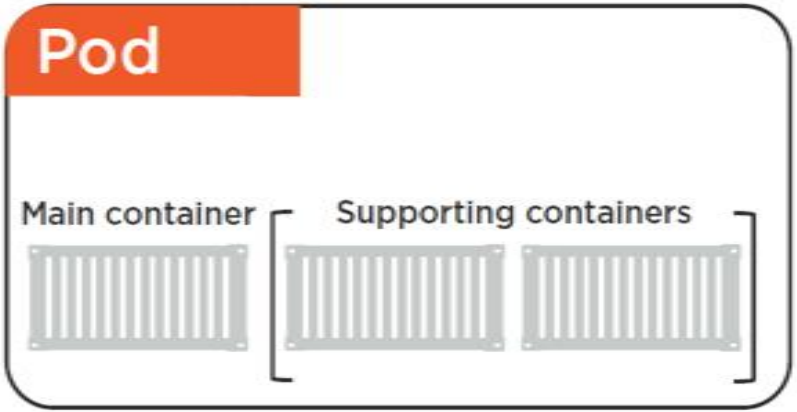
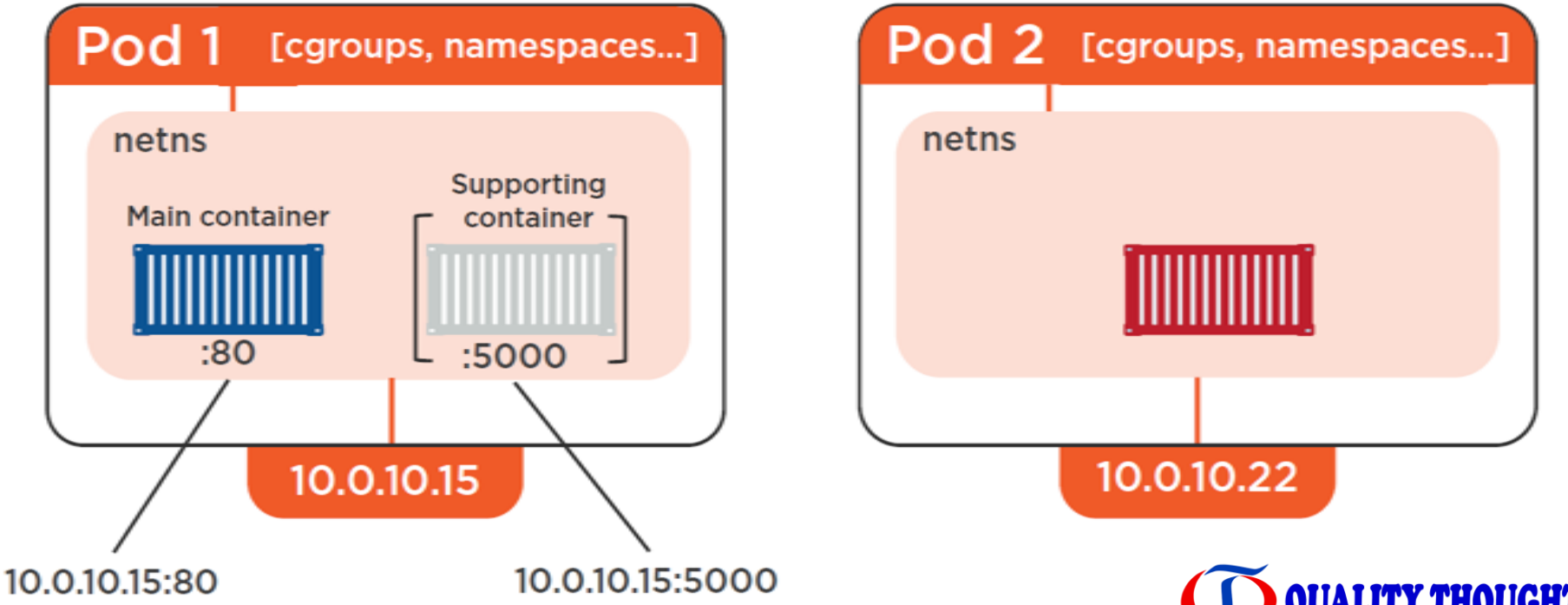
Theory

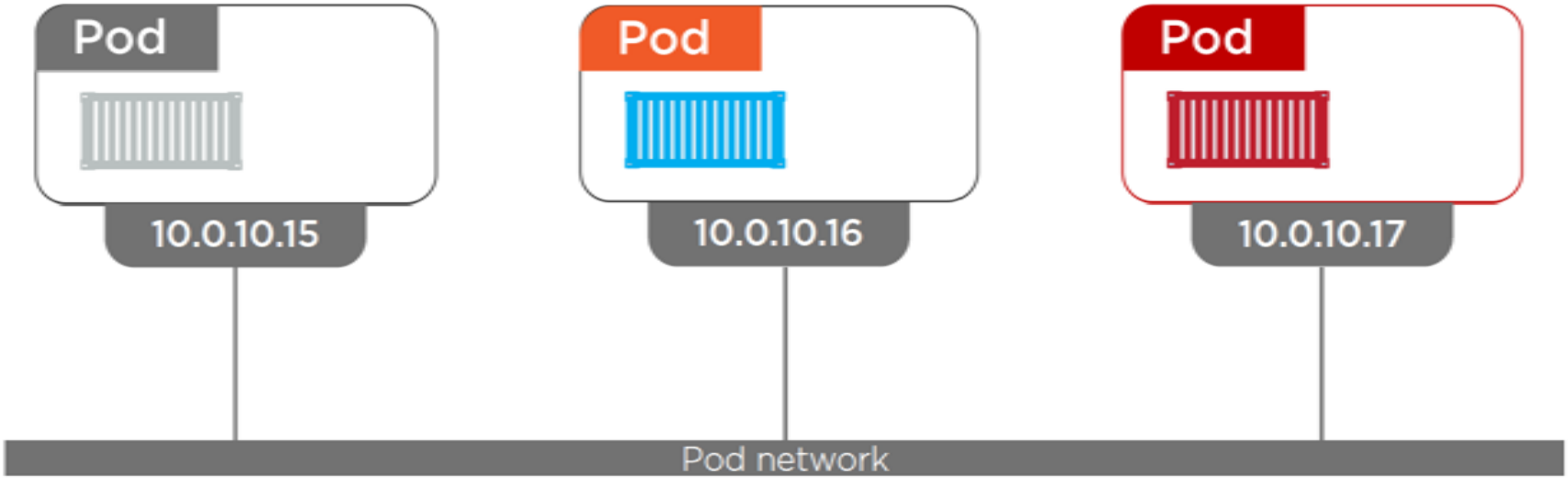



```
apiVersion: v1
kind: Pod
metadata:
  name: hello-pod
spec:
  containers:
  - name: hello-ctr
    image:…
    ports:
    - containerPort: 8080
```

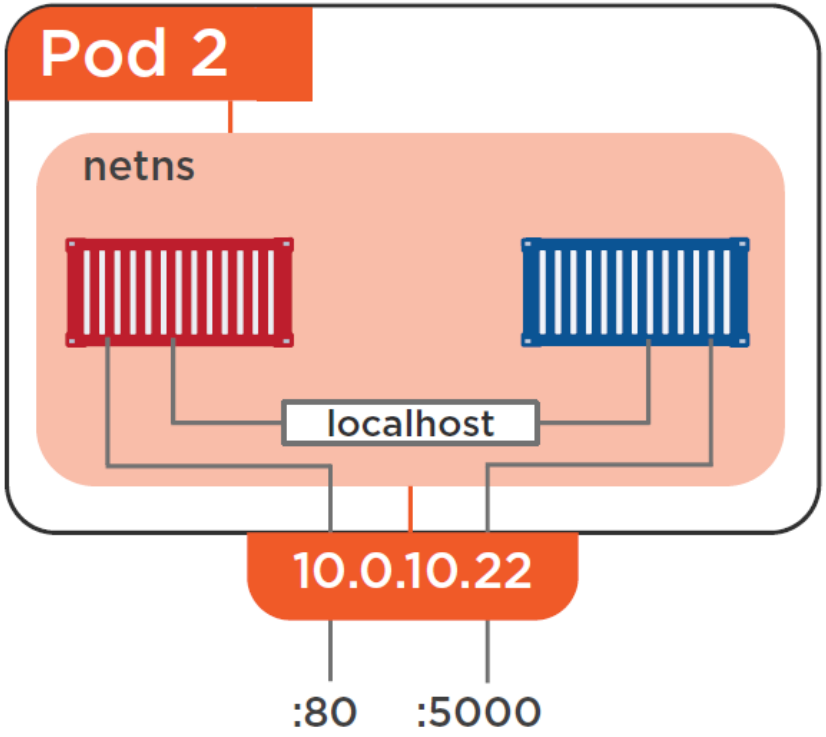
Theory



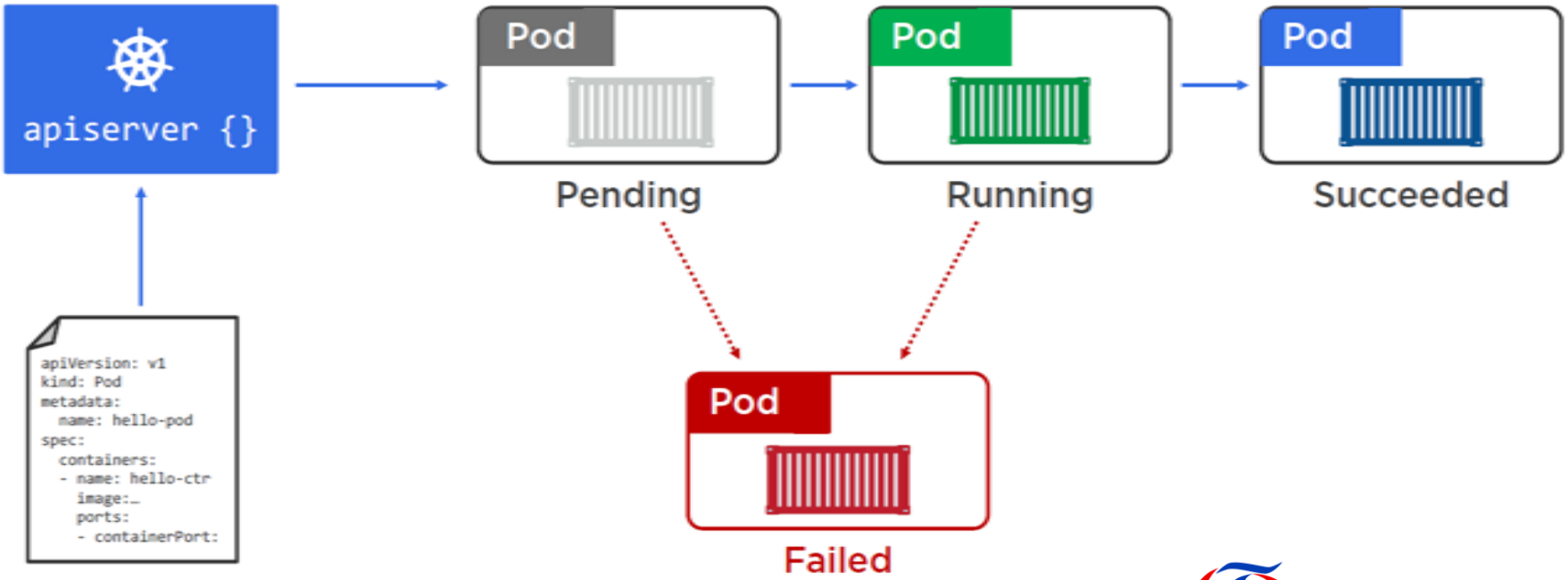
Inter-pod Communication



Intra-pod Communication



Pods:



Pods:

Smallest/atomic unit of scheduling

Have one or more containers

Scheduled on nodes (minions)

Declarative via manifest files

STEPS TO CREATE PODS

```
Create a Sample Pod i.e file with pod.yml
apiVersion: v1
kind: Pod
metadata:
  name: hello-pod
spec:
  containers:
  - name: hello-ctr
      image: jenkins
      ports:
      - containerPort: 8080
```

STEPS TO CREATE PODS

Execute the following commands

- kubectl get nodes
- kubectl create -f pod.yml
- kubectl get pods
- kubectl describe pods
- kubectl get pods -o wide
- kubectl get pods/hello-pod
- kubectl get pods --all-namespaces
- kubectl delete pods/hello-pod

STEPS TO CREATE REPLICATION CONTROLLER

Write a manifest with rc.yml (any name) with followint content

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: hello-rc
spec:
  replicas: 10
  selector:
          app: hello-world
  template:
          metadata:
          labels:
          app: hello-world
          spec:
          containers:
          - name: hello-ctr
          image: jenkins
          ports:
          - containerPort: 8080
```

STEPS TO CREATE REPLICATION CONTROLLER

Execute following commands

```
kubectl create -f rc.yml
kubectl get rc -o wide
kubectl describe rc
kubectl apply -f rc.yml
kubectl get rc
kubectl get pods
```

Refer: https://coreos.com/kubernetes/docs/latest/replication-controller.html

Summary

Smallest unit of scheduling

(Atomic)

Pod manifests

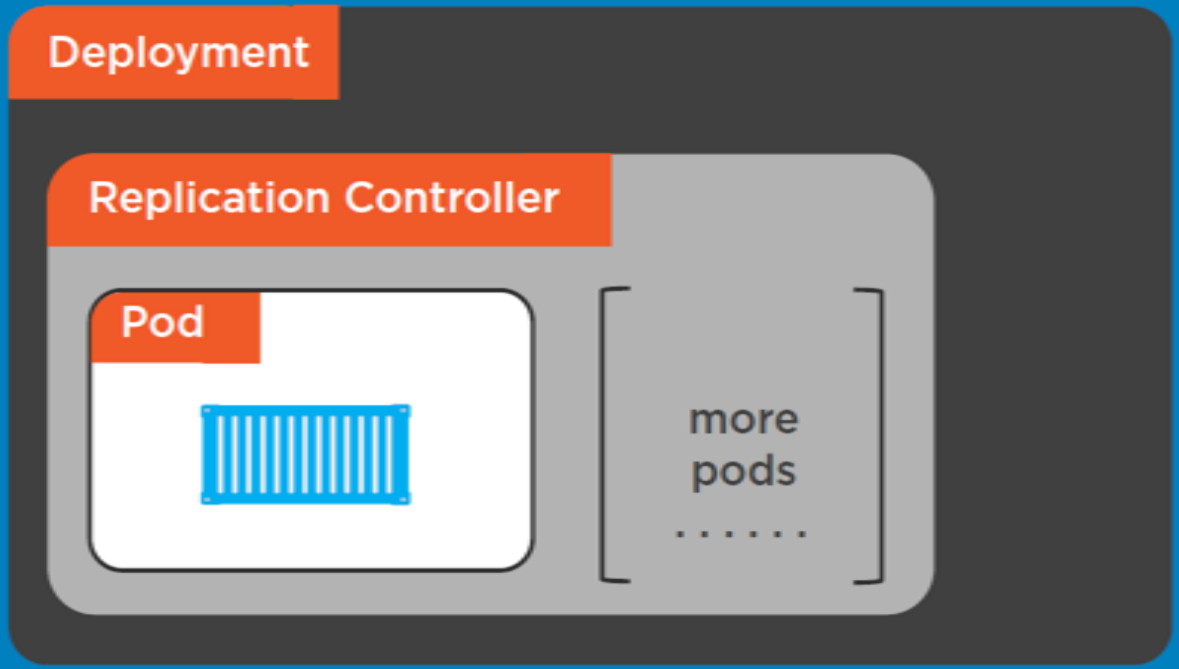
(YAML or JSON)

Deployed via other objects

(Replication Controllers etc...)

Replication Controllers

(Implement desired state)


Deployment
Replication Controller
Pod
more
pods
. . . . . .

Kubernetes Services

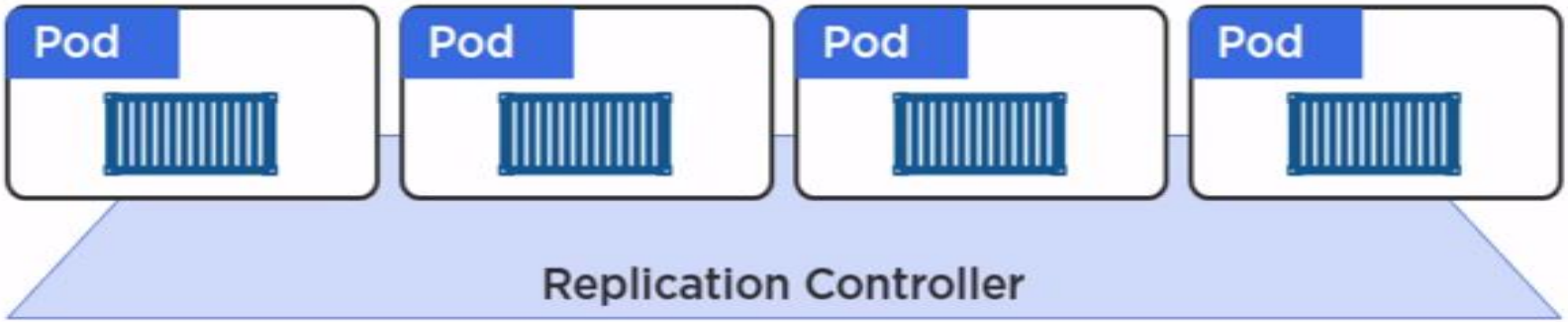
- The theory of Services
- Create a Service the iterative way
- Create a Service the declarative way
- Real-world application
- Recap

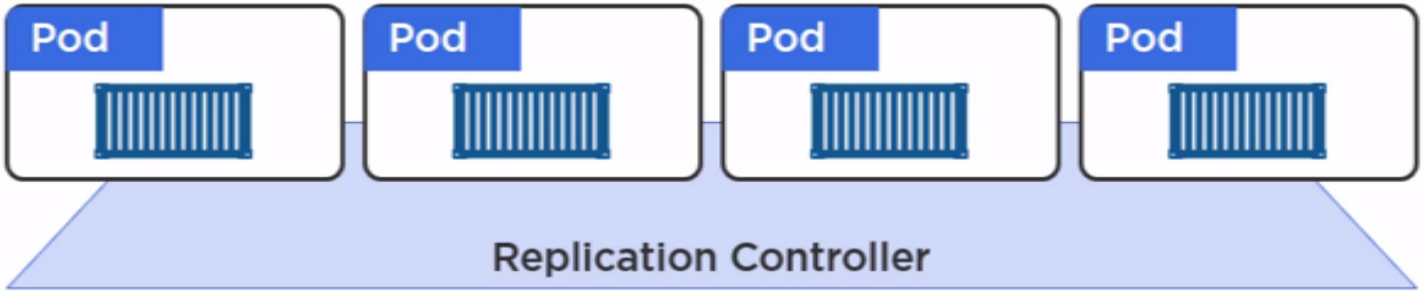
Kubernetes Services

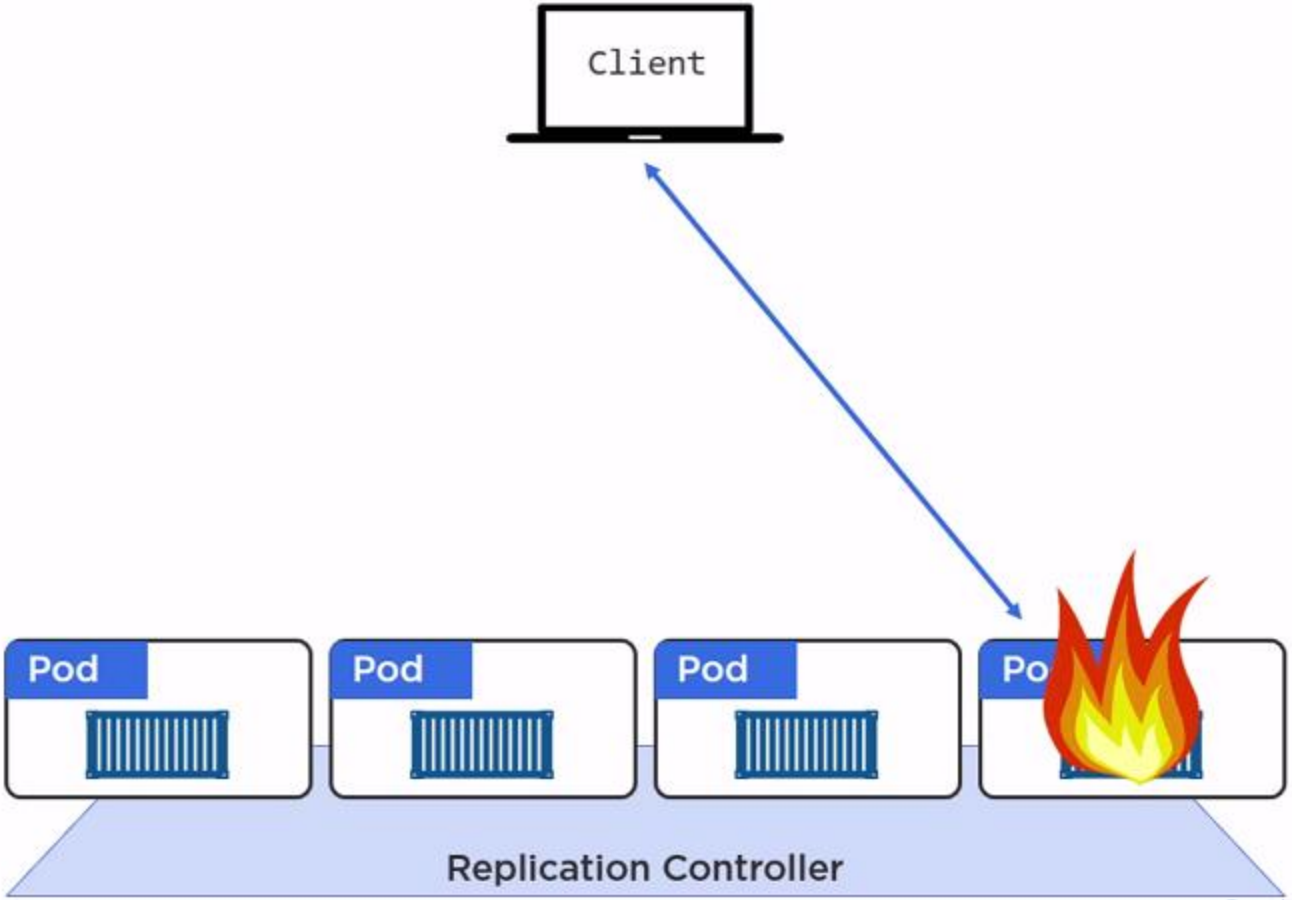
The Theory

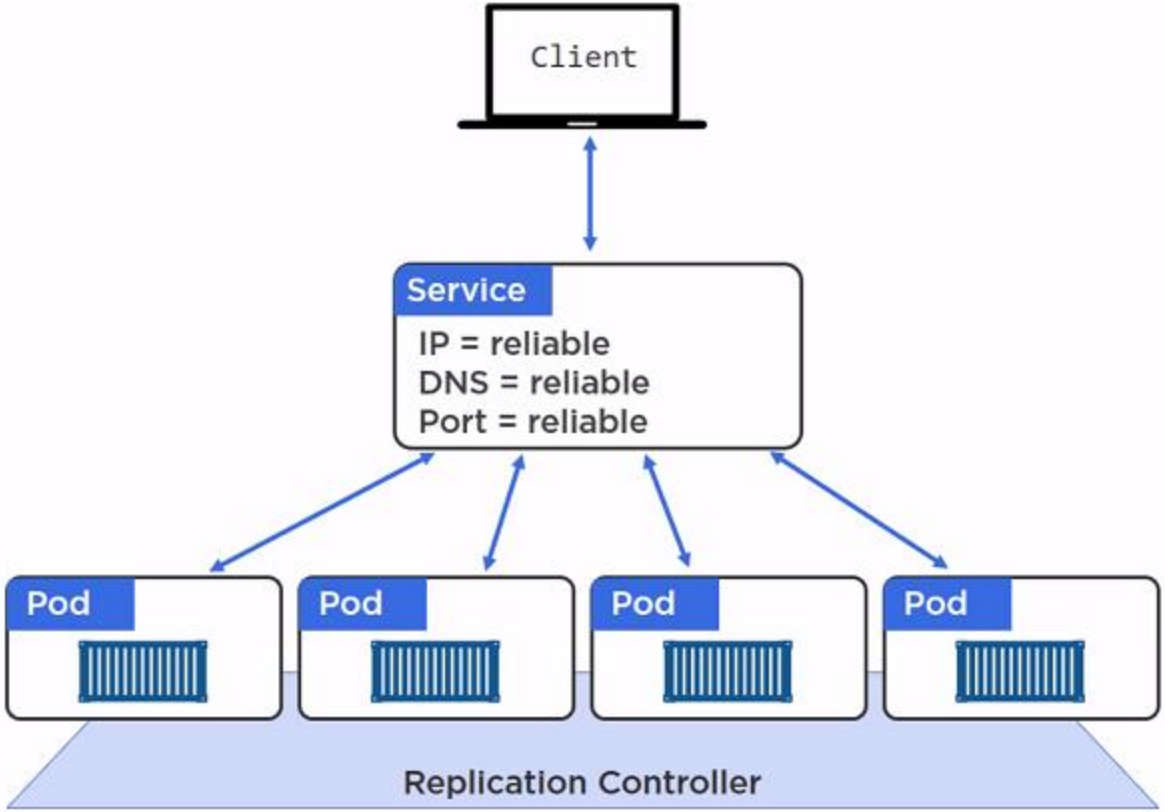
```
$ kubectl get pods
NAME              READY     STATUS    RESTARTS   AGE
hello-rc-1qch1    1/1       Running   0          8d
hello-rc-39q0s    1/1       Running   0          8d
hello-rc-3fr7t    1/1       Running   0          8d
hello-rc-5qzpl    1/1       Running   0          8d
…
```

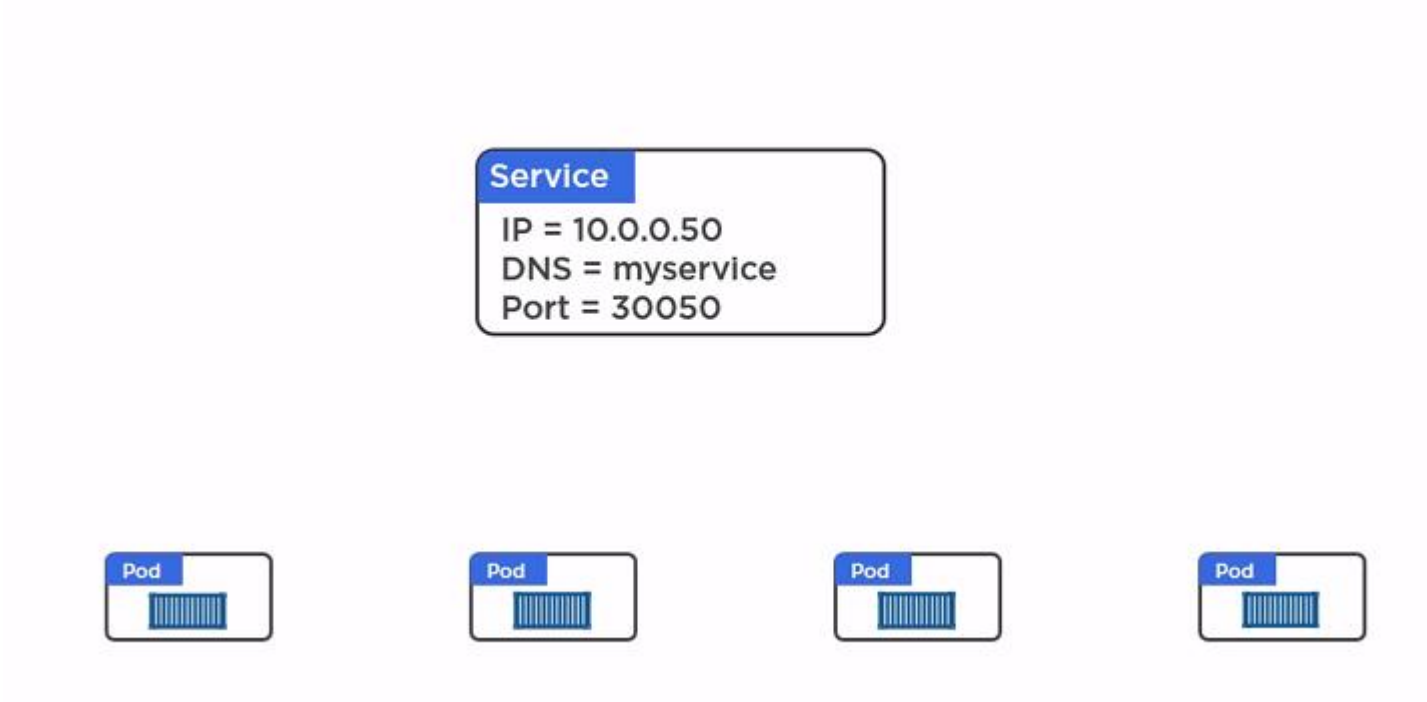
- How do we access our app?
  - From outside the cluster
  - From inside the cluster

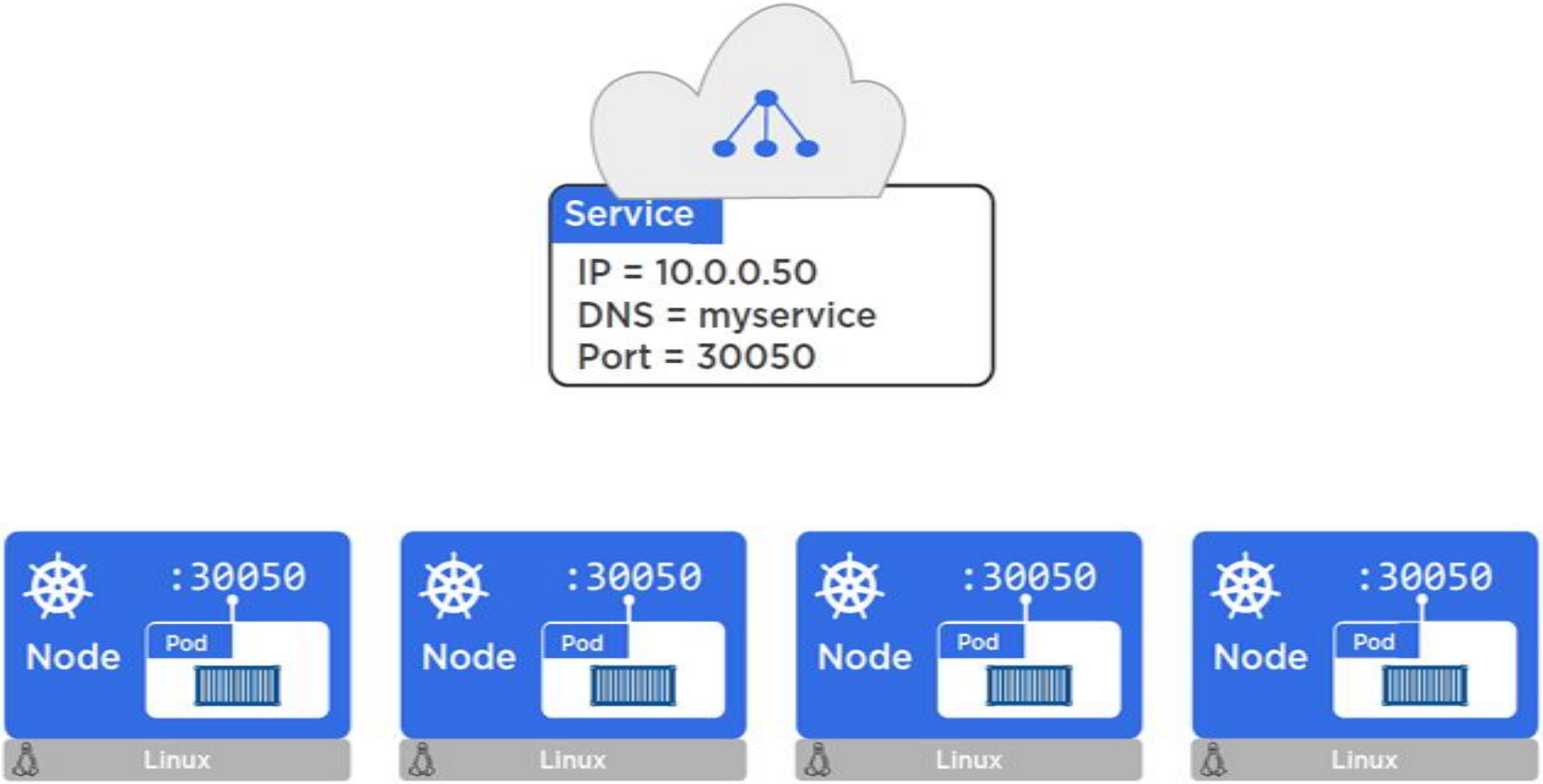


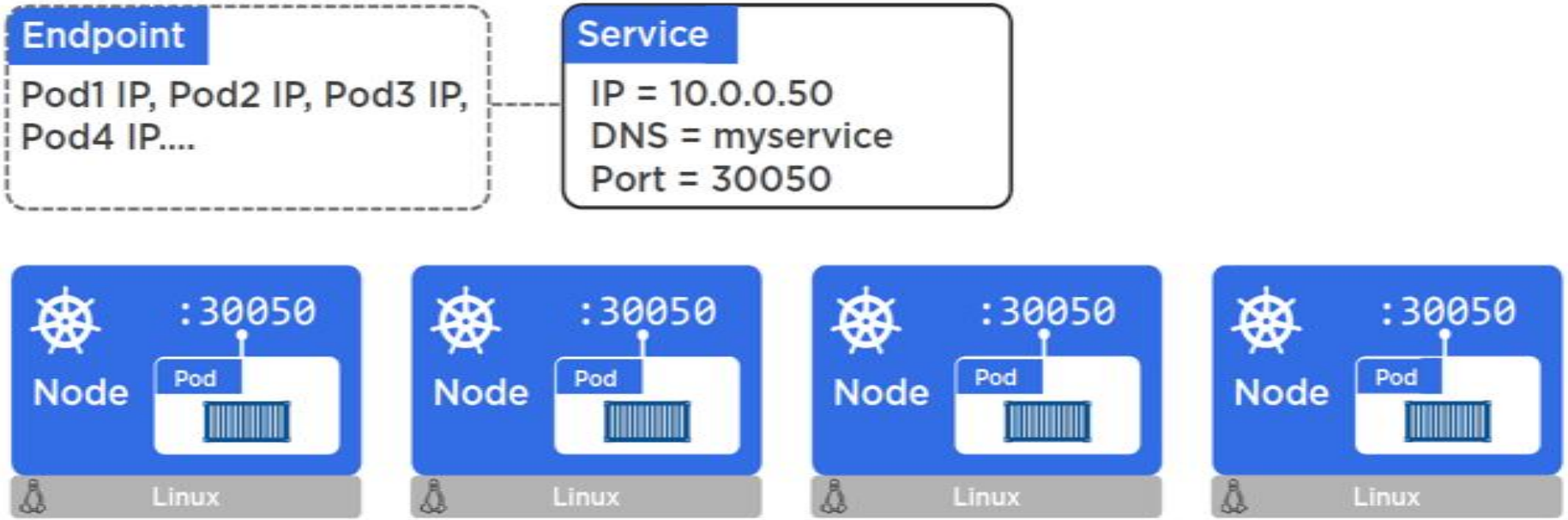


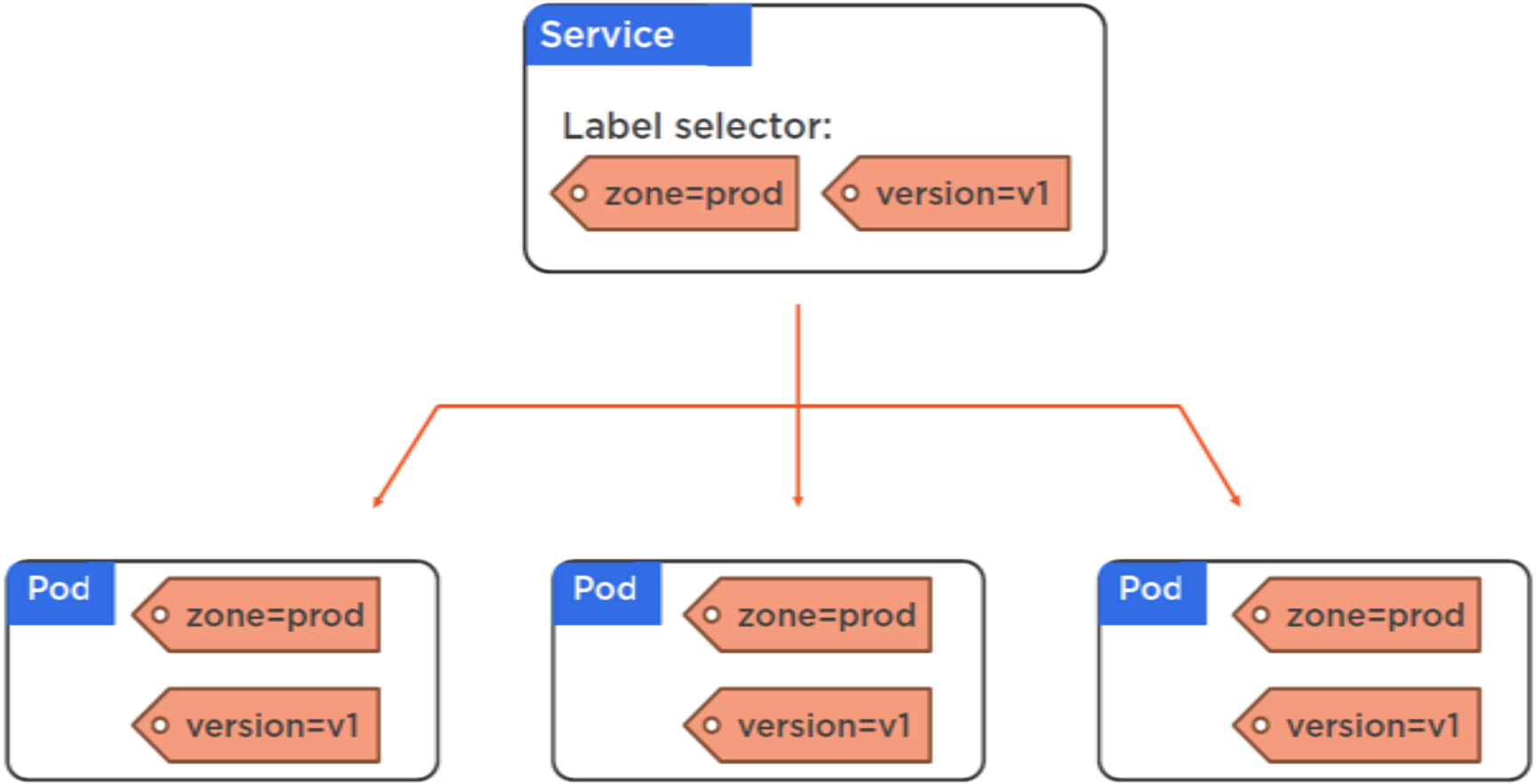







Service
IP = 10.0.0.50
DNS = myservice
Port = 30050
:30050
Node
Pod
Linux
:30050
Node
Pod
Linux
:30050
Node
Pod
Linux
:30050
Node
Pod
Linux





Service Discovery

- DNS based (best)
- Environment variables

ServiceType:

ClusterIP: Stable internal cluster IP

NodePort: Exposes the app outside of the cluster by adding a cluster-wide port on top of ClusterIP

LoadBalancer: Integrates NodePort with cloud-based load balancers

Creating SERVICE

Create the svc.yml file with following content

```
apiVersion: v1
kind: Service
metadata:
  name: hello-svc
  labels:
       app: nginx
spec:
  type: NodePort
  ports:
  - port: 80
       nodePort: 30001
       protocol: TCP
  selector:
       app: nginx
```

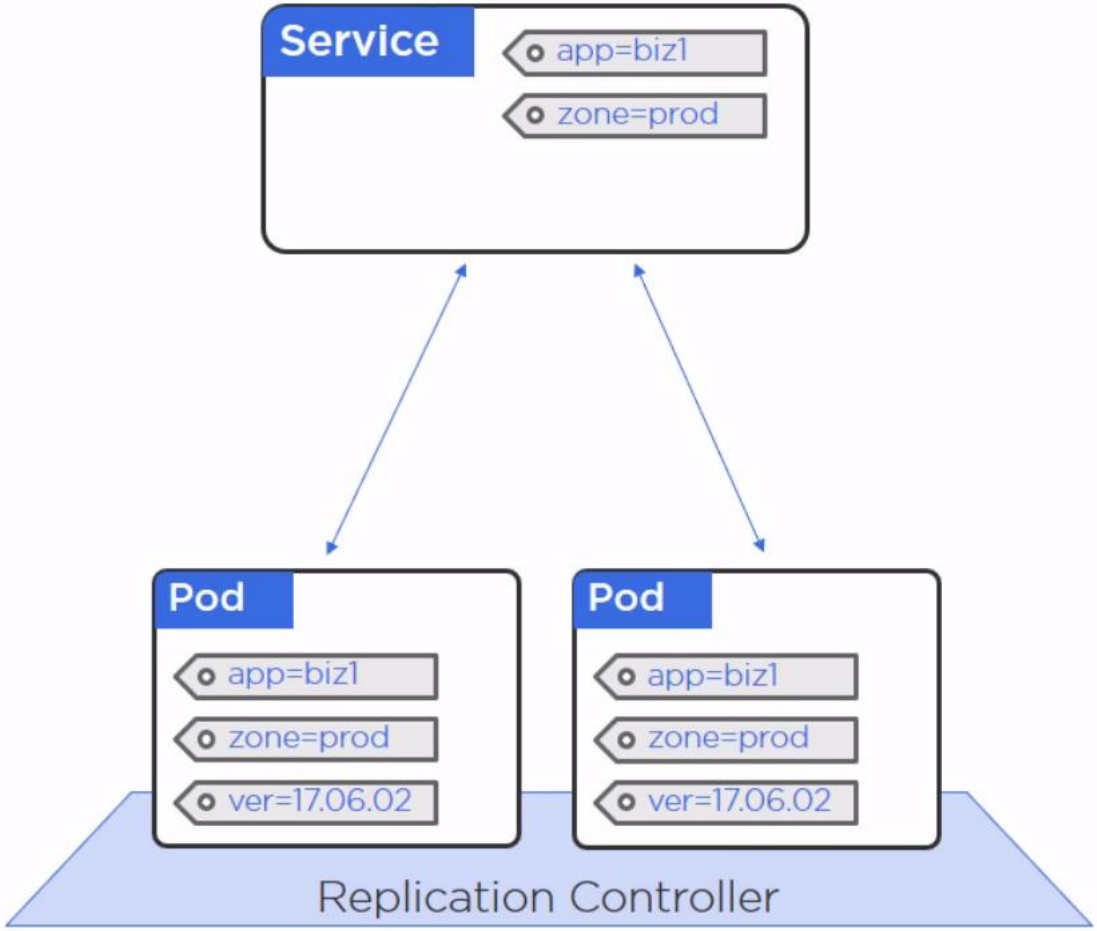
Creating SERVICE

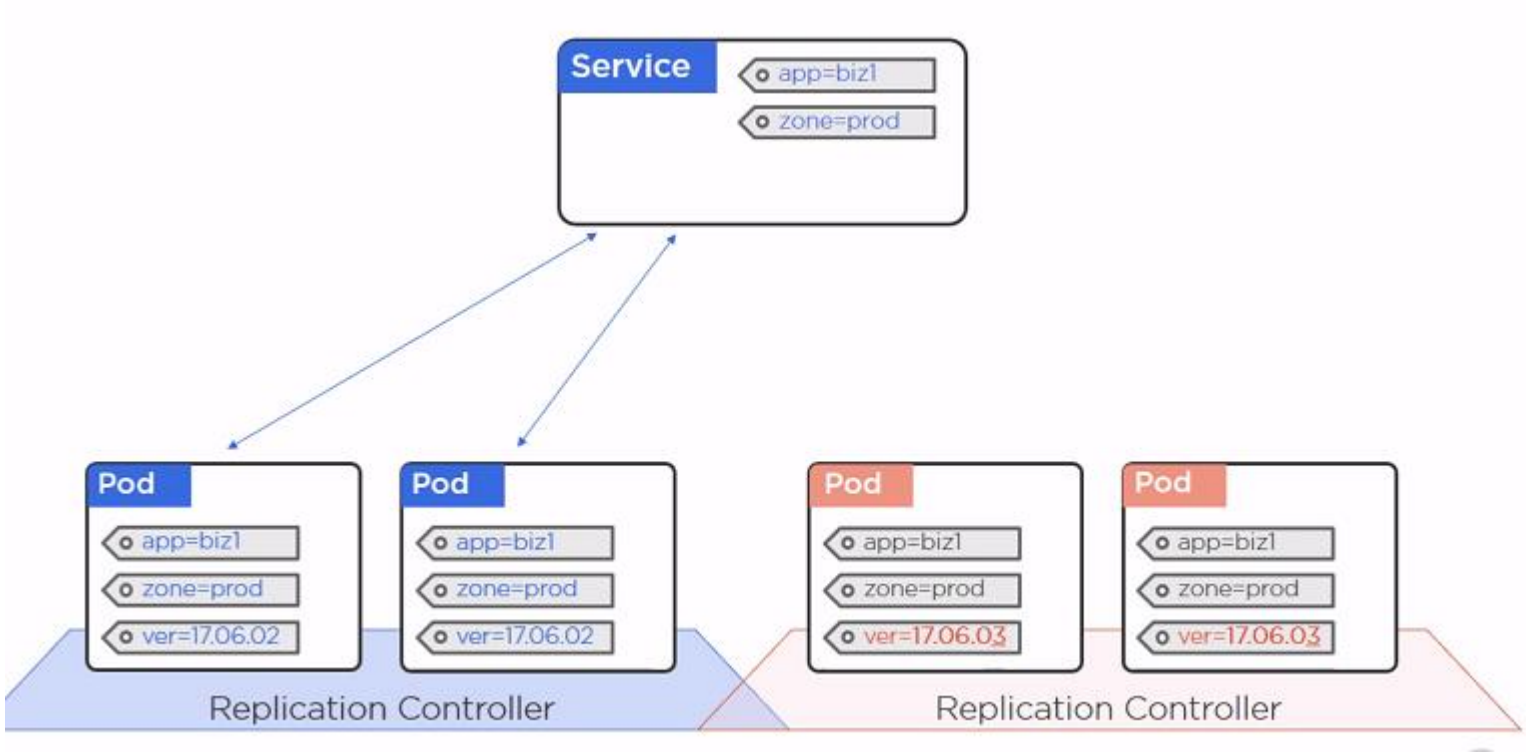
```
Ensure you give name of app “kubectl describe pods | grep app”
### The following command deploys a new Service from the "svc.yml". The YAML
    file is shown at the bottom of this document
  $ kubectl create -f svc.yml
### The following commands list all running Services and then describe the
    Service called "hello-svc"
$ kubectl get svc
$ kubectl describe svc hello-svc

### The following commands list all Endpoint objects on the cluster and then
    describes the Endpoint object called "hello-svc"
kubectl get ep
 kubectl describe ep hello-svc
```

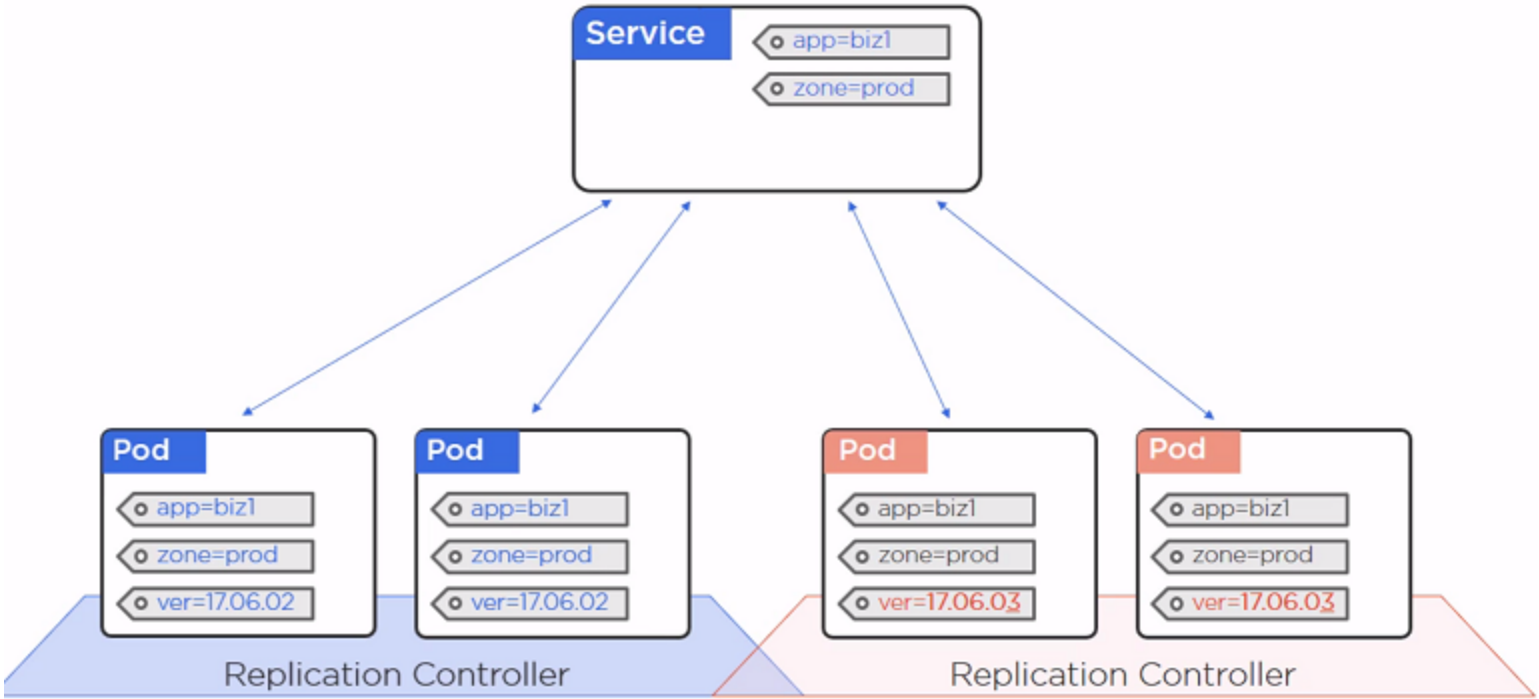
Kubernetes Services

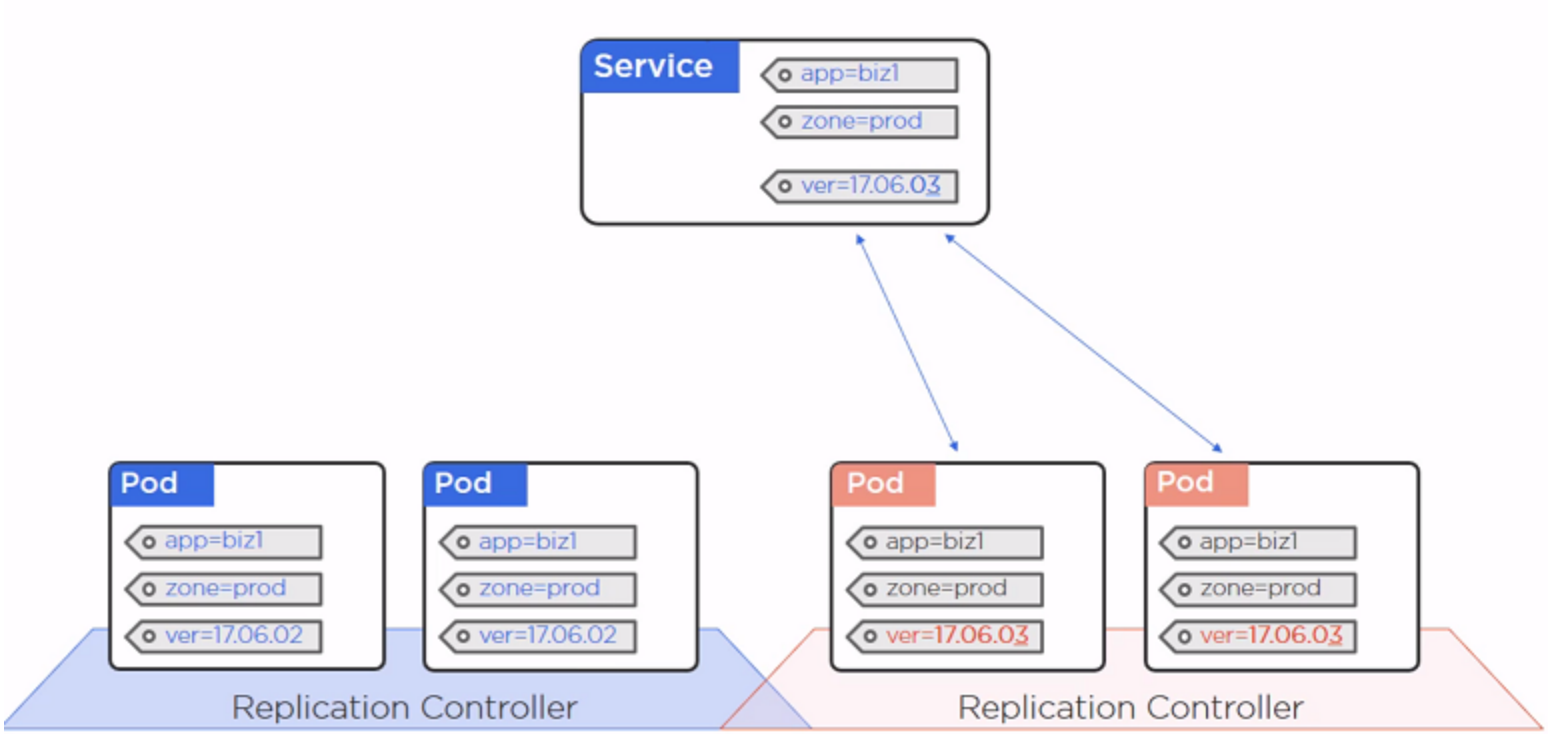
In the real world

Updating Business Apps









Kubernetes Services Summary

Reliable network endpoint

IP address

DNS name

Port

Expose Pods to the outside world

NodePort
Provides a cluster-wide port

LoadBalancer
Integrates with cloud-based load balancers

Kubernetes Deployments

- The theory of Deployments
- Create a new Deployment
- Update a Deployment
  - Rolling update and a rollback
- Recap

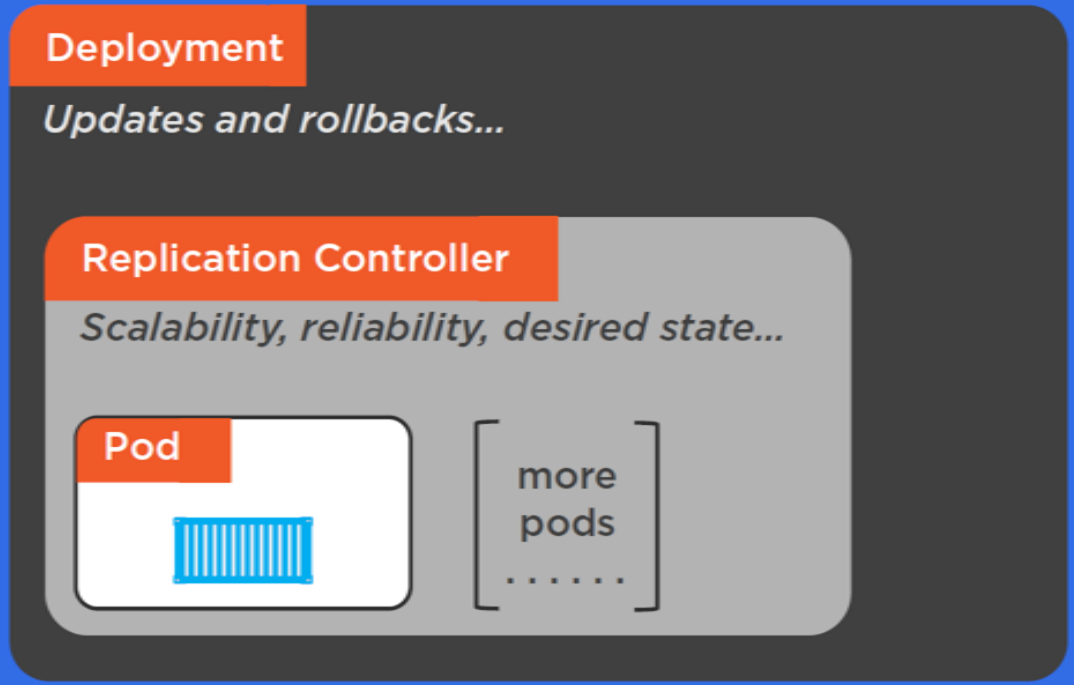
Updates & Rollbacks

Kubernetes Deployments

The Theory

Kubernetes Deployments
The Theory

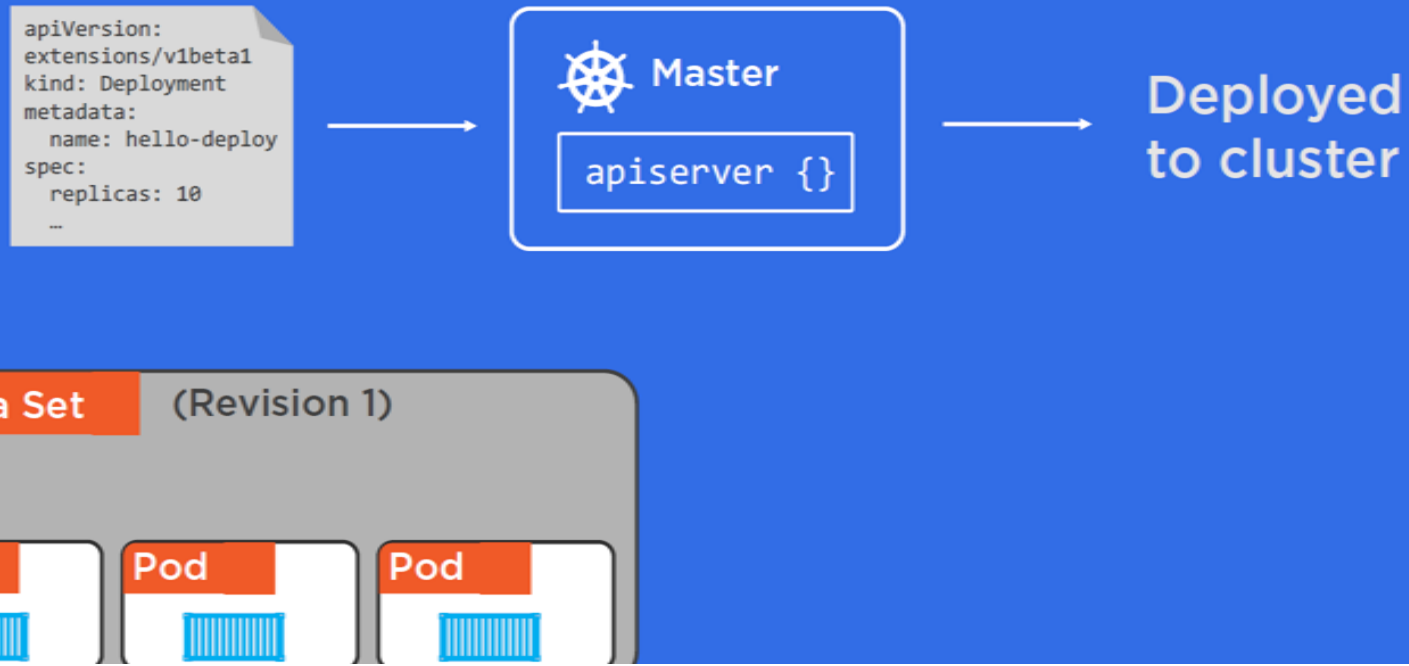
Updates & Rollbacks



Kubernetes Deployments

The Theory

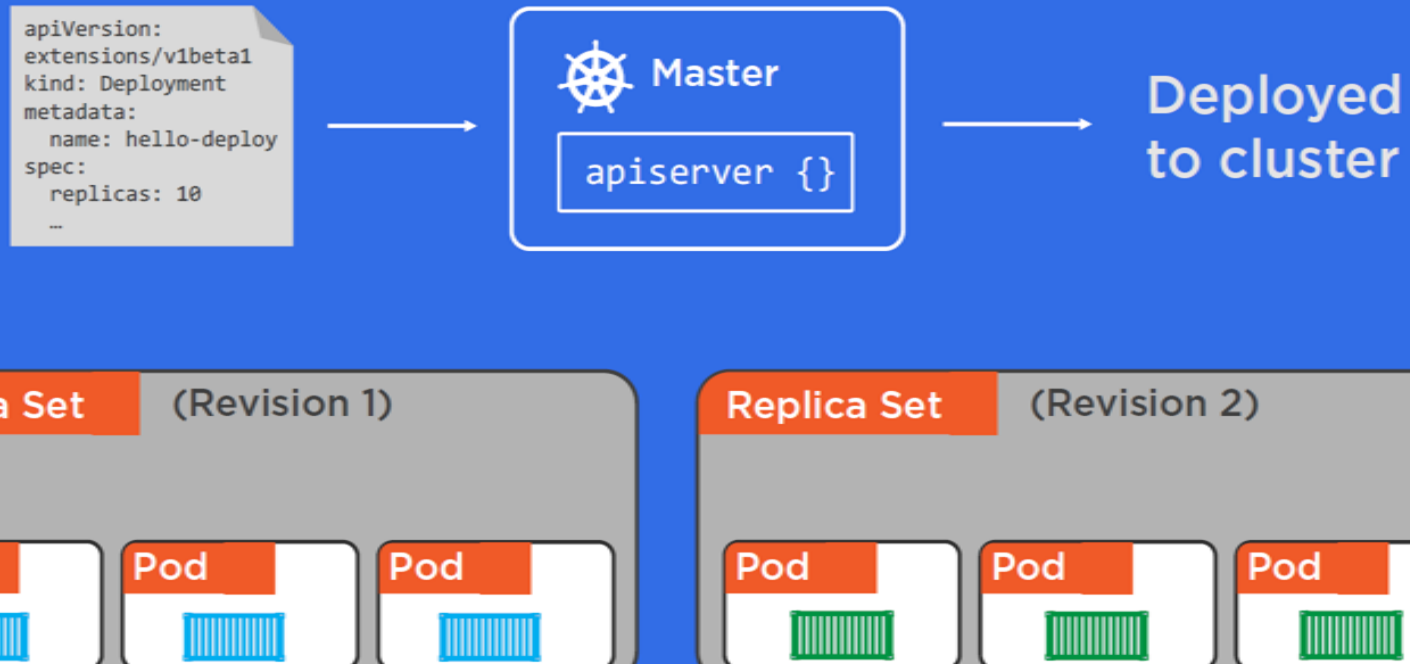
Updates & Rollbacks



Kubernetes Deployments

The Theory

Updates & Rollbacks



CREATING DEPLOYMENT

Create deploy.yml with following content

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: jenkins-deploy
spec:
  replicas: 2
  minReadySeconds: 2
  strategy:
          type: RollingUpdate
          rollingUpdate:
          maxUnavailable: 1
          maxSurge: 1
  template:
          metadata:
          labels:
          app: jenkins
          spec:
          containers:
          - name: jenkins-pod
          image: jenkins
          ports:
          - containerPort: 8080
```

CREATING DEPLOYMENT

```
kubectl create -f deploy.yml
kubectl describe deploy jenkins-deploy
kubectl get rs
kubectl describe rs
```

ROLLING UPDATE TO THE DEPLOYMENT

```
kubectl apply -f deploy.yml --record

kubectl rollout status deployments jenkins-deploy

kubectl get deploy jenkins-deploy

kubectl rollout history deployments jenkins-deploy

kubectl get rs
```

UNDO ROLLED UPDATE

```
kubectl describe deploy jenkins-deploy

kubectl rollout undo deployment jenkins-deploy --to-
revision=1

kubectl get deploy

kubectl rollout status deployments jenkins-deploy
```

Kubernetes Deployments Summary

Deployments are the future!

Updates & Rollbacks

`$ kubectl run …`
(iterative)

Be Declarative!

`$ kubectl create -f <manifest>`

↓

Check in to source control

↓

Make changes to **same** file

↓

Apply change with
`$ kubectl apply …`