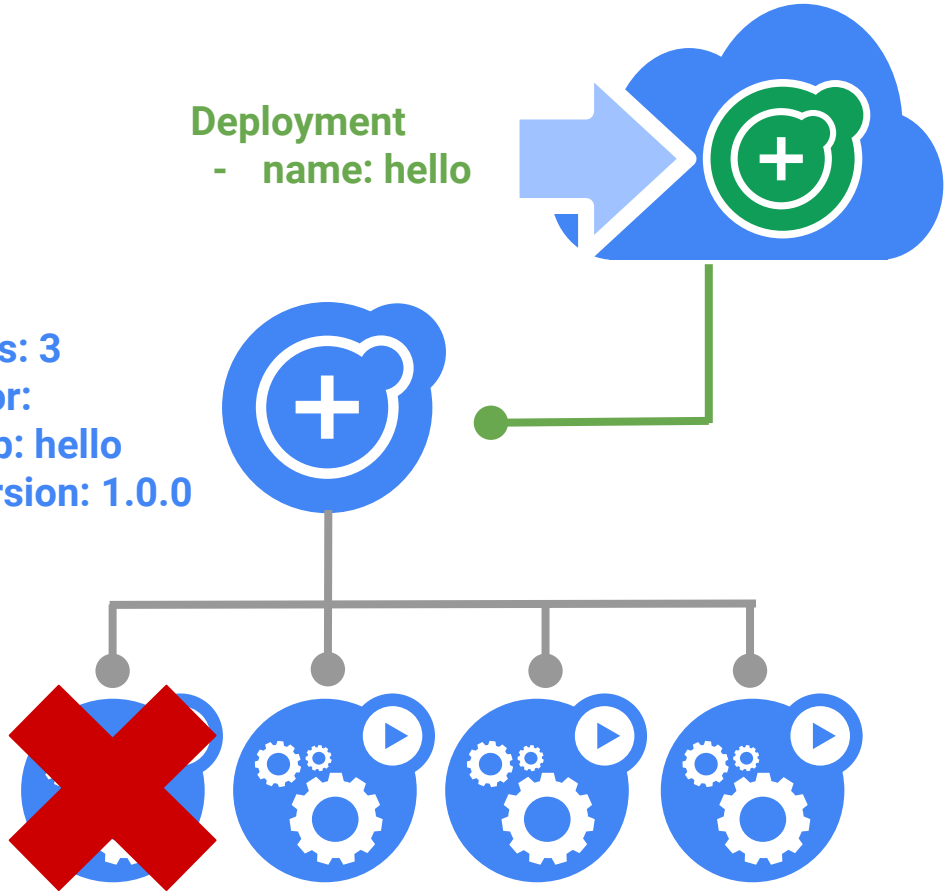
Pod
- containers:
  - name: hello
  - image: hello



Deployments

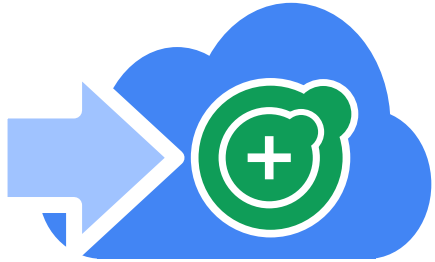


Deployments



Rolling Updates

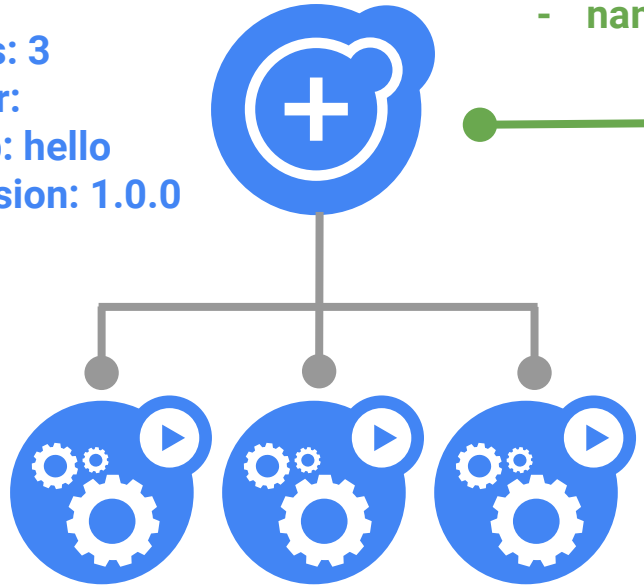
Rolling Updates (1 of 8)



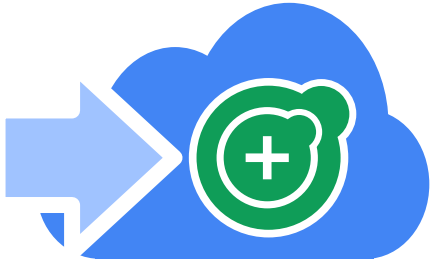
kubectl apply ...

Deployment
- name: hello

ReplicaSet
- replicas: 3
- selector:
  - app: hello
  - version: 1.0.0

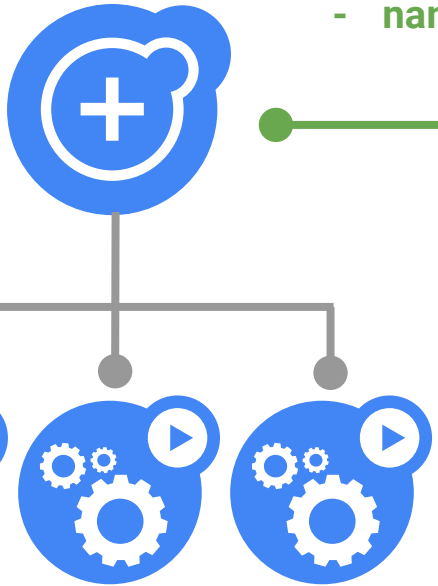


Rolling Updates (2 of 8)



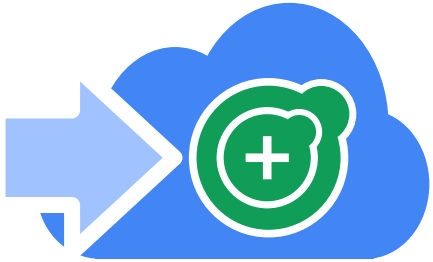
Deployment
- **name: hello**

ReplicaSet
- **replicas: 3**
- **selector:**
  - **app: hello**
  - **version: 1.0.0**



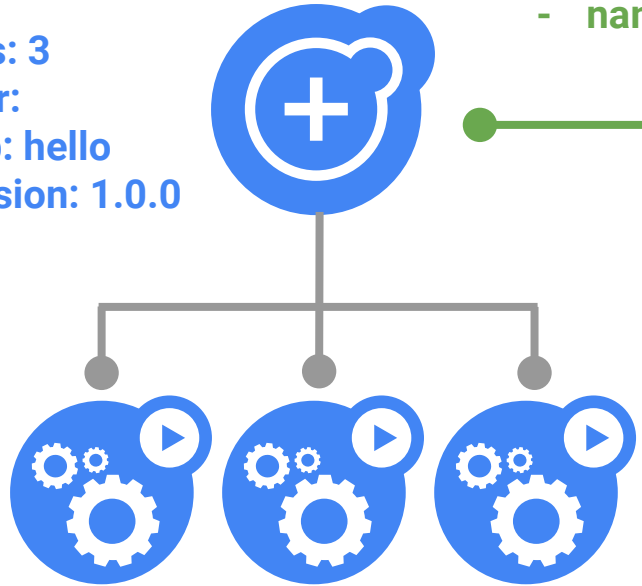
ReplicaSet
- **replicas: 0**
- **selector:**
  - **app: hello**
  - **version: 2.0.0**

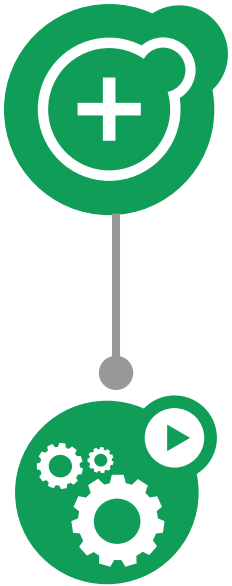
Rolling Updates (3 of 8)



Deployment
- **name: hello**

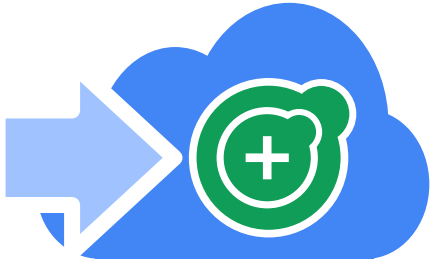
ReplicaSet
- **replicas: 3**
- **selector:**
  - **app: hello**
  - **version: 1.0.0**





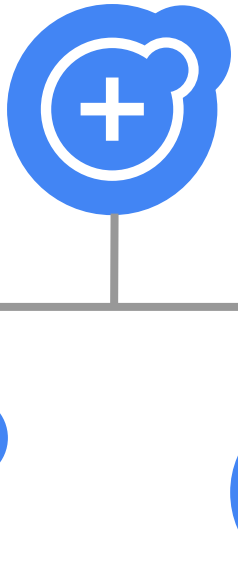
ReplicaSet
- **replicas: 1**
- **selector:**
  - **app: hello**
  - **version: 2.0.0**

Rolling Updates (4 of 8)

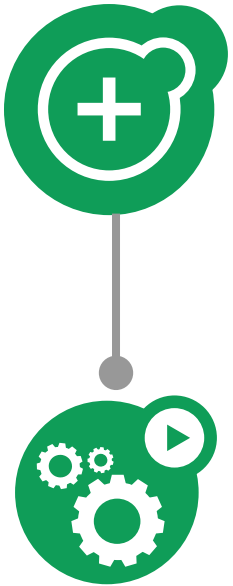


Deployment
- **name: hello**

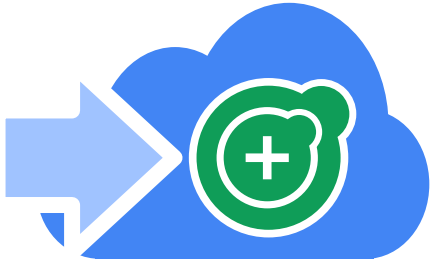
ReplicaSet
- **replicas: 2**
- **selector:**
  - **app: hello**
  - **version: 1.0.0**

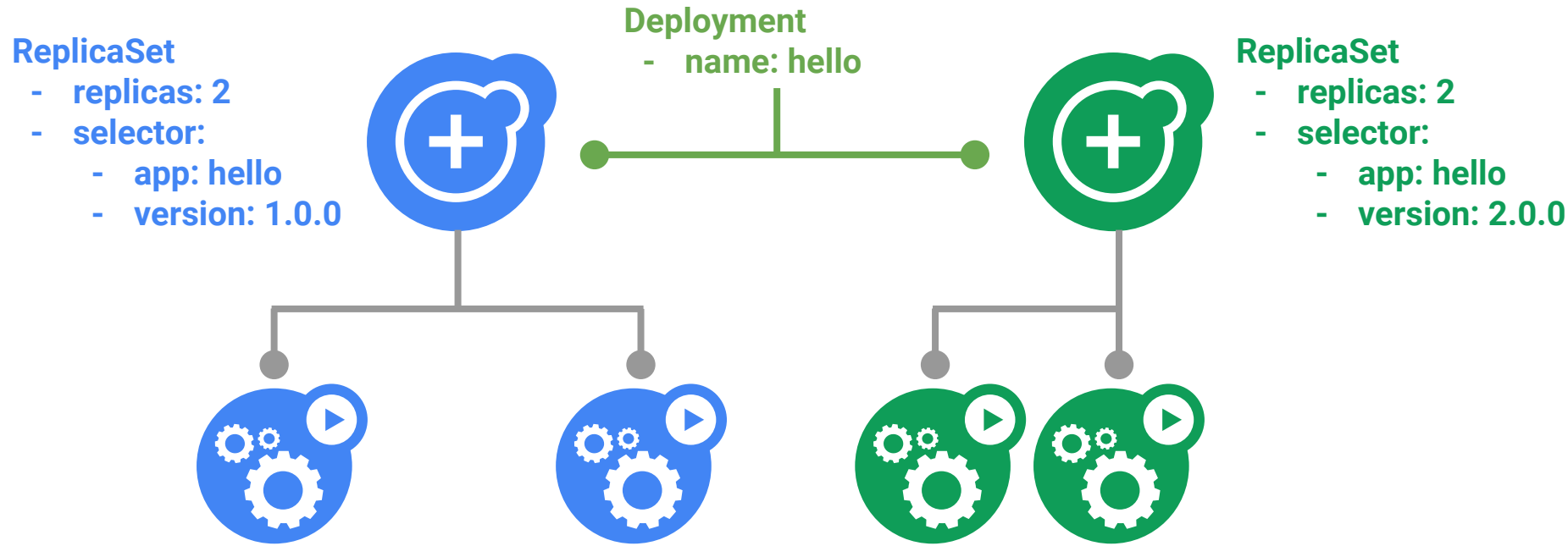


ReplicaSet
- **replicas: 1**
- **selector:**
  - **app: hello**
  - **version: 2.0.0**

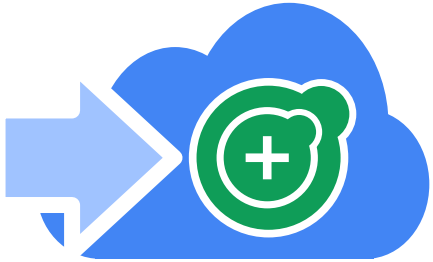


Rolling Updates (5 of 8)





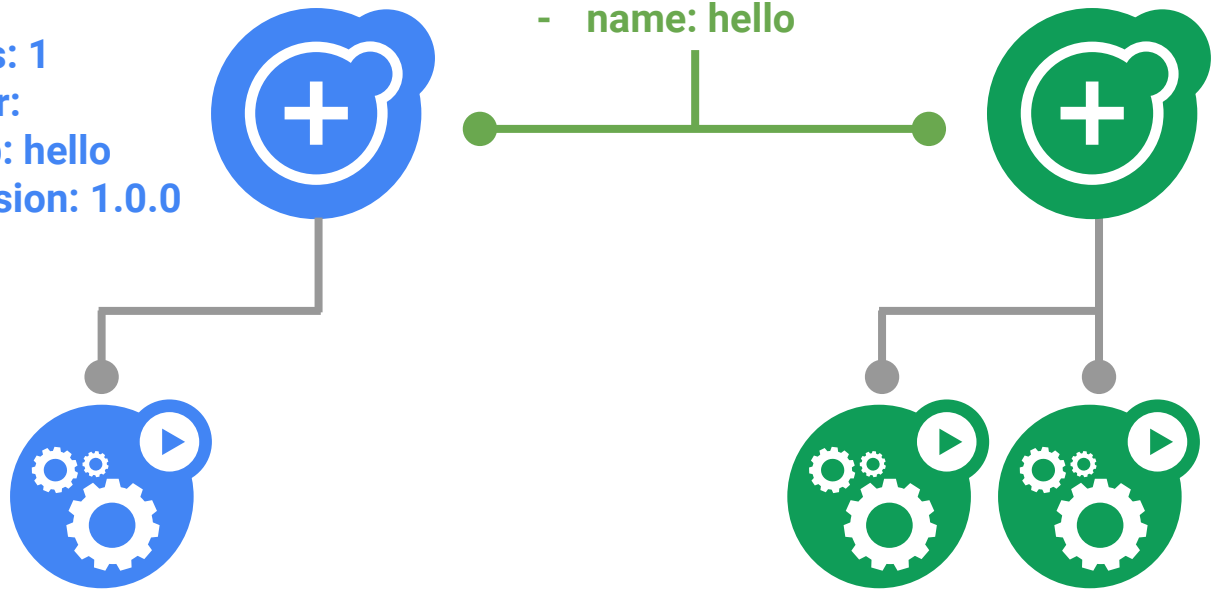
Rolling Updates (6 of 8)



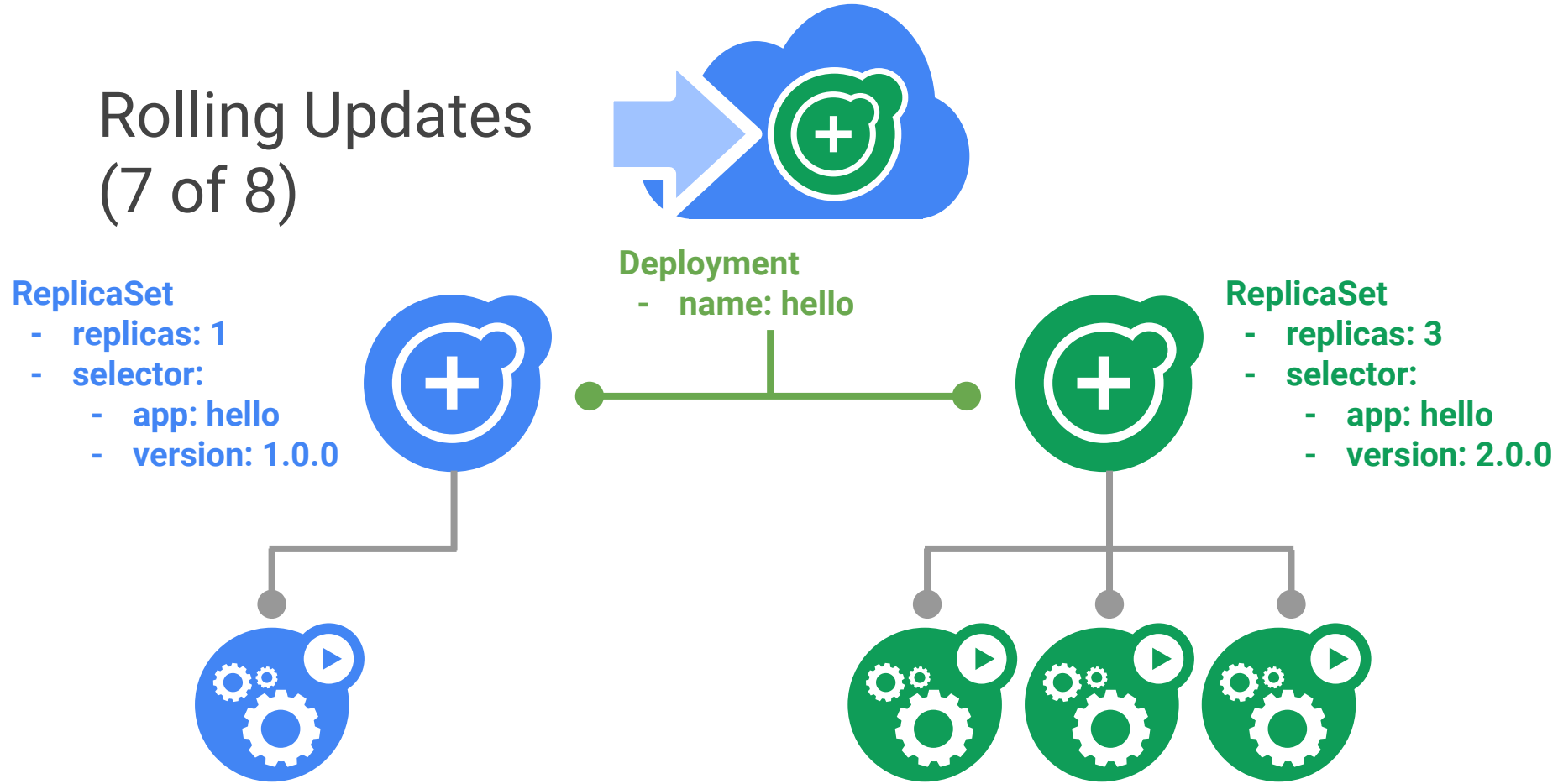
Deployment
- name: hello

ReplicaSet
- replicas: 1
- selector:
  - app: hello
  - version: 1.0.0

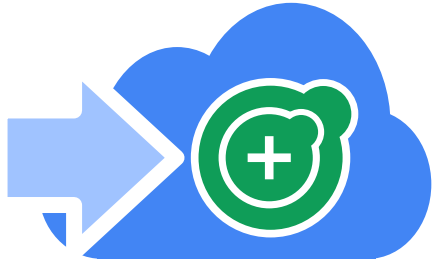
ReplicaSet
- replicas: 2
- selector:
  - app: hello
  - version: 2.0.0



Rolling Updates (7 of 8)



Rolling Updates (8 of 8)



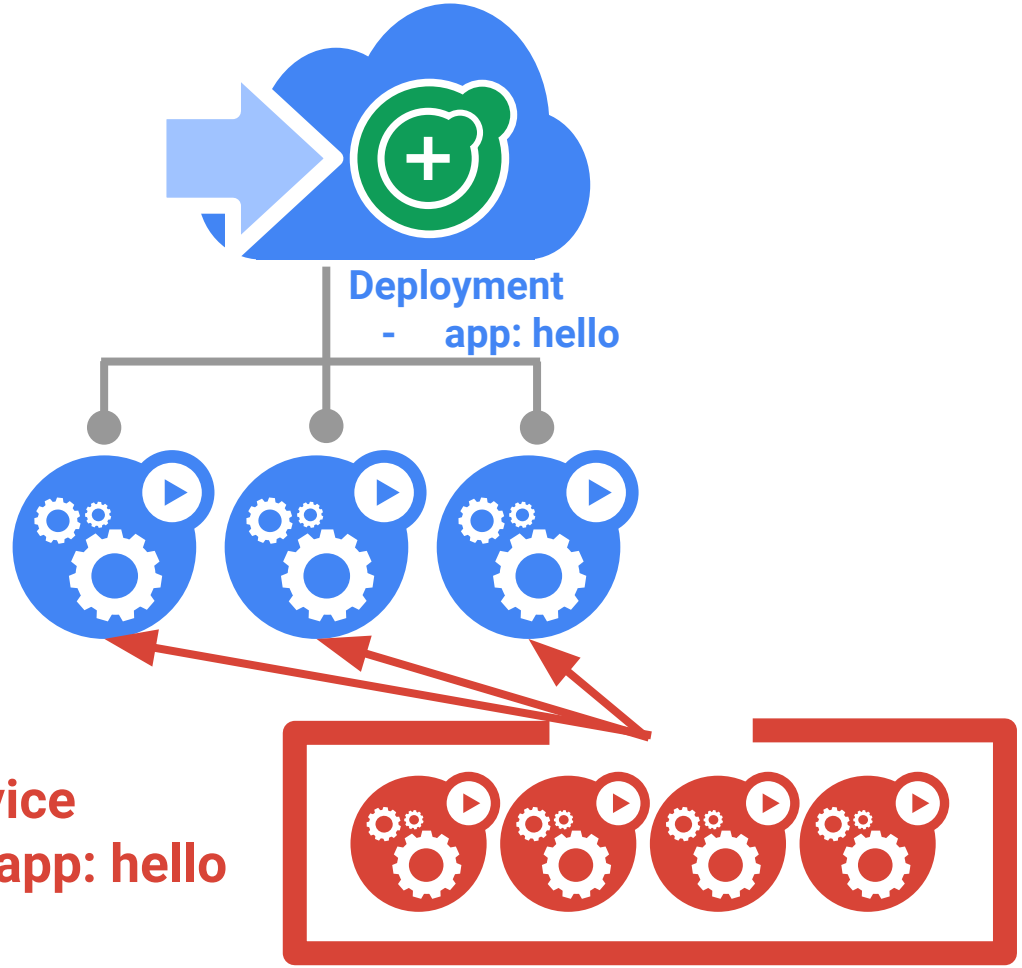
ReplicaSet
- **replicas: 0**
- **selector:**
  - **app: hello**
  - **version: 1.0.0**

Deployment
- **name: hello**

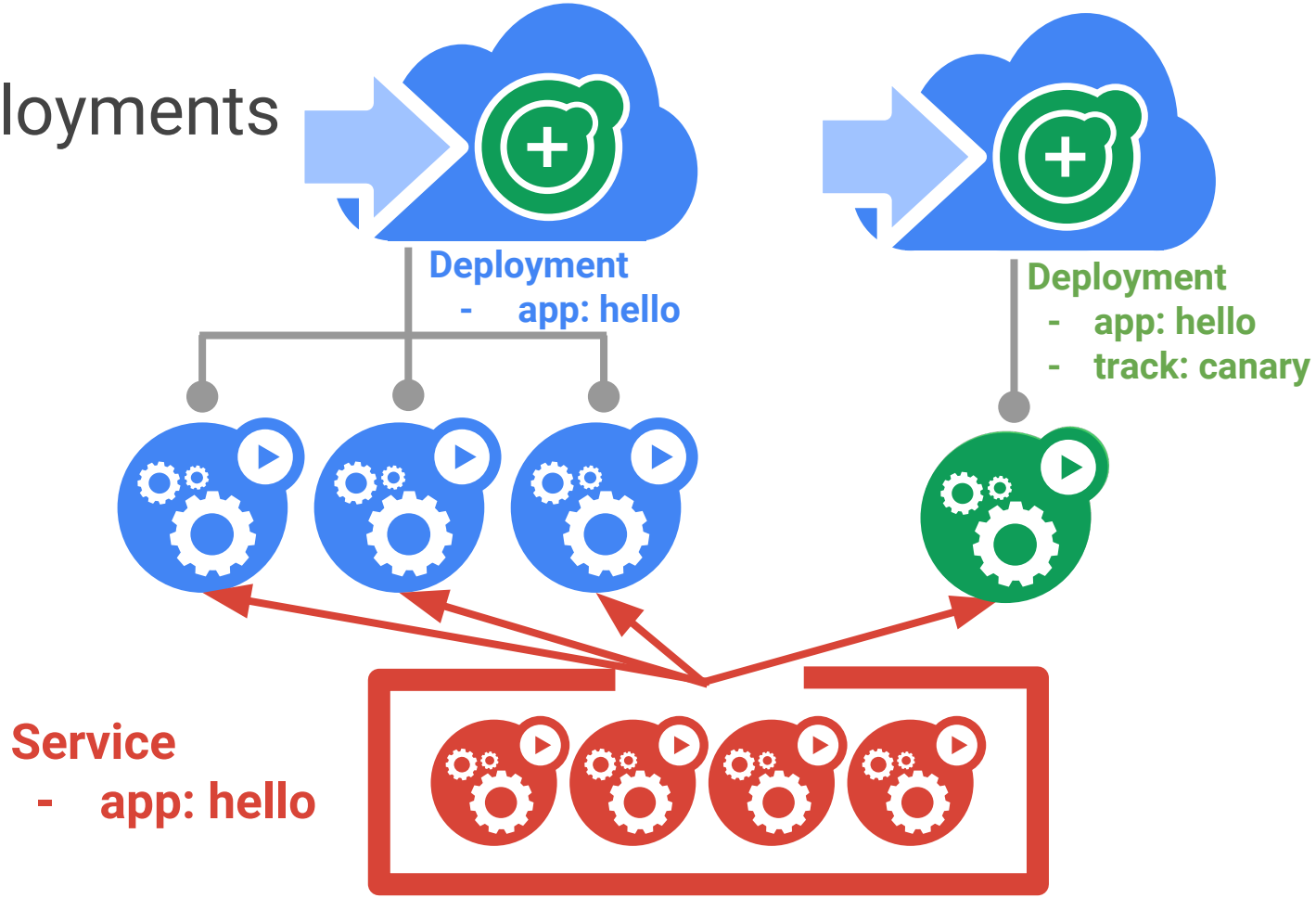
ReplicaSet
- **replicas: 3**
- **selector:**
  - **app: hello**
  - **version: 2.0.0**

Canary Deployments

Canary Deployments (1 of 2)

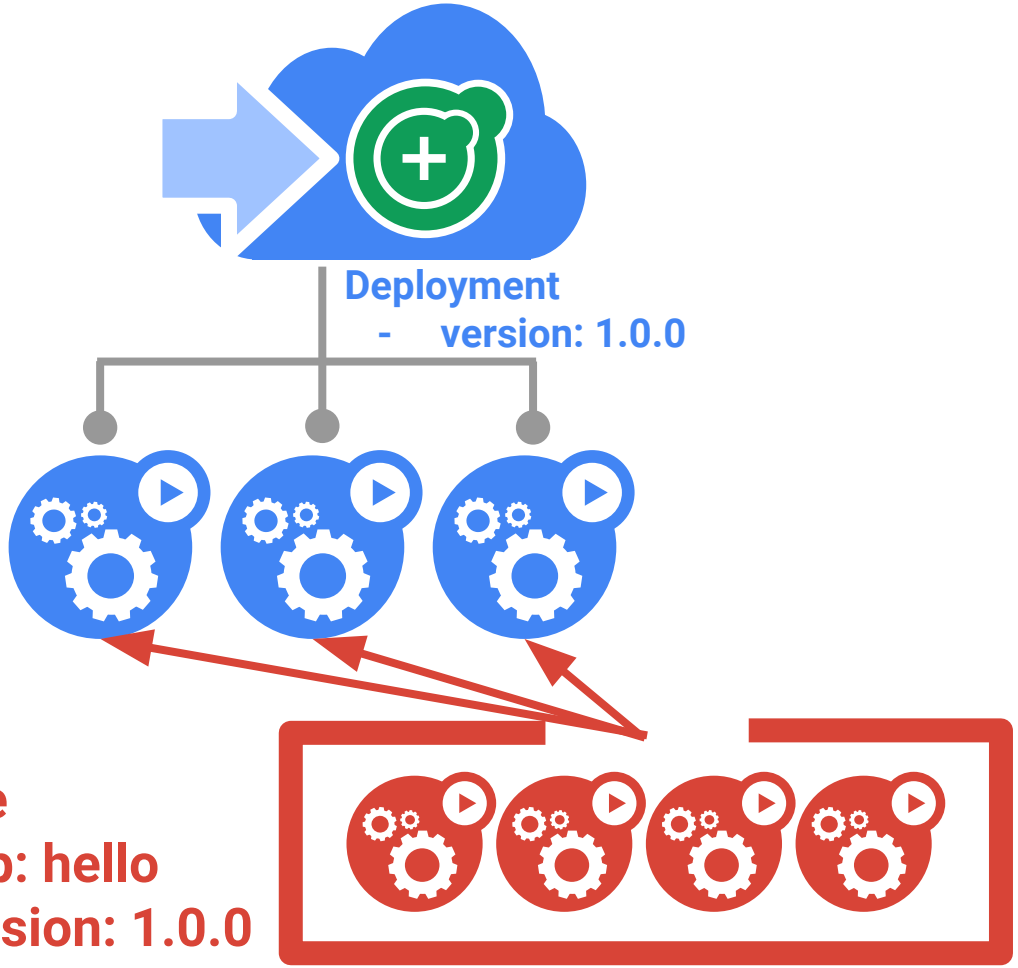


Canary Deployments (2 of 2)

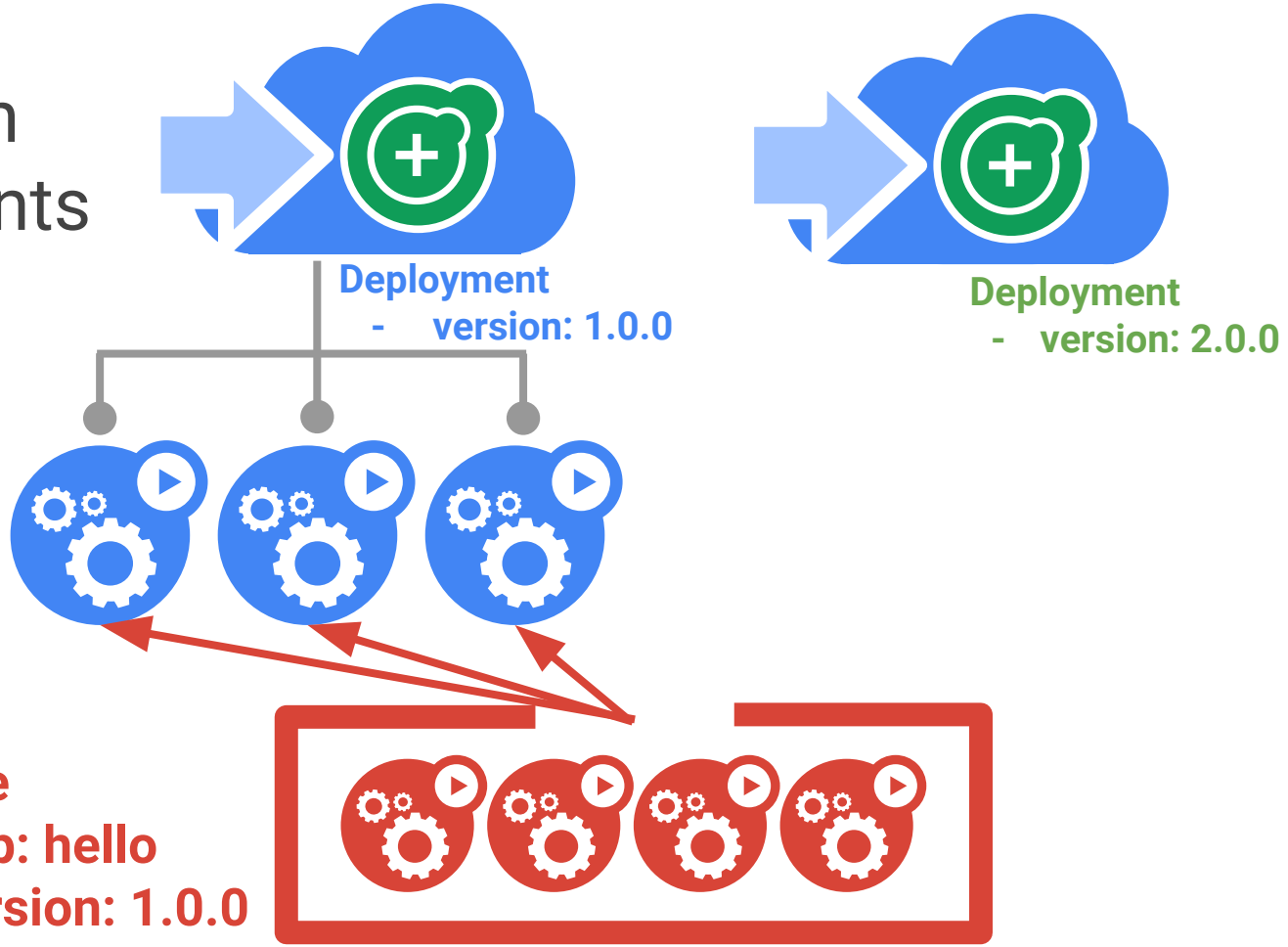


Blue-Green Deployments

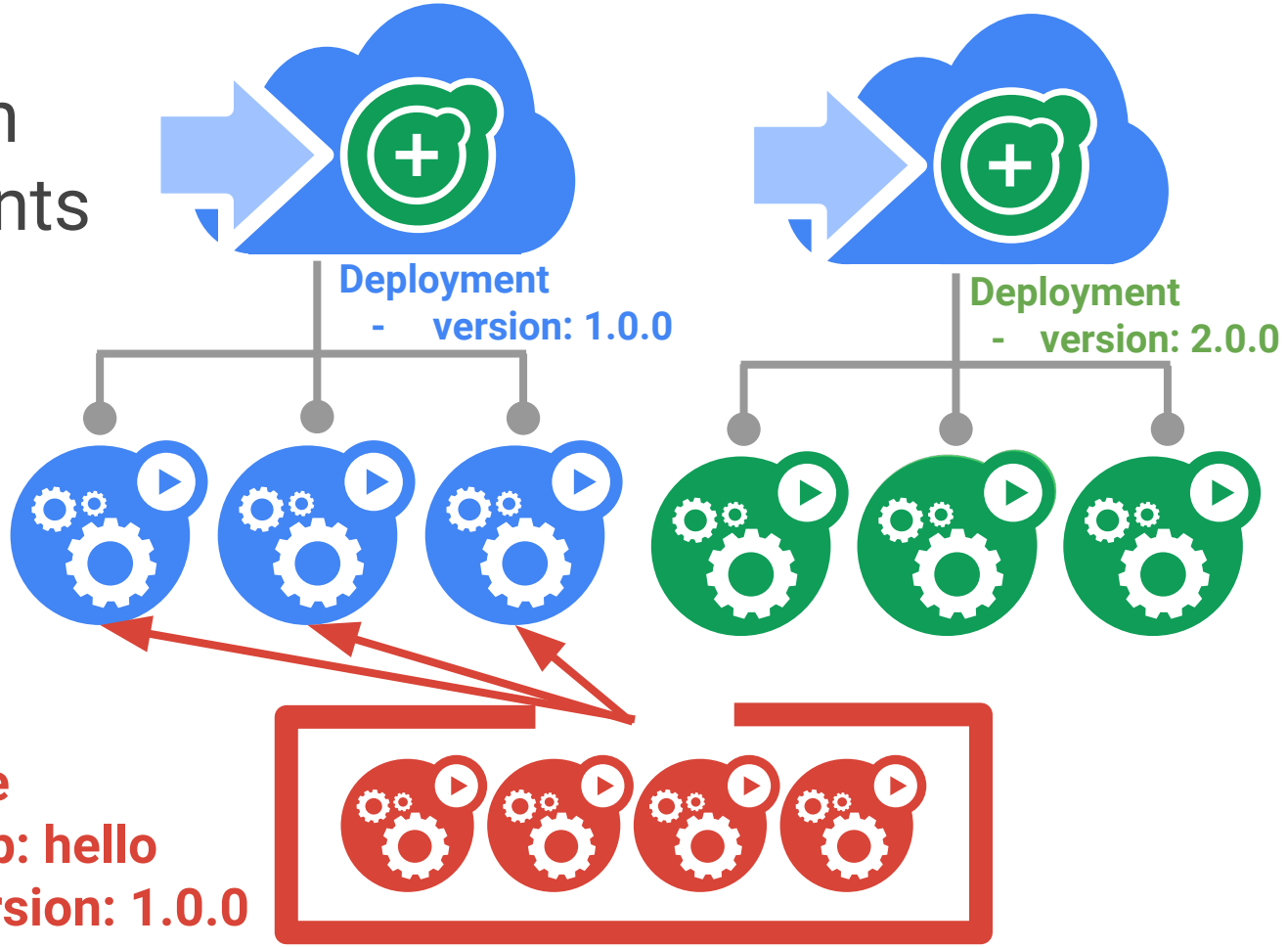
Blue-Green Deployments (1 of 4)



Blue-Green Deployments (2 of 4)



Blue-Green Deployments (3 of 4)



Blue-Green Deployments (4 of 4)