Kubernetes Architecture

Nigel Poulton

@nigelpoulton www.nigelpoulton.com

Module Outline

Big picture view

Masters

Nodes

Pods

Services

Deployments

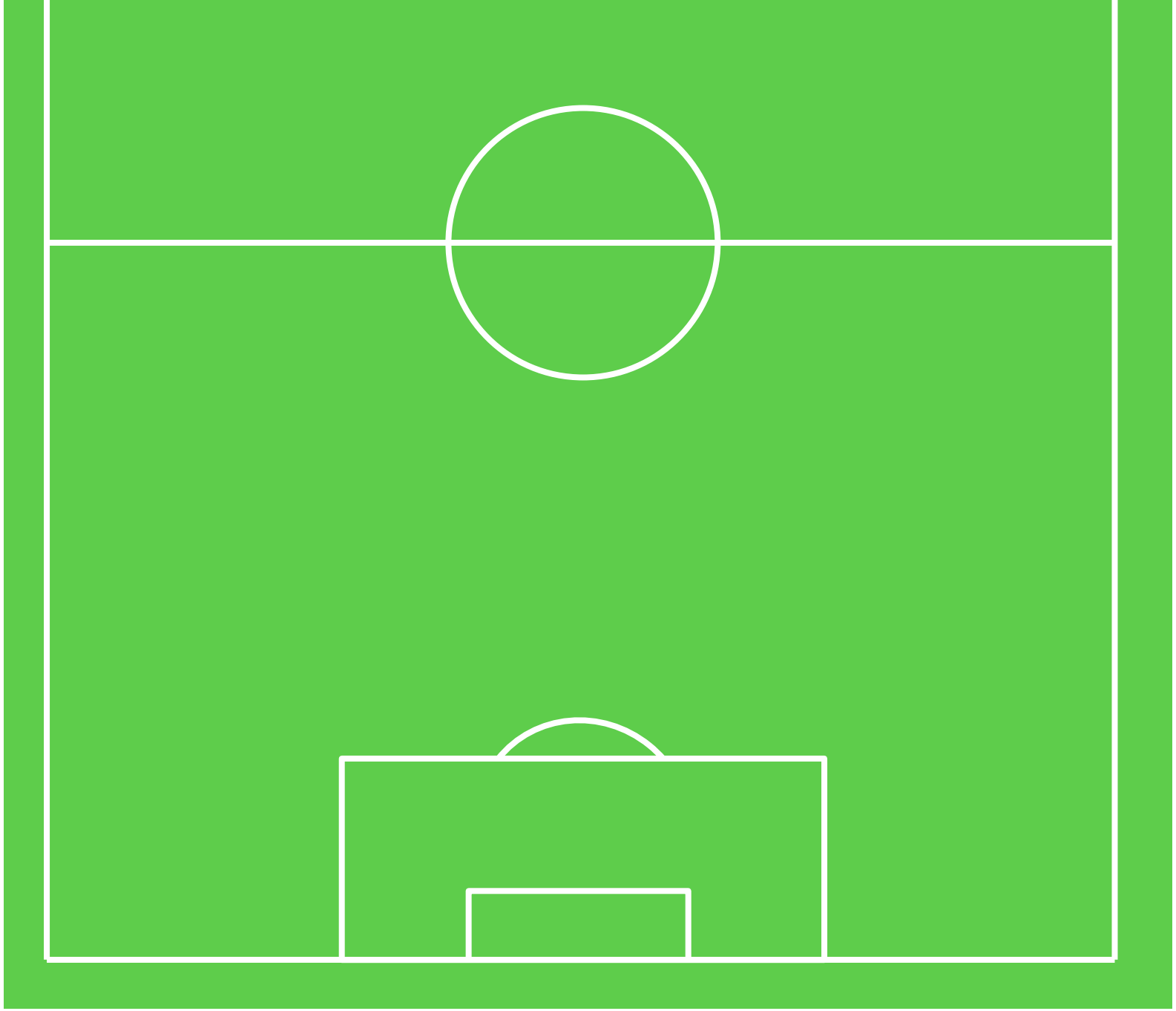
Recap

Kubernetes

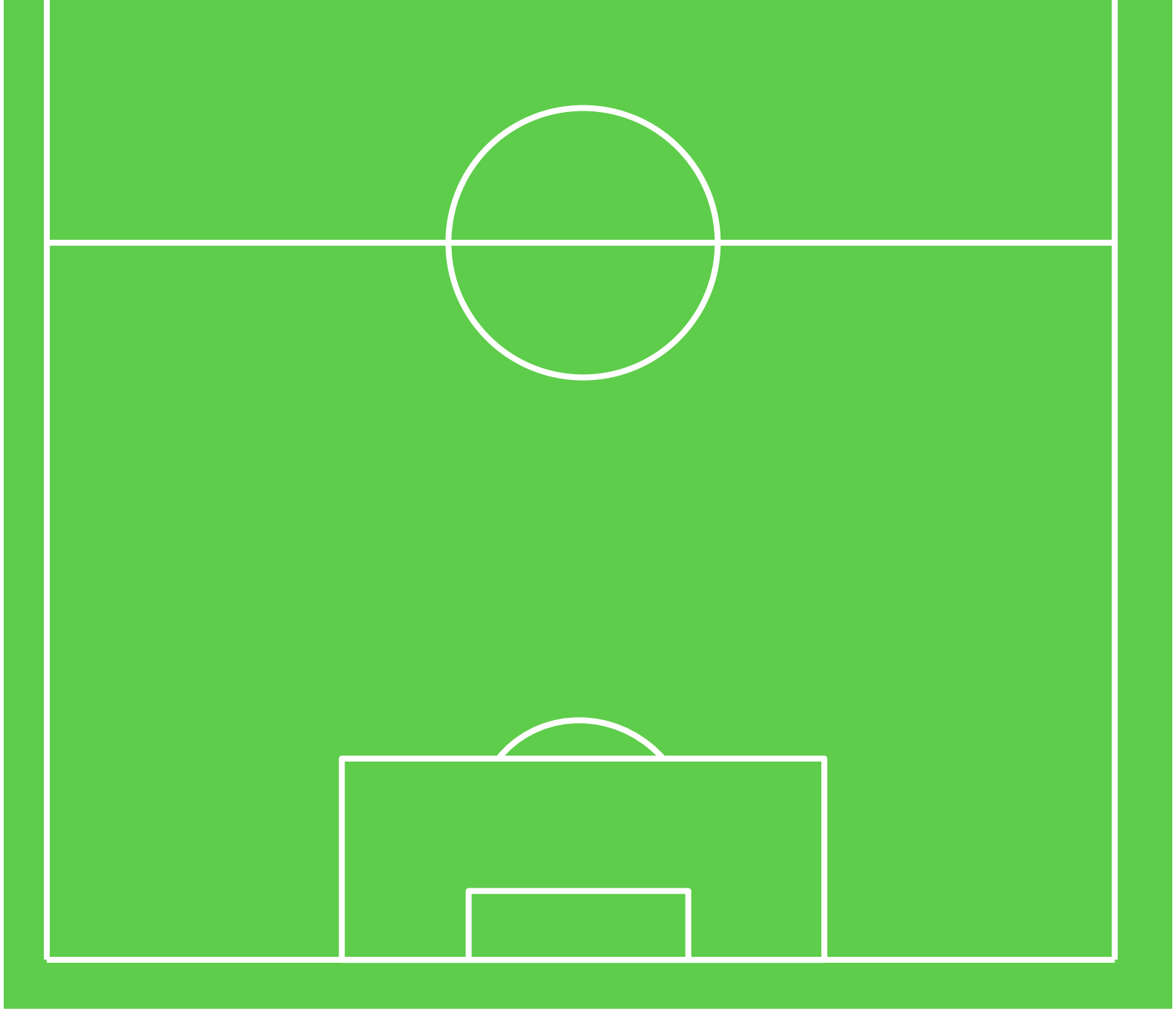
Big Picture View

Team

Manager
(coach)



Team



Team

Manager
(coach)

S1

S2

S3

HTTPS

HTTPS

Search

Auth

K/V store

MySQL

Log

Load balancer

HTTPS

HTTPS

Search

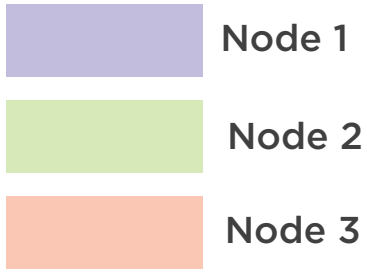
Auth

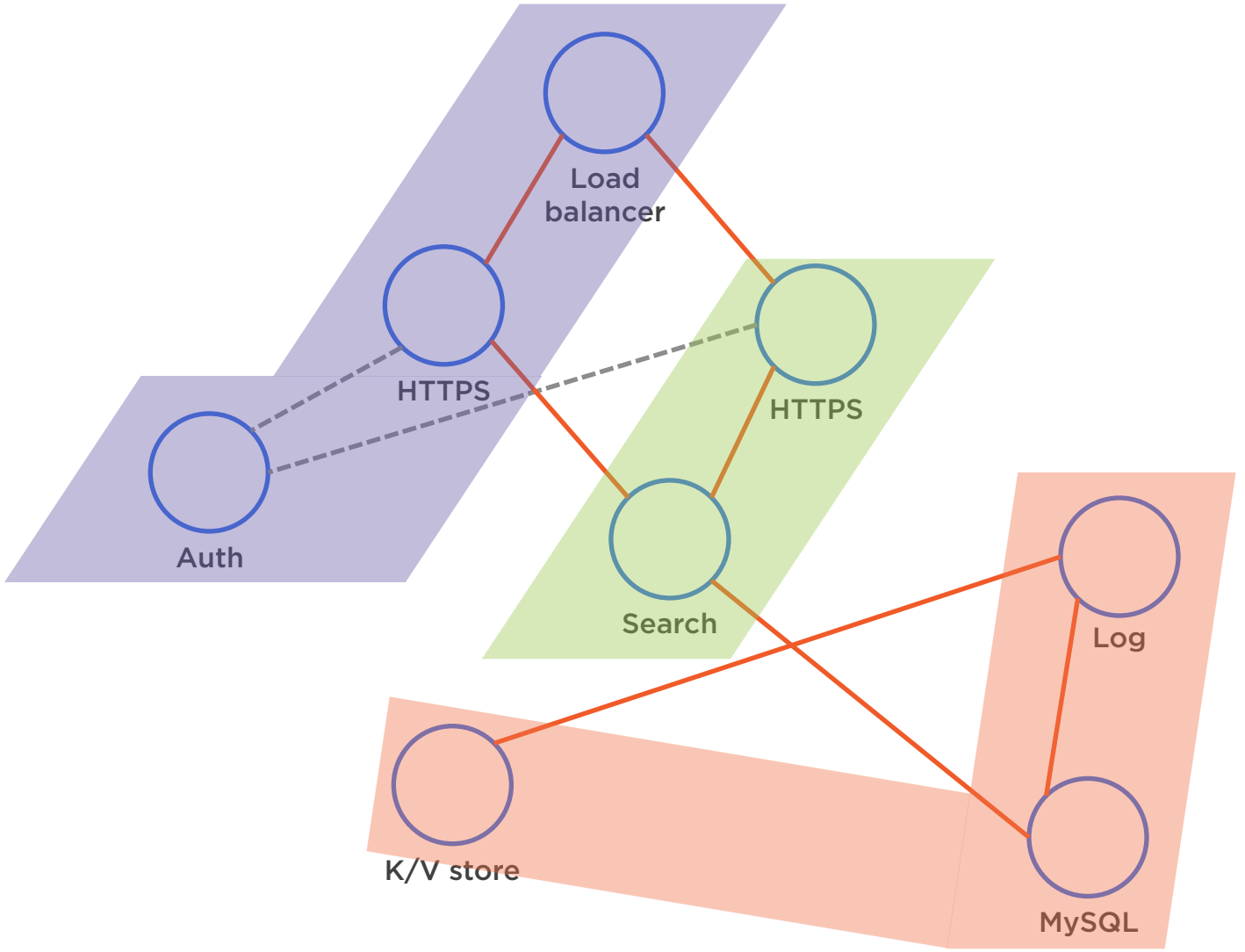
K/V store

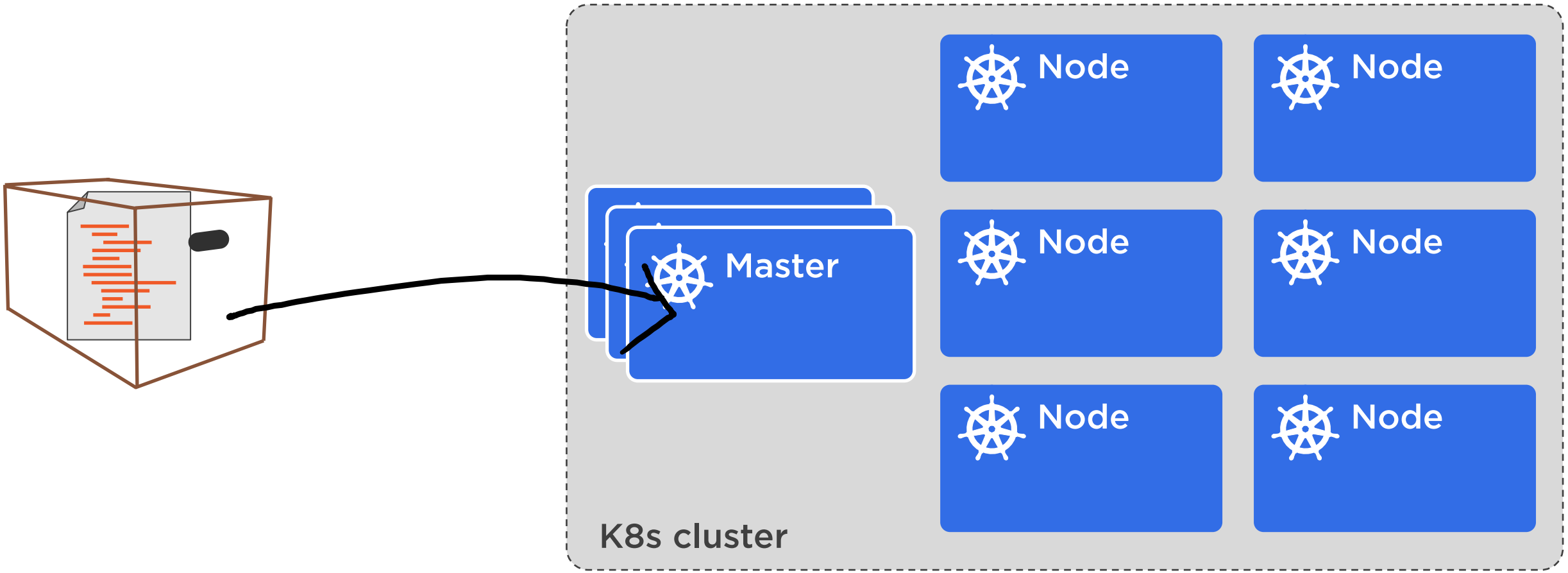
MySQL

Log

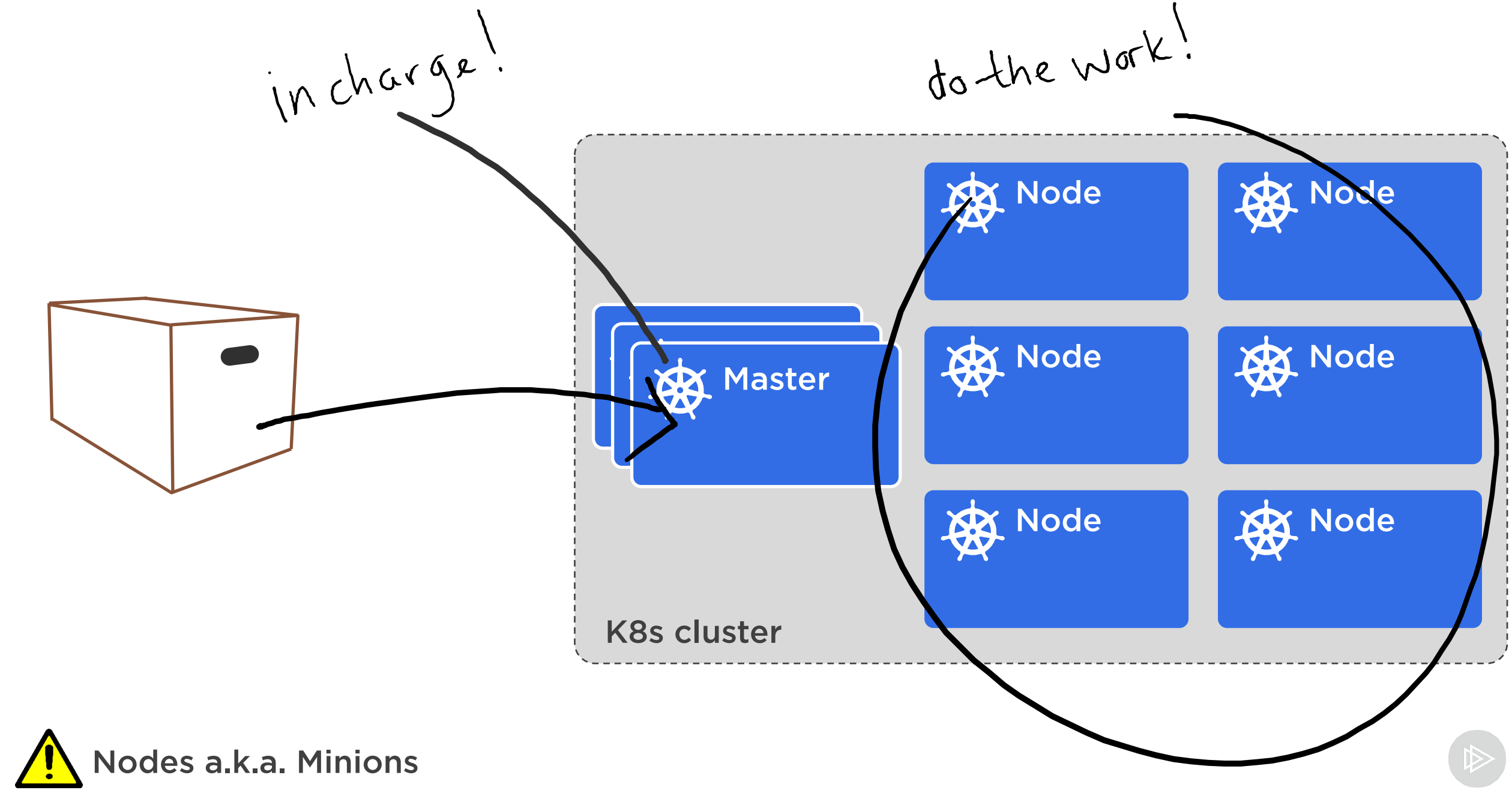
Load balancer


Node 1
Node 2
Node 3

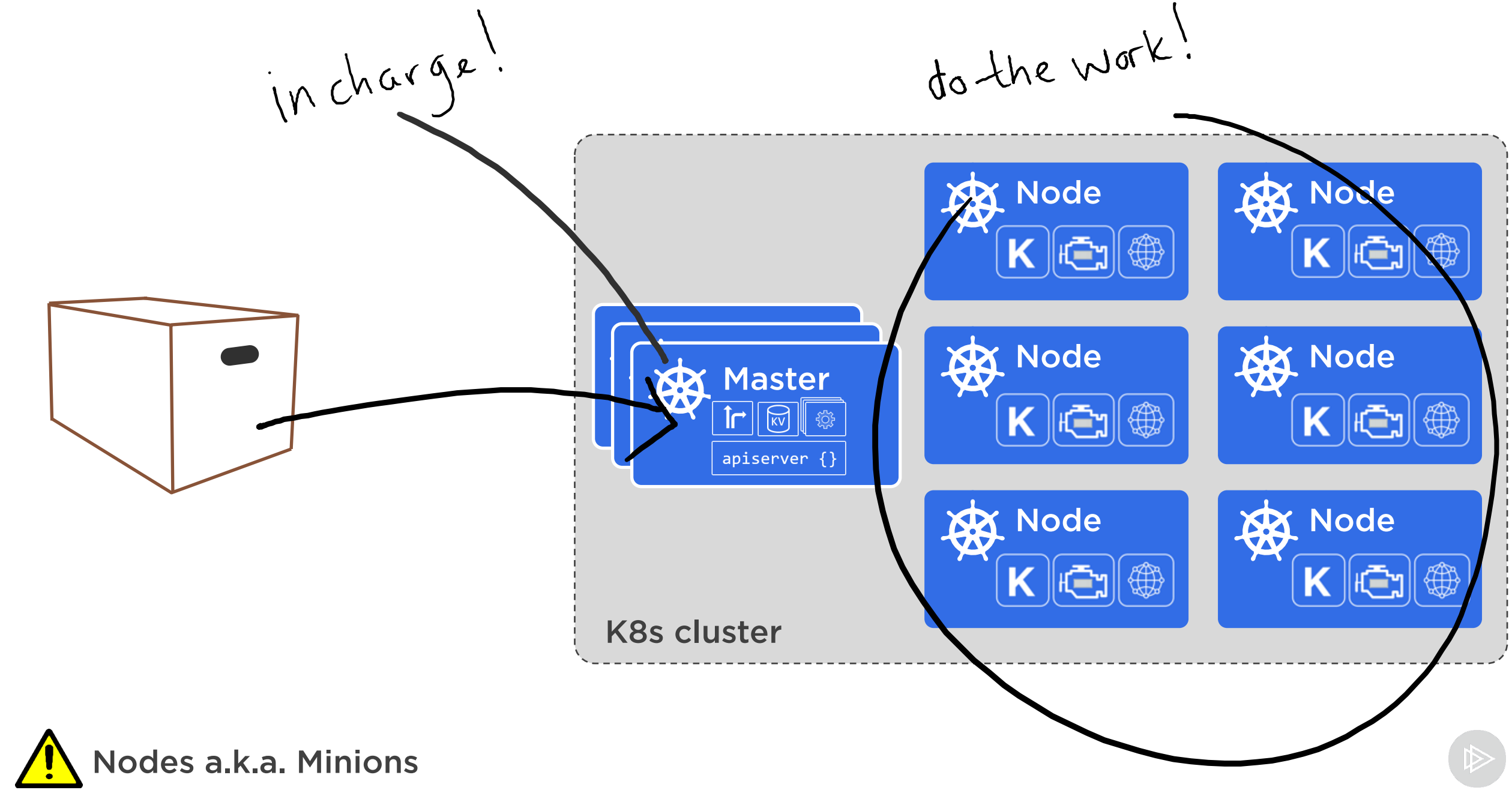

Load
balancer
HTTPS
HTTPS
Auth
Search
Log
K/V store
MySQL



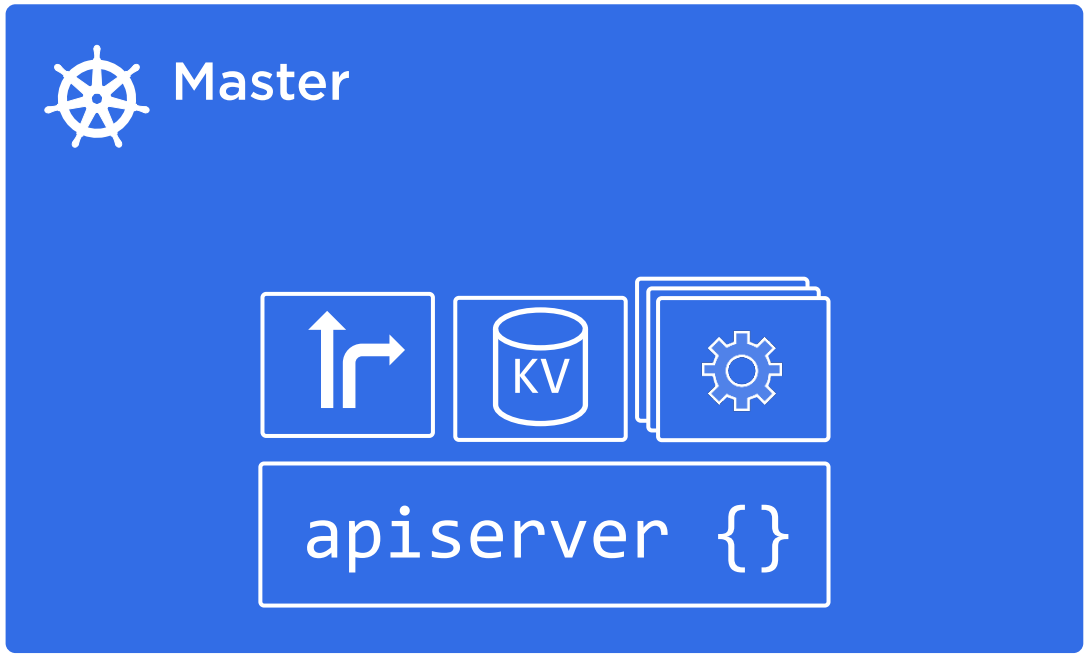
Nodes a.k.a. Minions

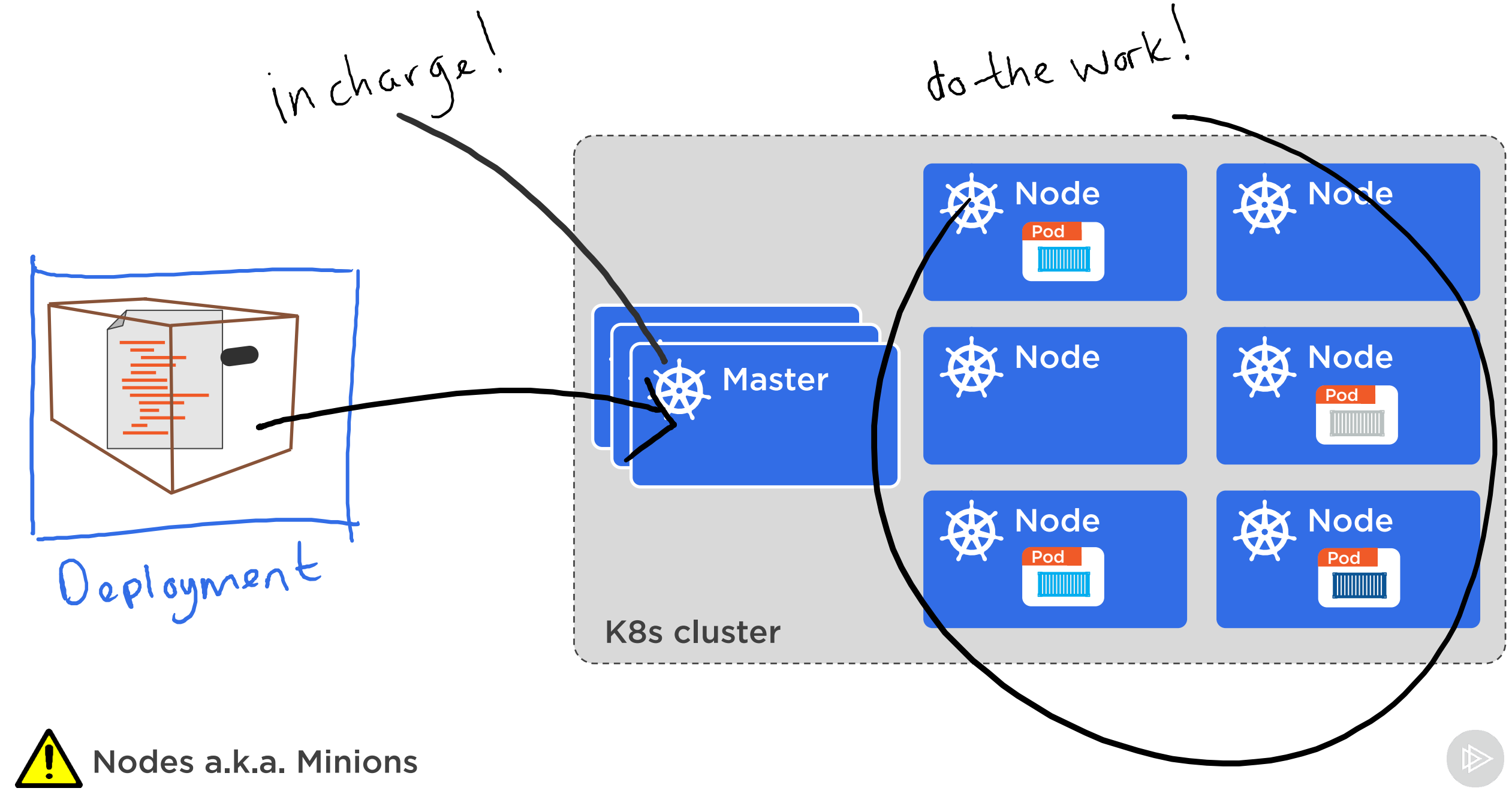


Nodes a.k.a. Minions



Nodes a.k.a. Minions


Master
KV
apiserver {}



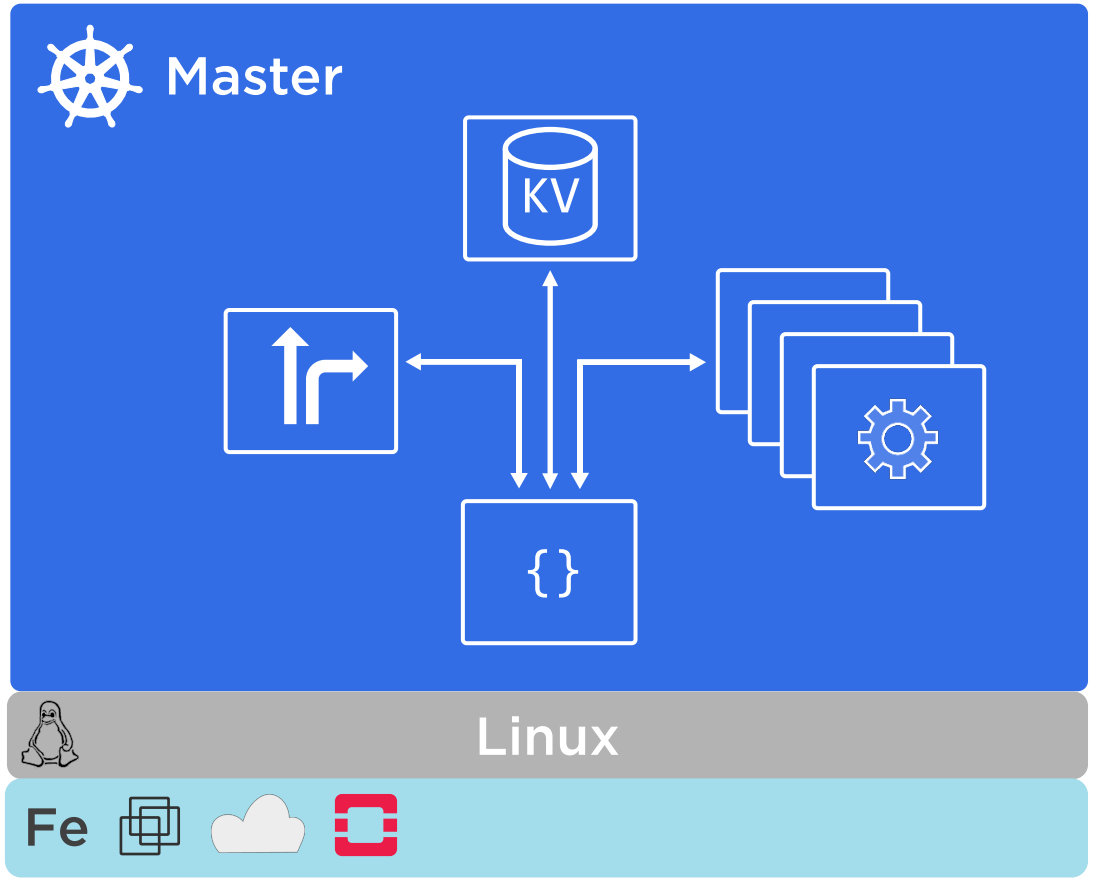
Nodes a.k.a. Minions

Masters

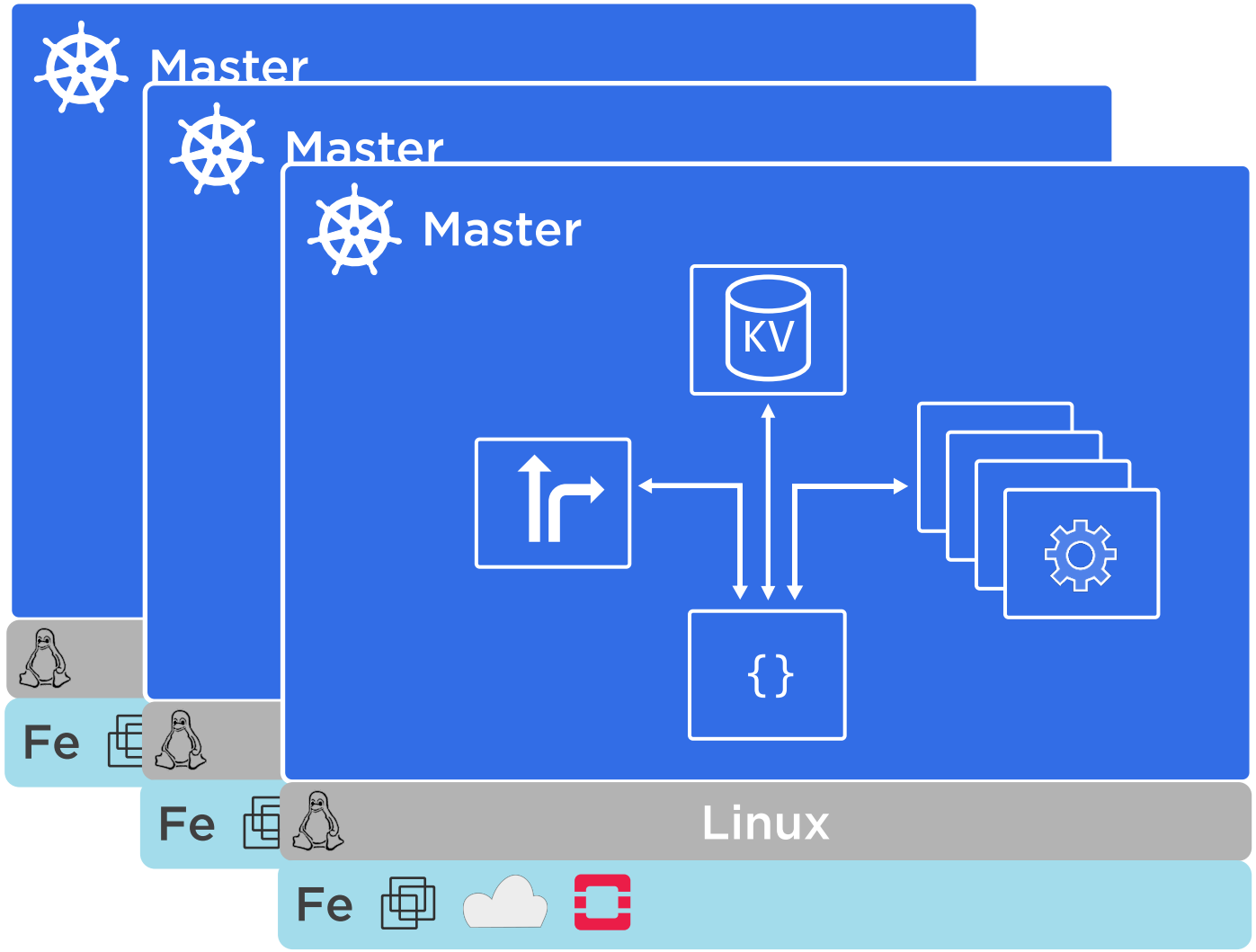
The Kubernetes Control Plane

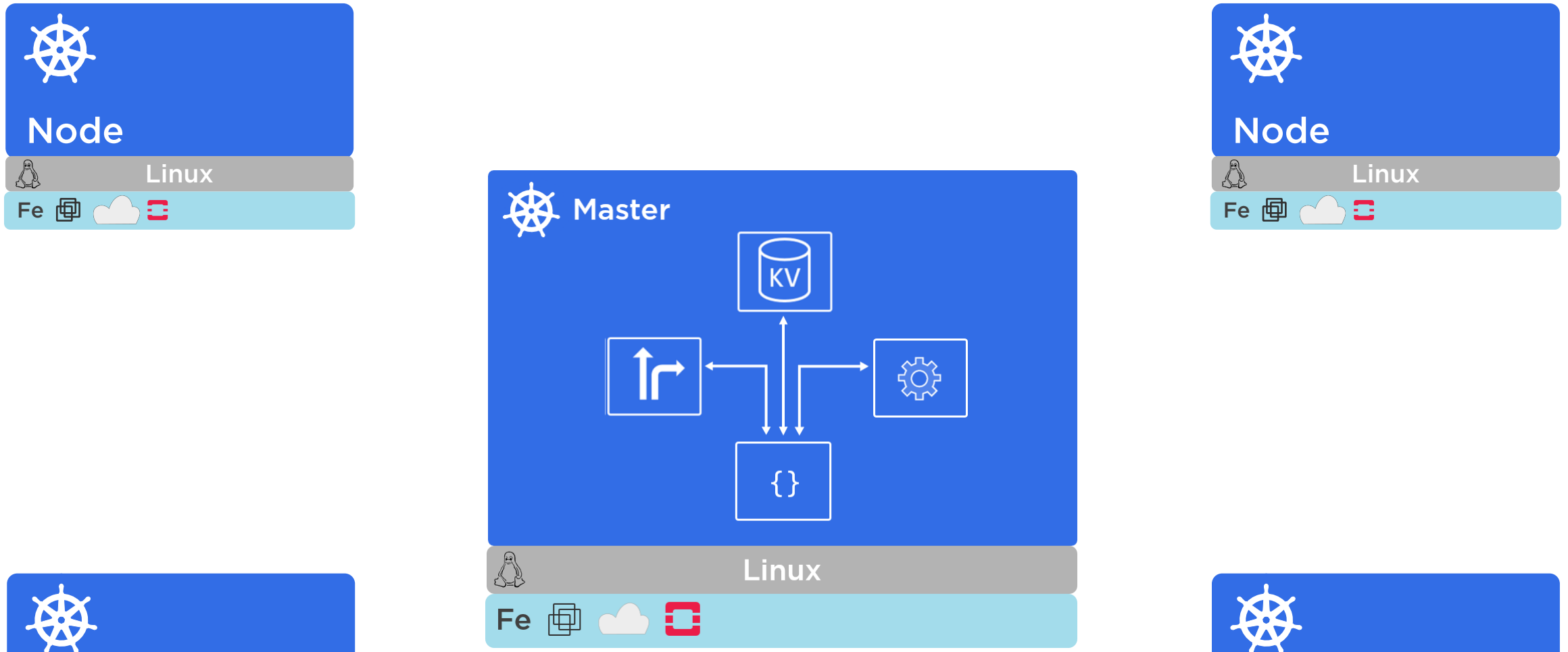
Masters

The Kubernetes Control Plane

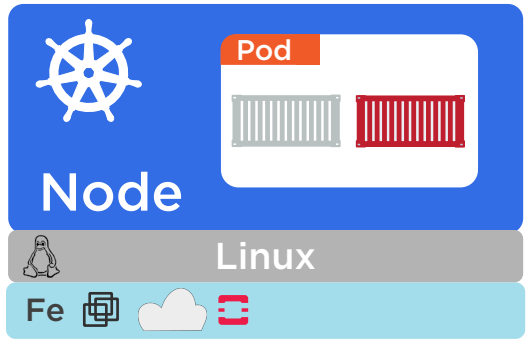

Master
KV
{}
Linux
Fe

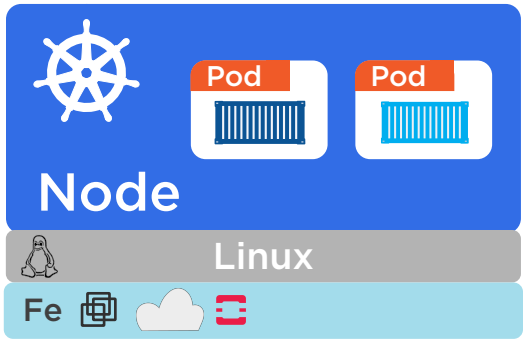
Multi-master HA

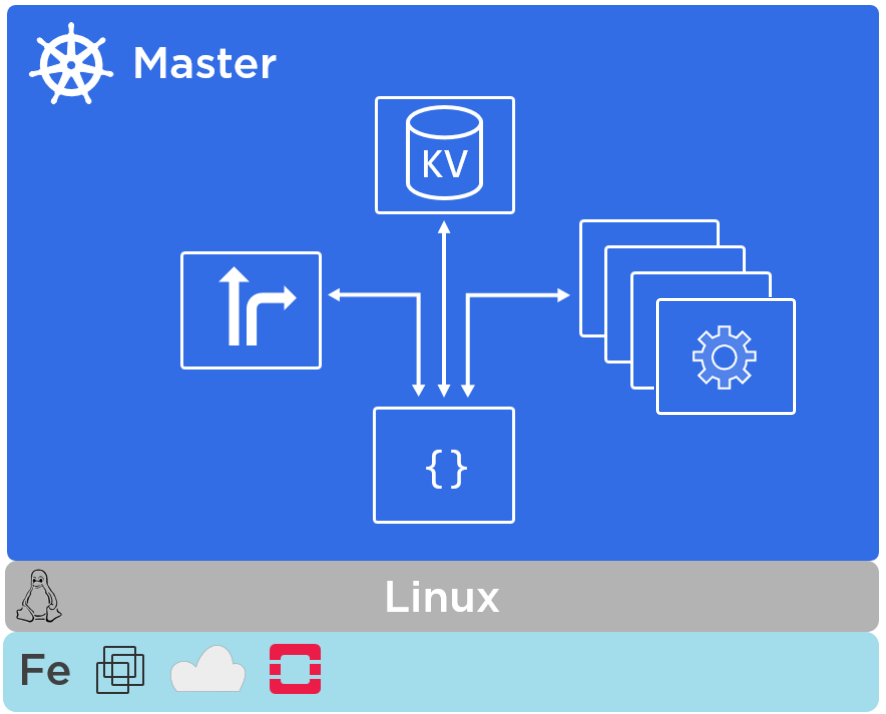


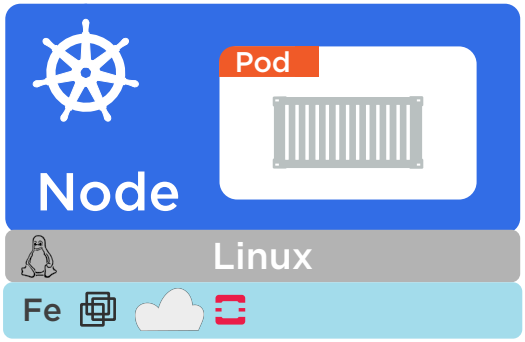


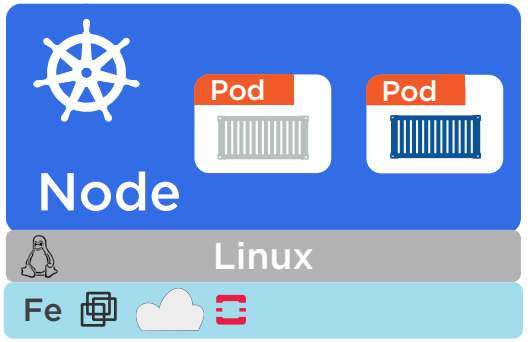
Distributed Control Plane
(future)



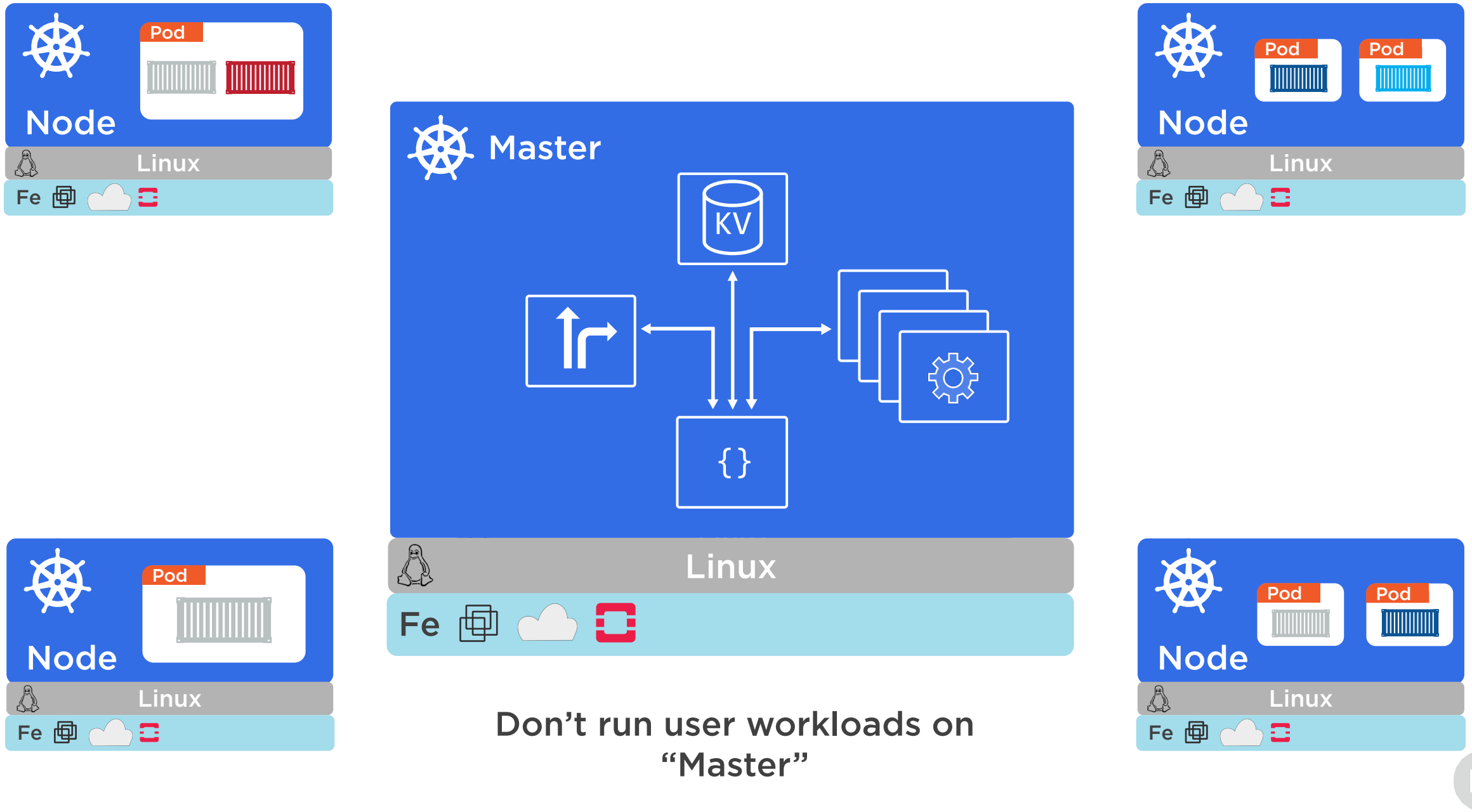




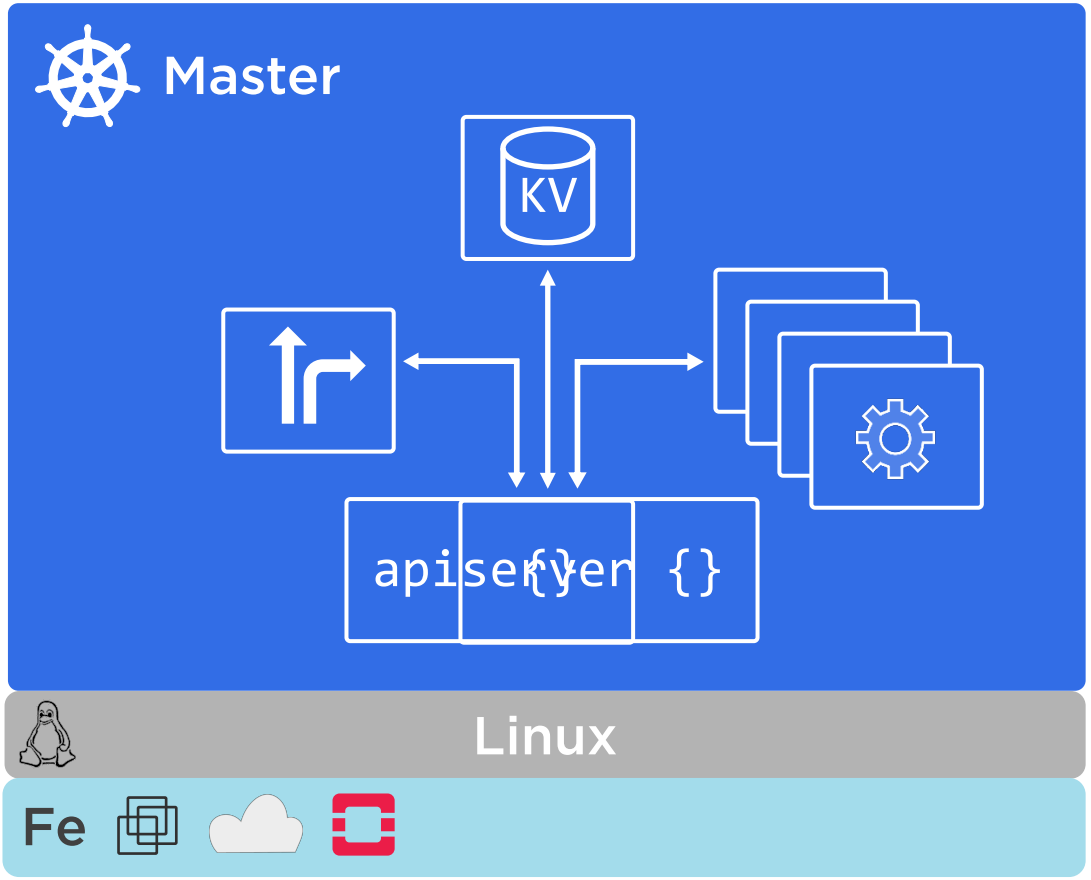


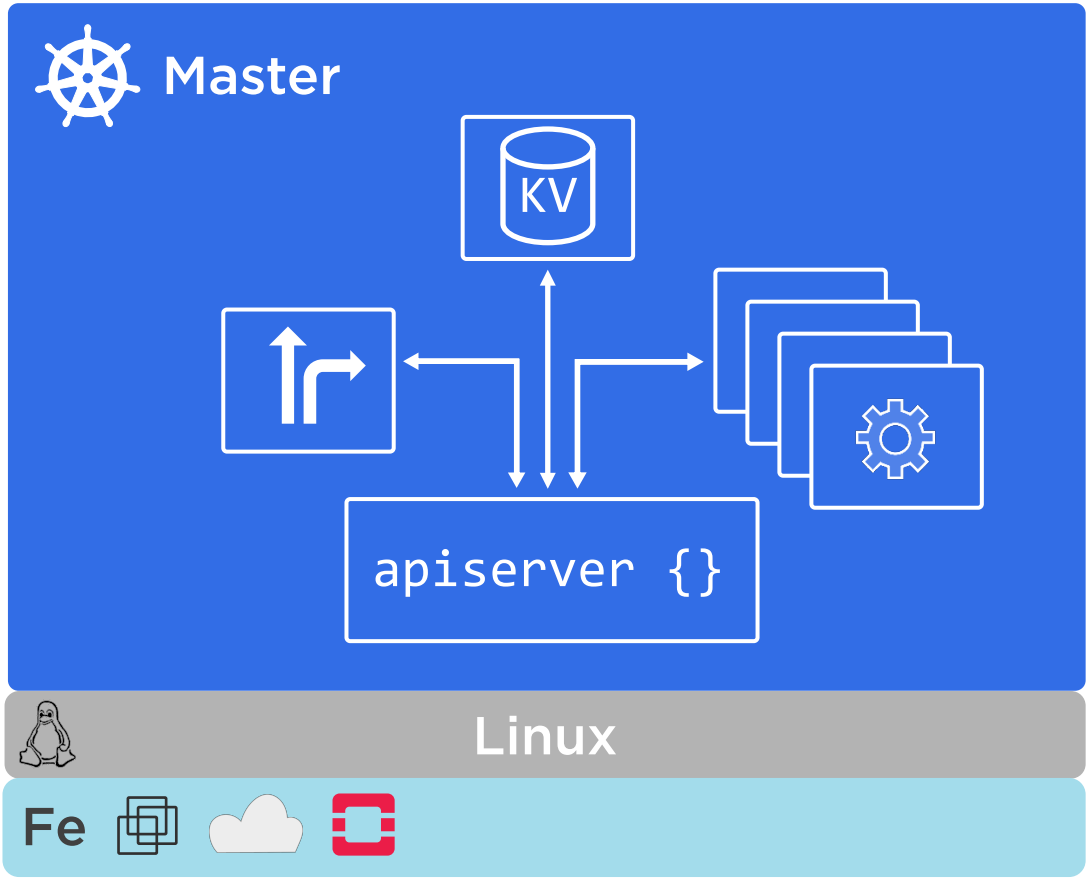


Don’t run user workloads on “Master”



Don’t run user workloads on “Master”


Master
KV
apiserver
{}
{}
Linux
Fe


Master
KV
apiserver {}
Linux
Fe

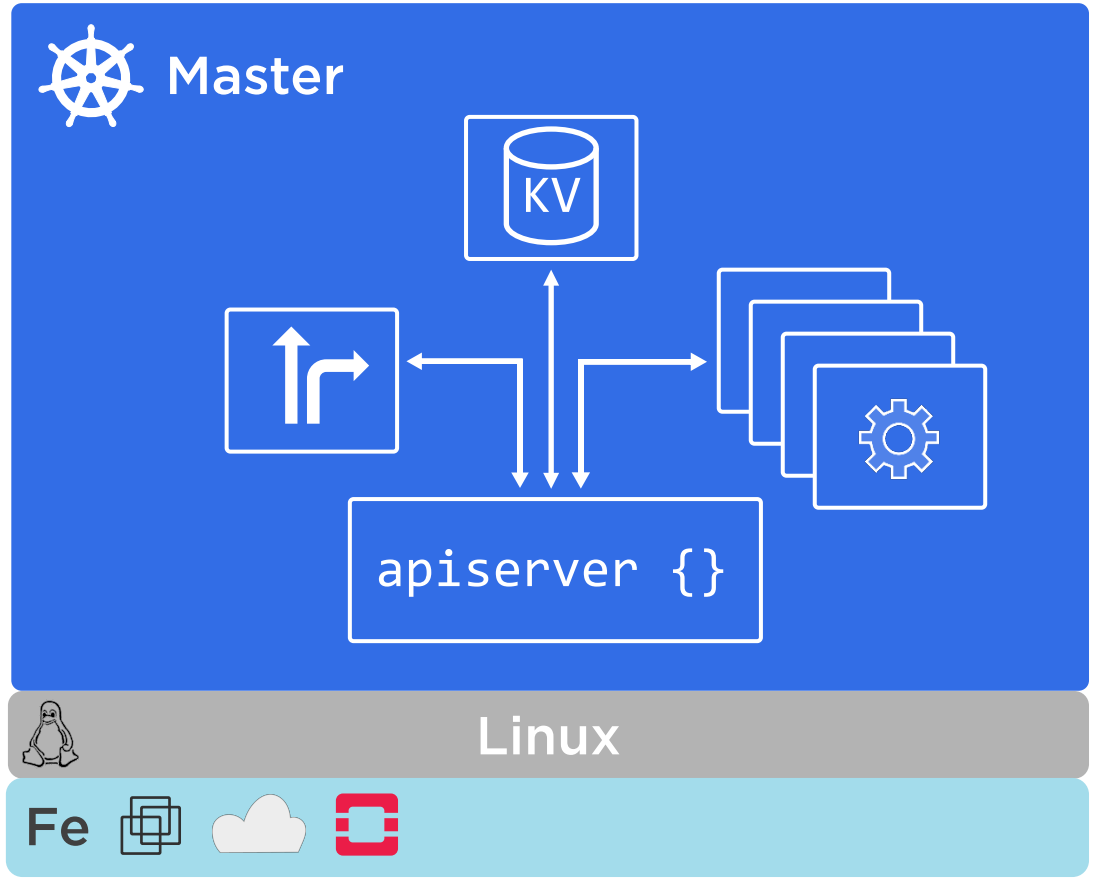
kube-apiserver

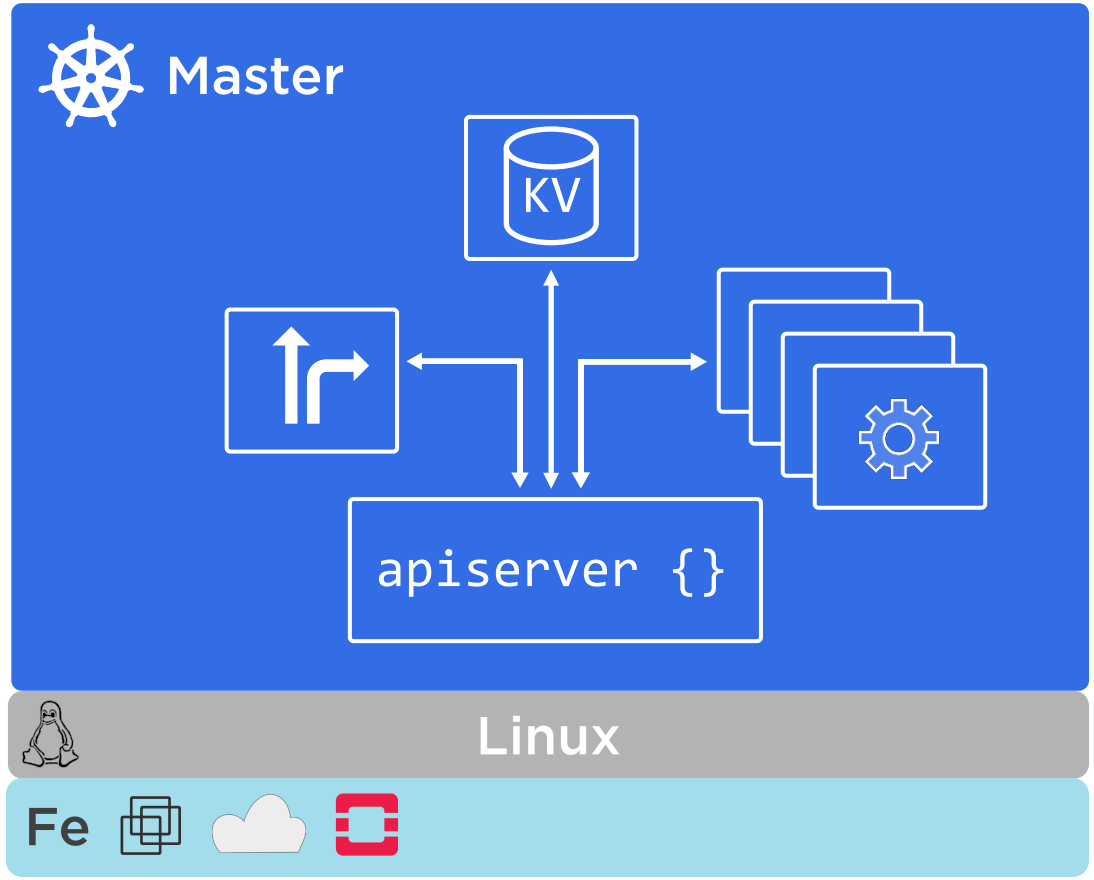
Front-end to the control plane

Exposes the API (REST)

Consumes JSON

(via manifest files)




Master
KV
apiserver {}
Linux
Fe

Cluster store

Persistent storage

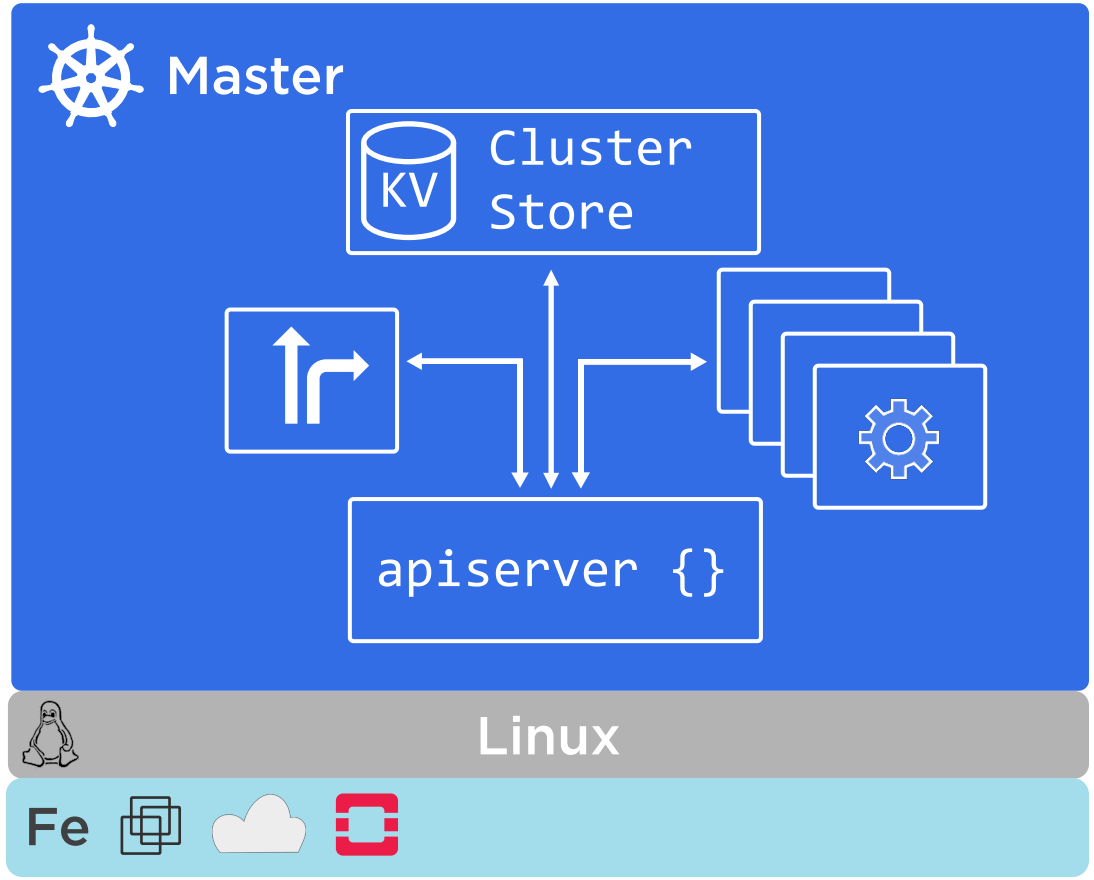
Cluster state and config

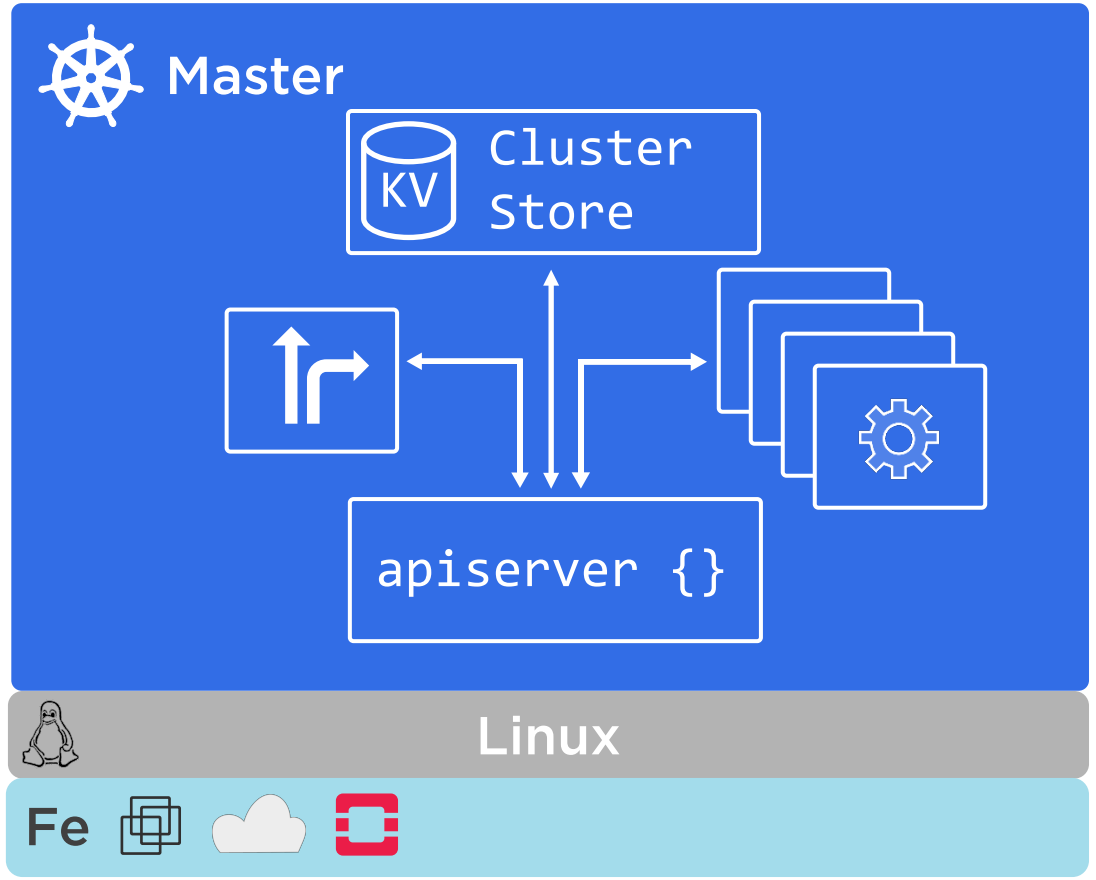
Uses **etcd**

Distributed, consistent, watchable...

The *"source of truth"* for the cluster

Have a backup plan for it!




Master
KV
Cluster
Store
apiserver {}
Linux
Fe

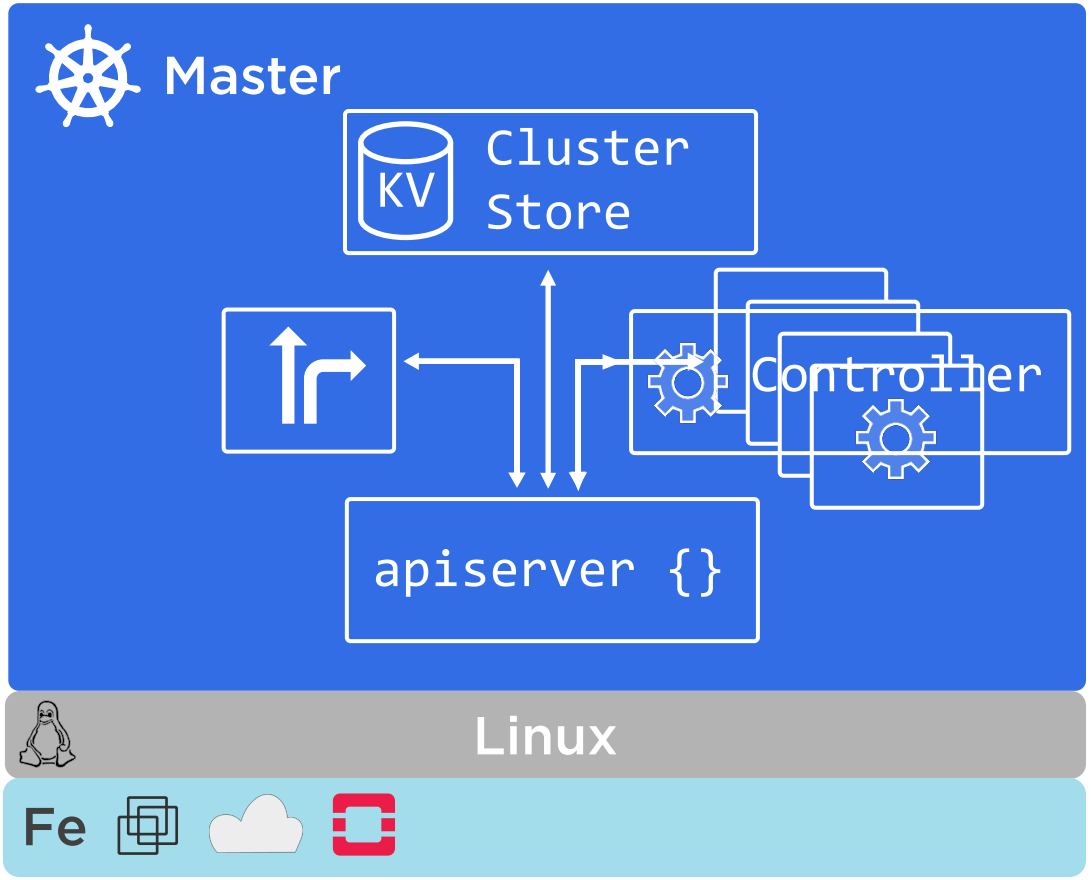
kube-controller-manager

Controller of controllers

- Node controller
- Endpoints controller
- Namespace controller
- ...

Watches for changes

Helps maintain *desired state*



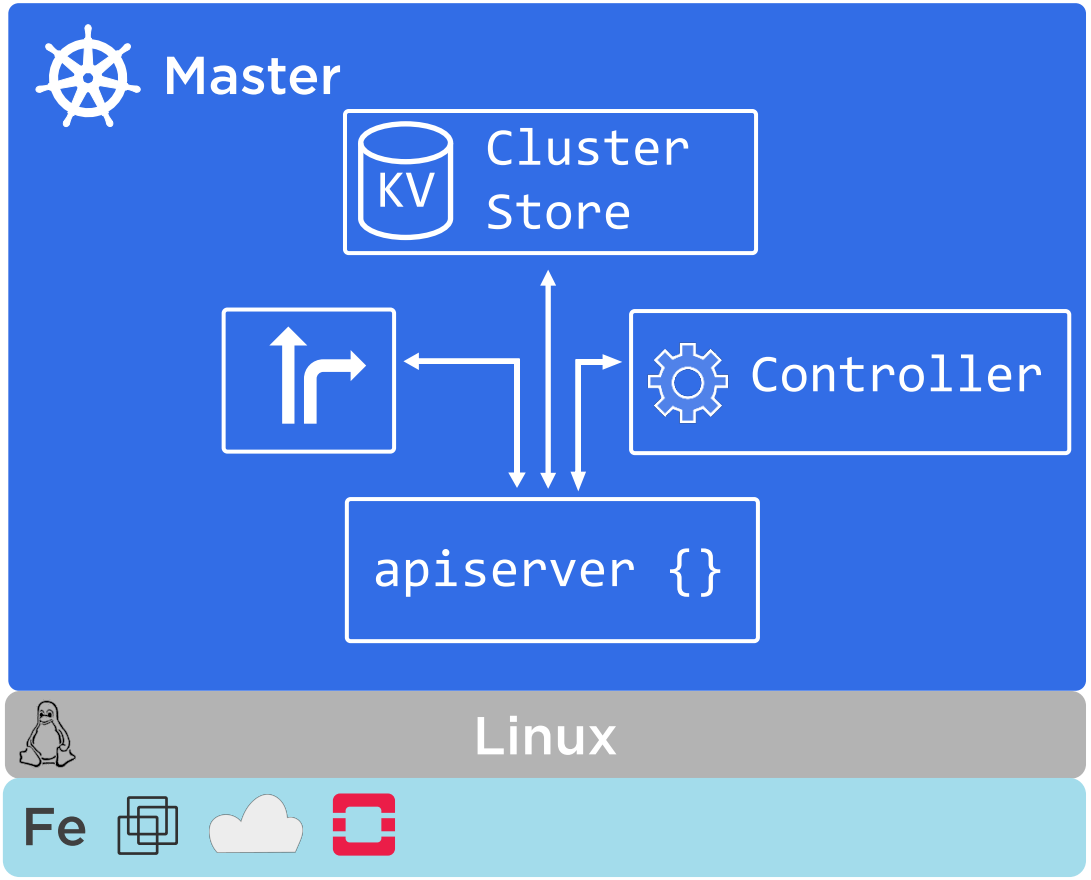
kube-controller-manager

Controller of controllers

- Node controller
- Endpoints controller
- Namespace controller
- …

Watches for changes

Helps maintain *desired state*

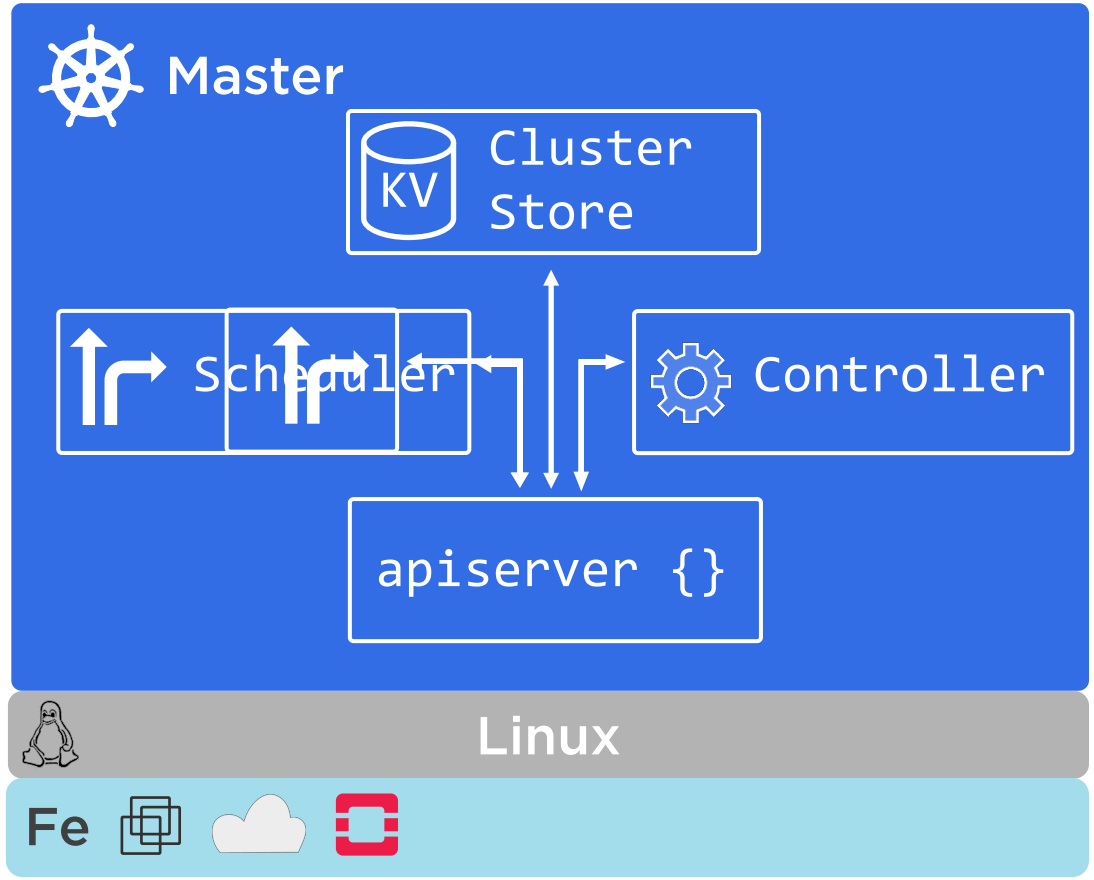


kube-scheduler

Watches **apiserver** for new pods

Assigns work to **nodes**

- affinity/anti-affinity
- constraints
- resources
- ...

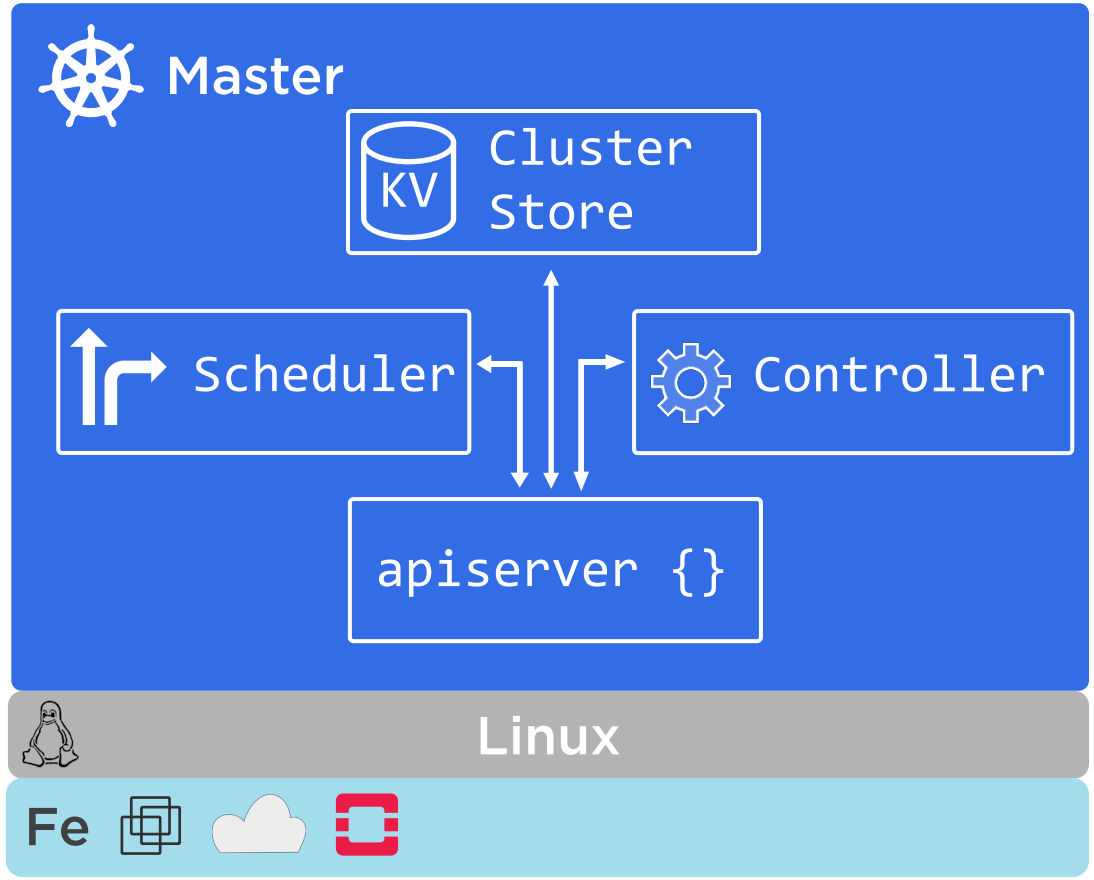


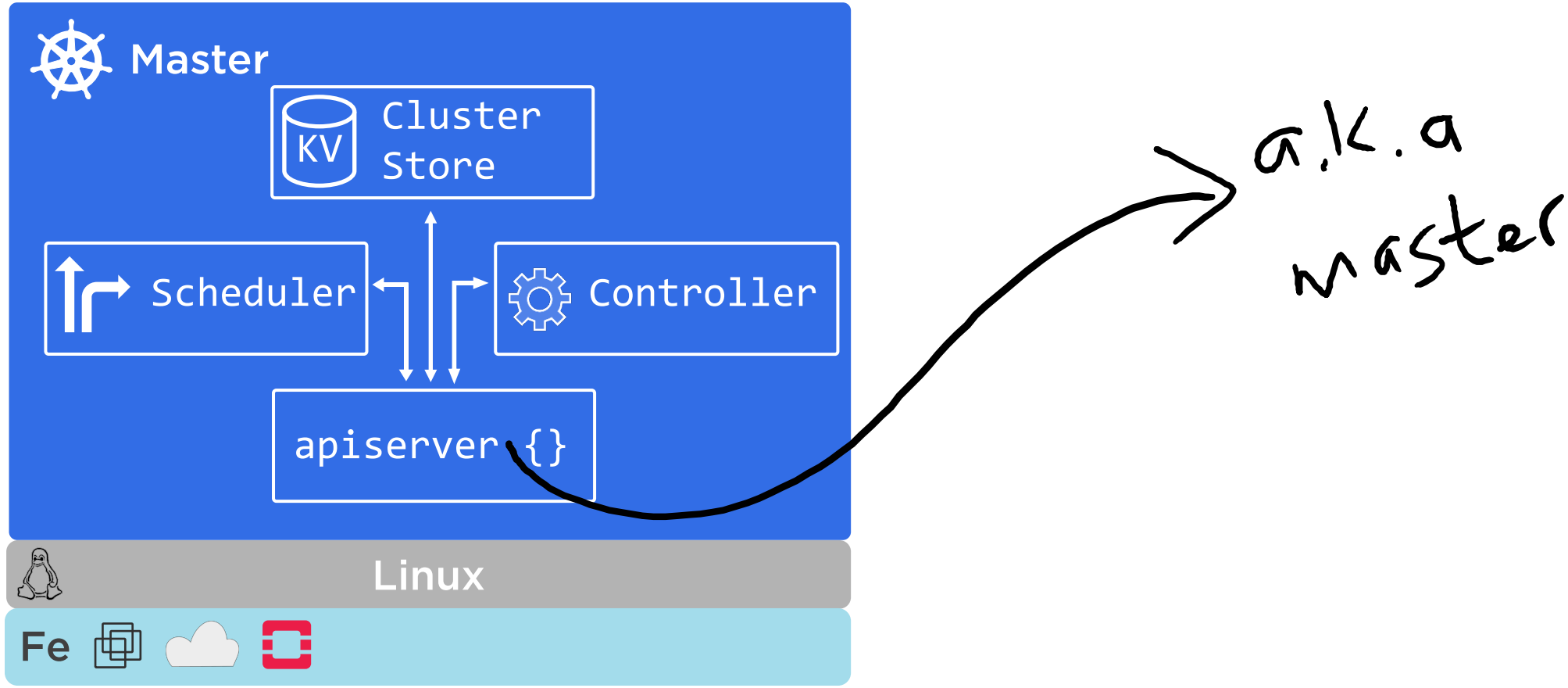
kube-scheduler

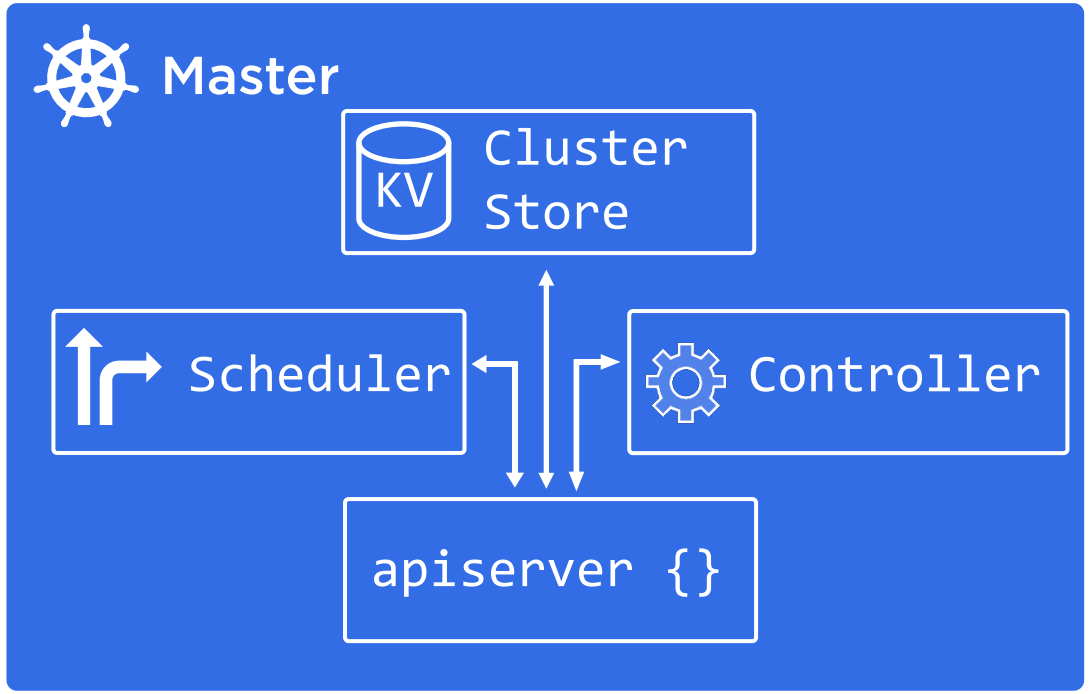
Watches **apiserver** for new pods

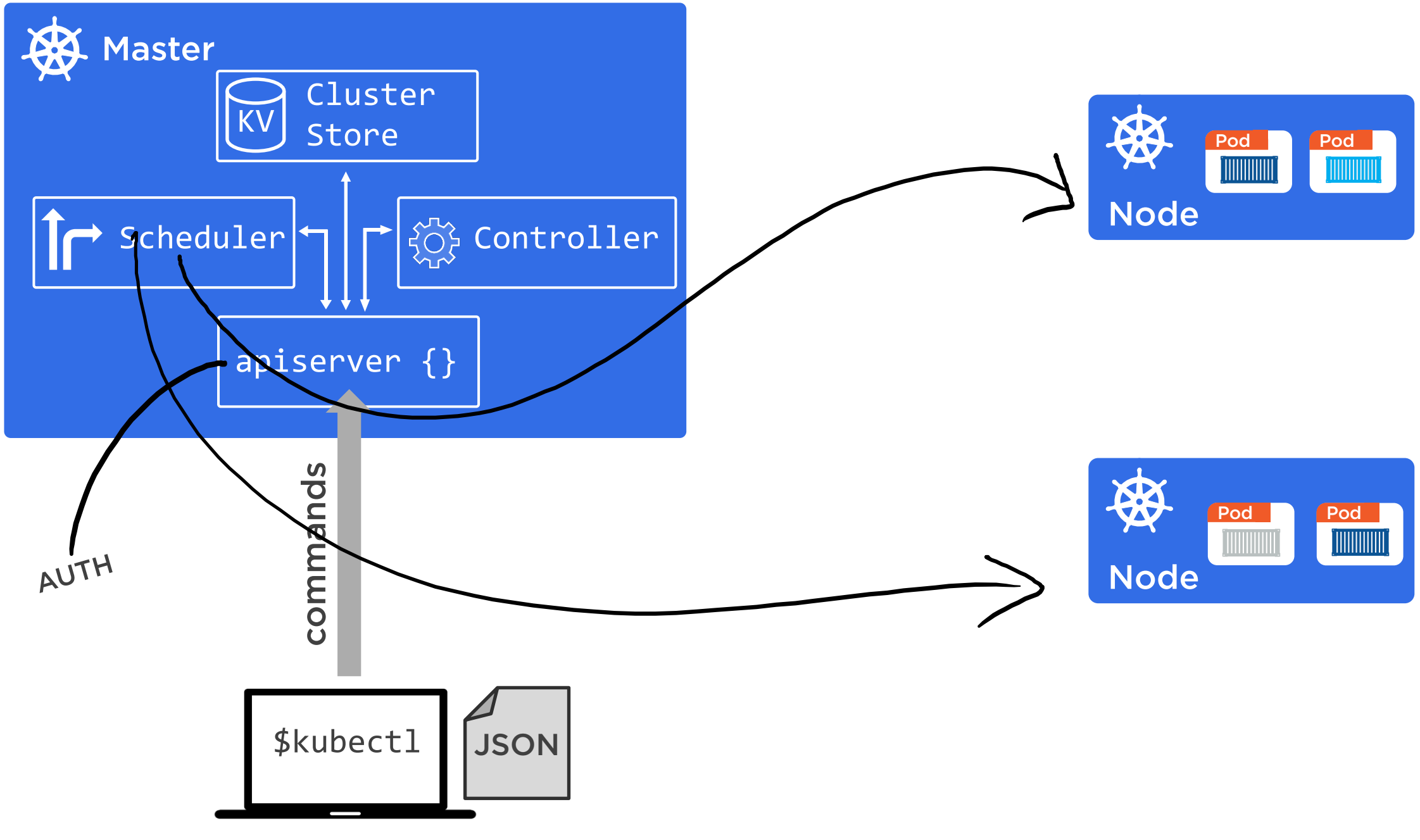
Assigns work to **nodes**

- affinity/anti-affinity
- constraints
- resources
- ...




Master
KV
Cluster
Store
Scheduler
Controller
apiserver {}
Linux
Fe
a.k.a
master


Master
KV
Cluster
Store
Scheduler
Controller
apiserver {}

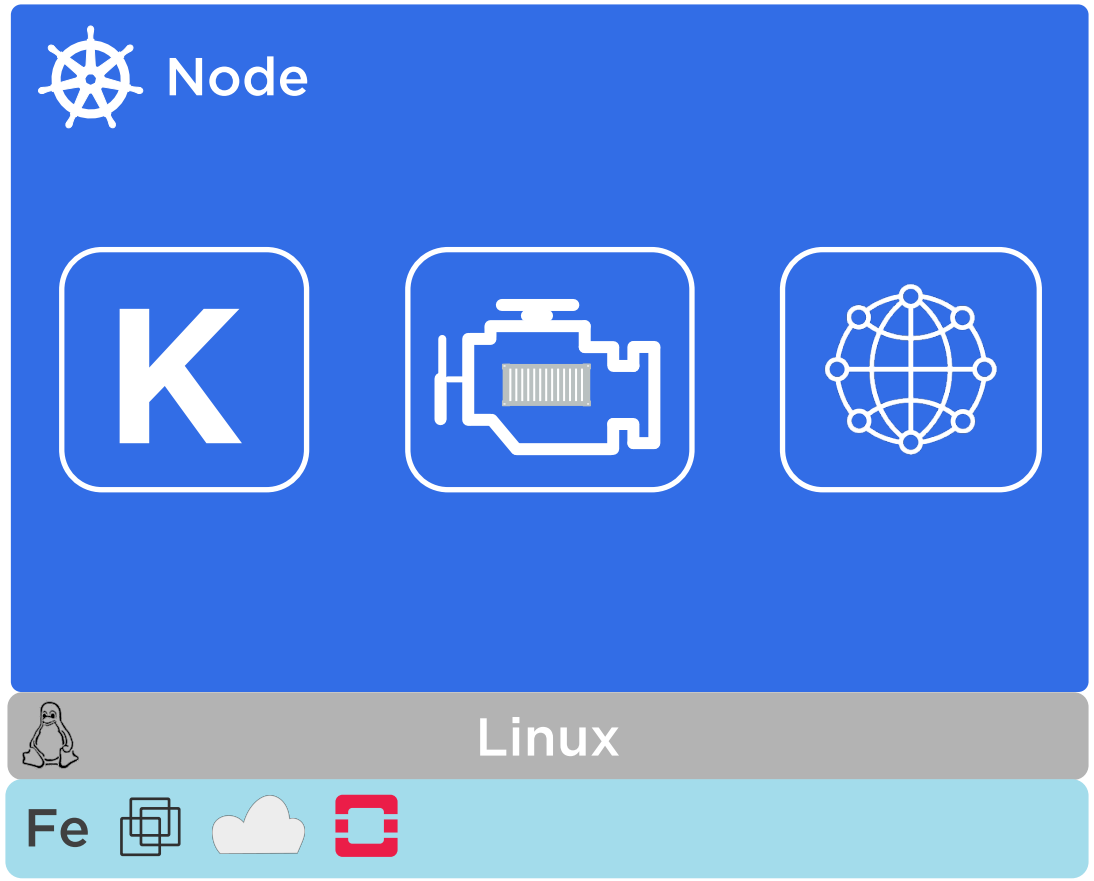

Master
KV
Cluster
Store
Scheduler
Controller
apiserver {}
AUTH
commands
$kubectl
JSON
Pod
Pod
Node
Pod
Pod
Node

Nodes
a.k.a. “Minions”

The Kubernetes Workers

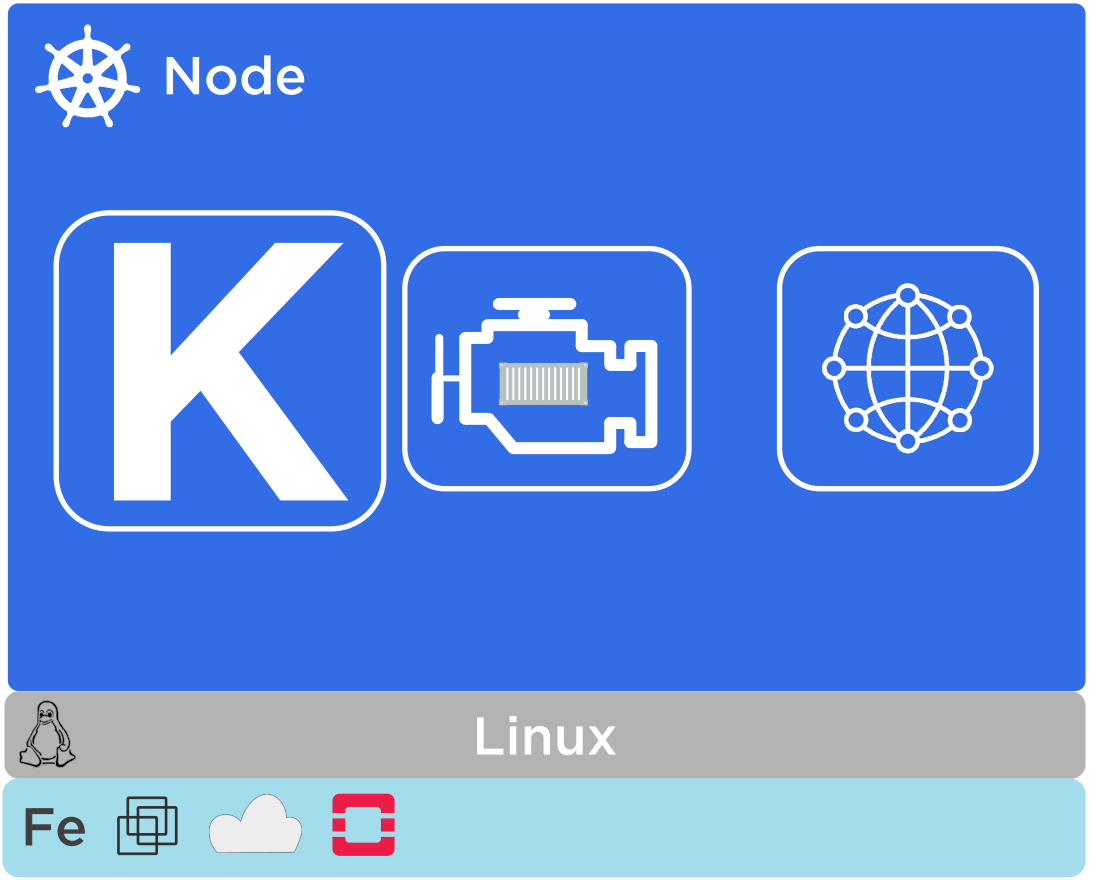
Nodes
a.k.a. “Minions”

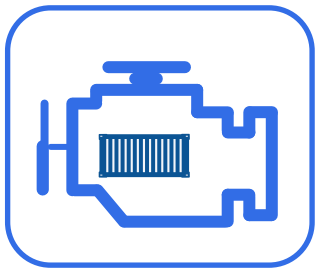
The Kubernetes Workers


Node
K
Linux
Fe

Kubelet

- The **main** Kubernetes agent
- Registers node with cluster
- Watches **apiserver**
- Instantiates **pods**
- Reports back to **master**
- Exposes endpoint on :10255





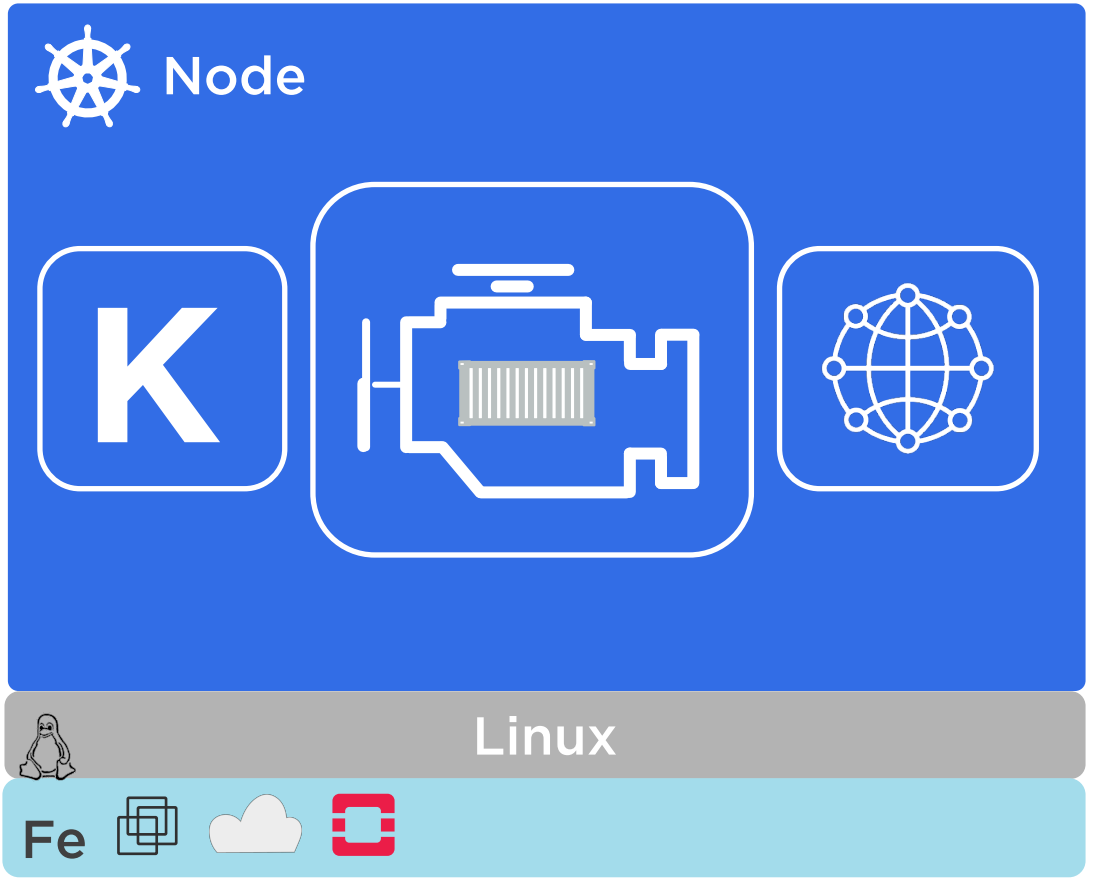
Container Engine

Does container management:

- Pulling images
- Starting/stopping containers
- ...

Pluggable:

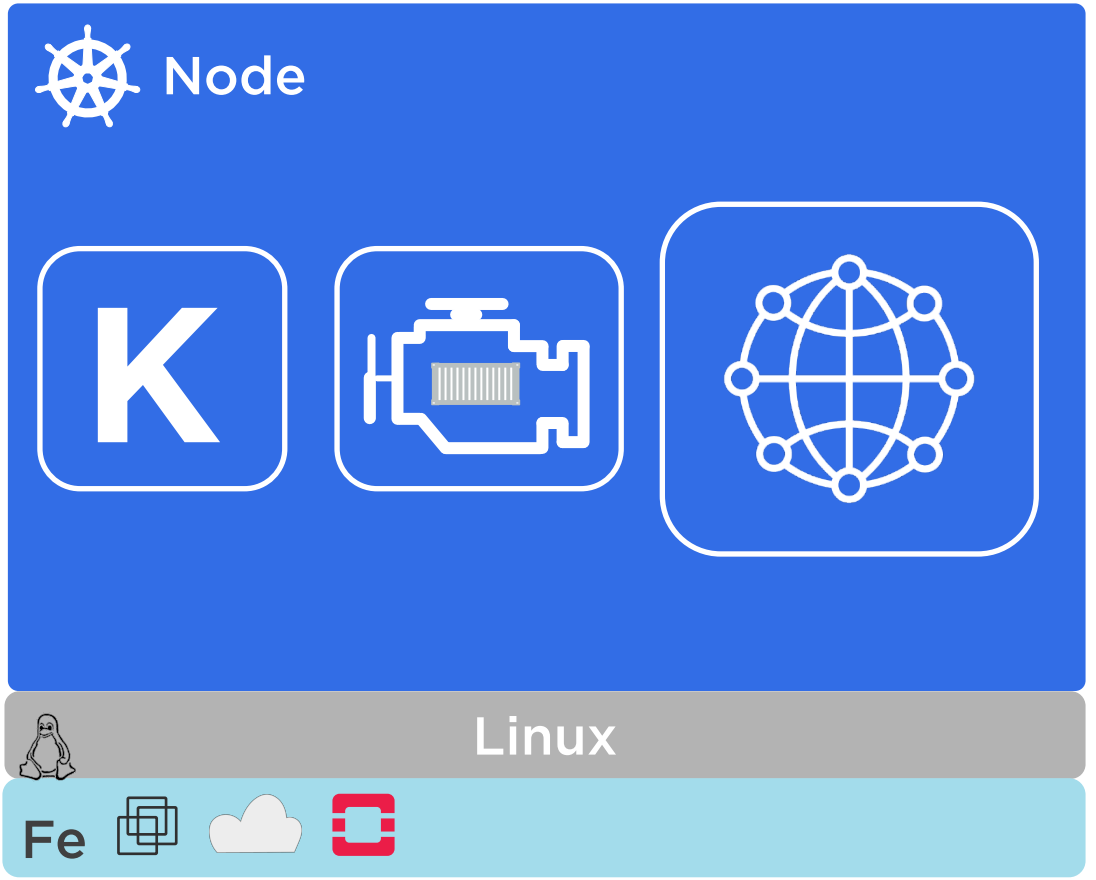
- Usually **Docker**
- Can be **rkt**



kube-proxy

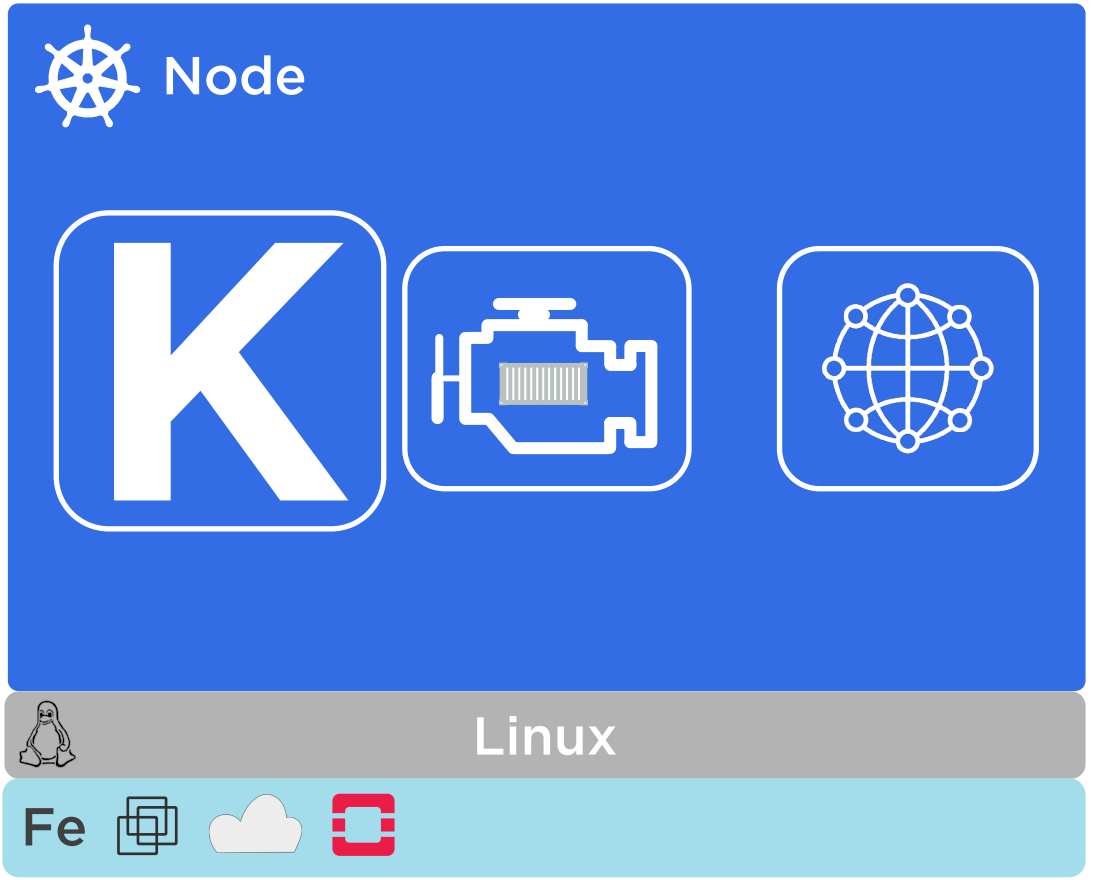
Kubernetes networking:

- Pod IP addresses
  - All containers in a pod share a single IP
- Load balances across all pods in a **service**



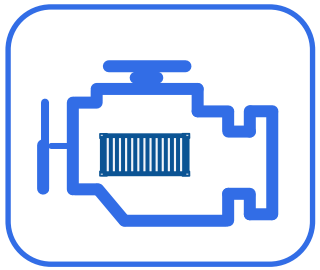
Kubelet

Main Kubernetes agent



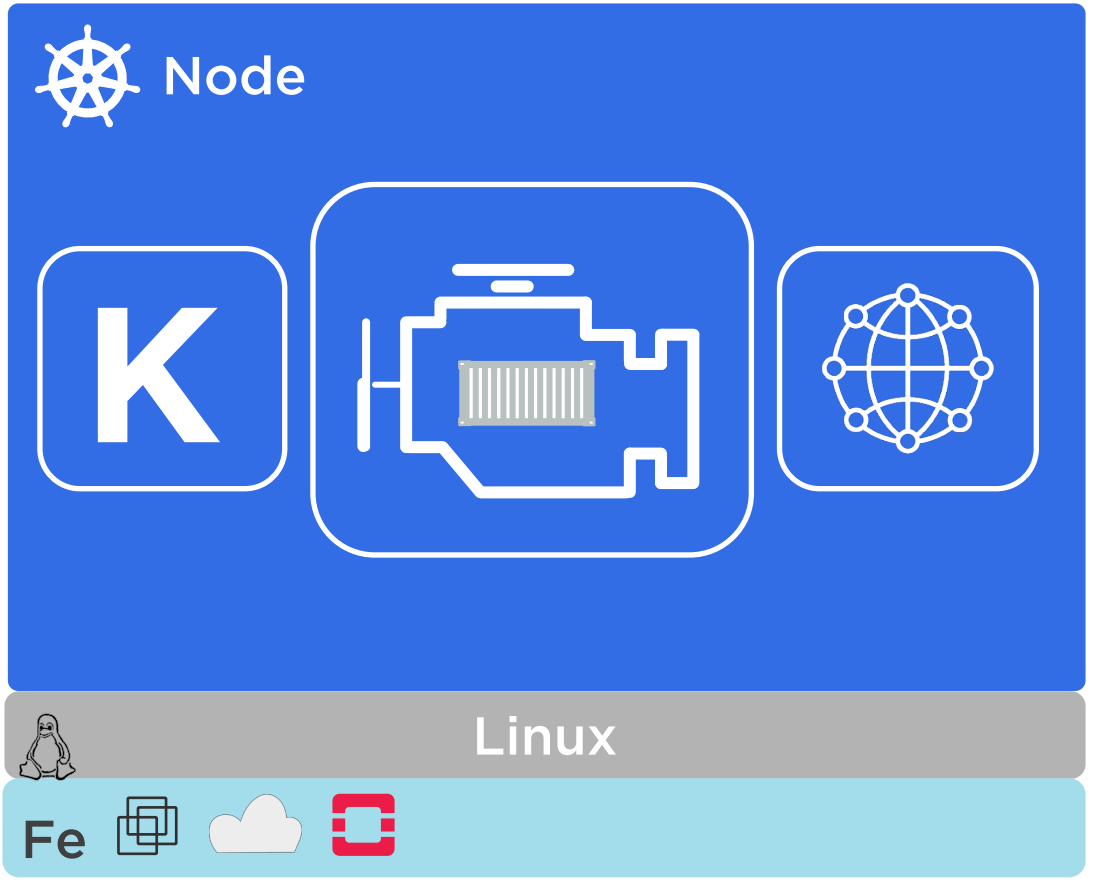
Kubelet

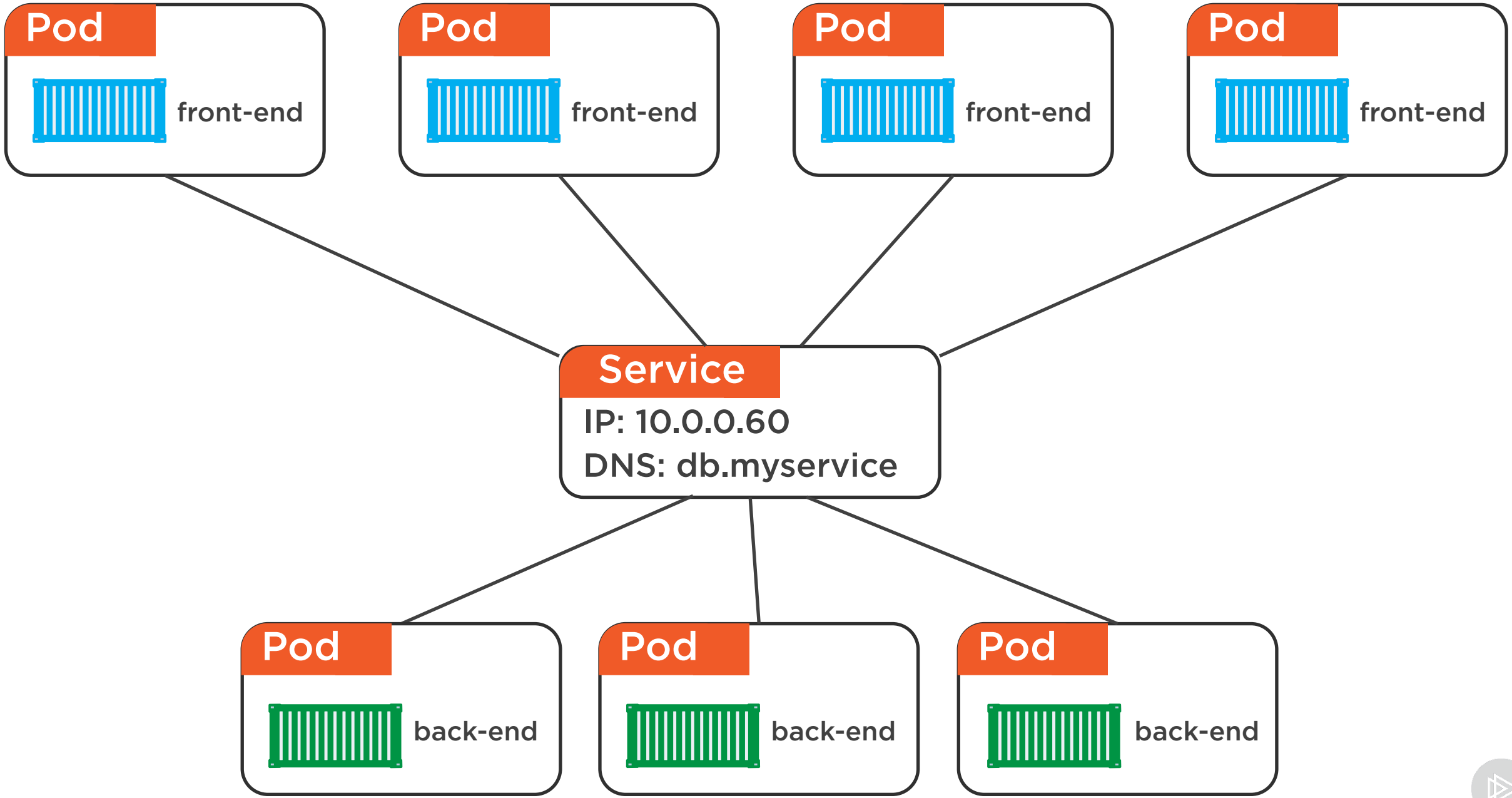
Main Kubernetes agent



Container engine

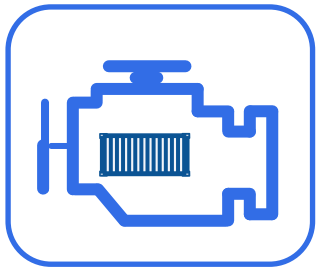
Docker or rkt




Pod
front-end
Pod
front-end
Pod
front-end
Pod
front-end
Service
IP: 10.0.0.60
DNS: db.myservice
Pod
back-end
Pod
back-end
Pod
back-end

Kubelet

Main Kubernetes agent

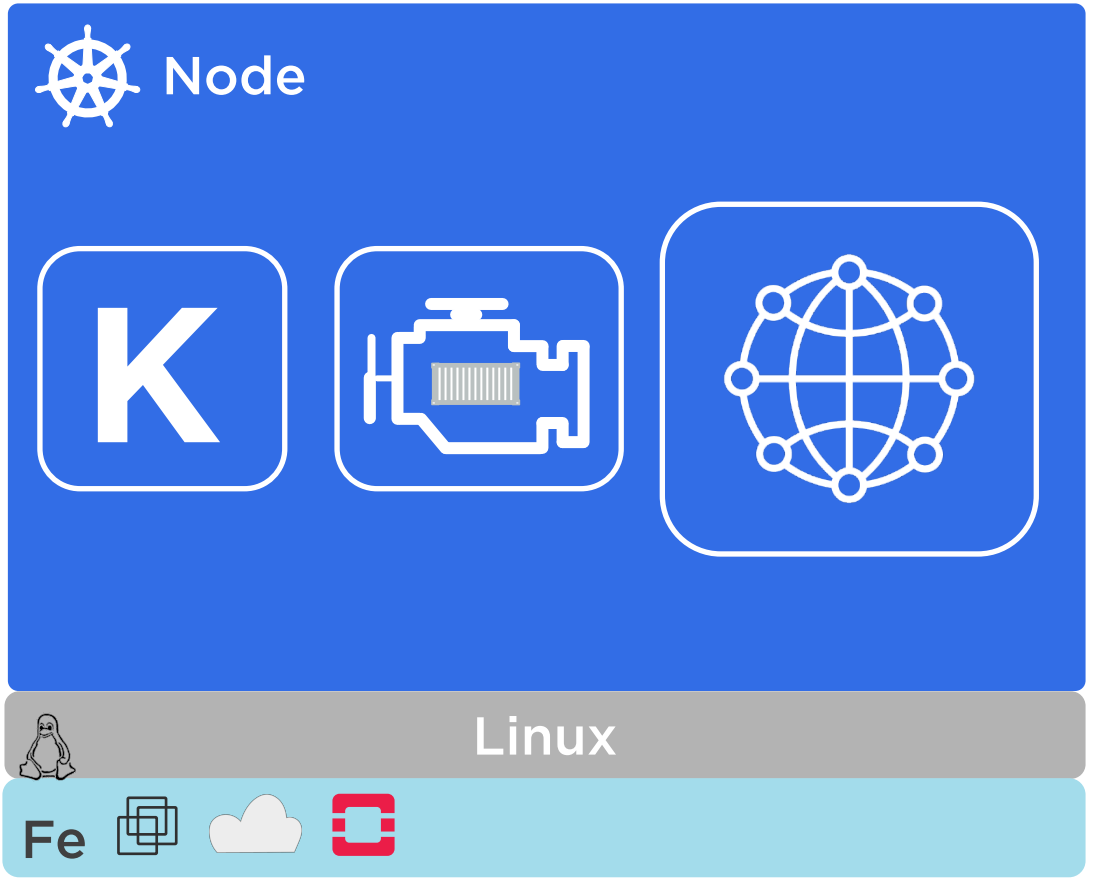


Container engine

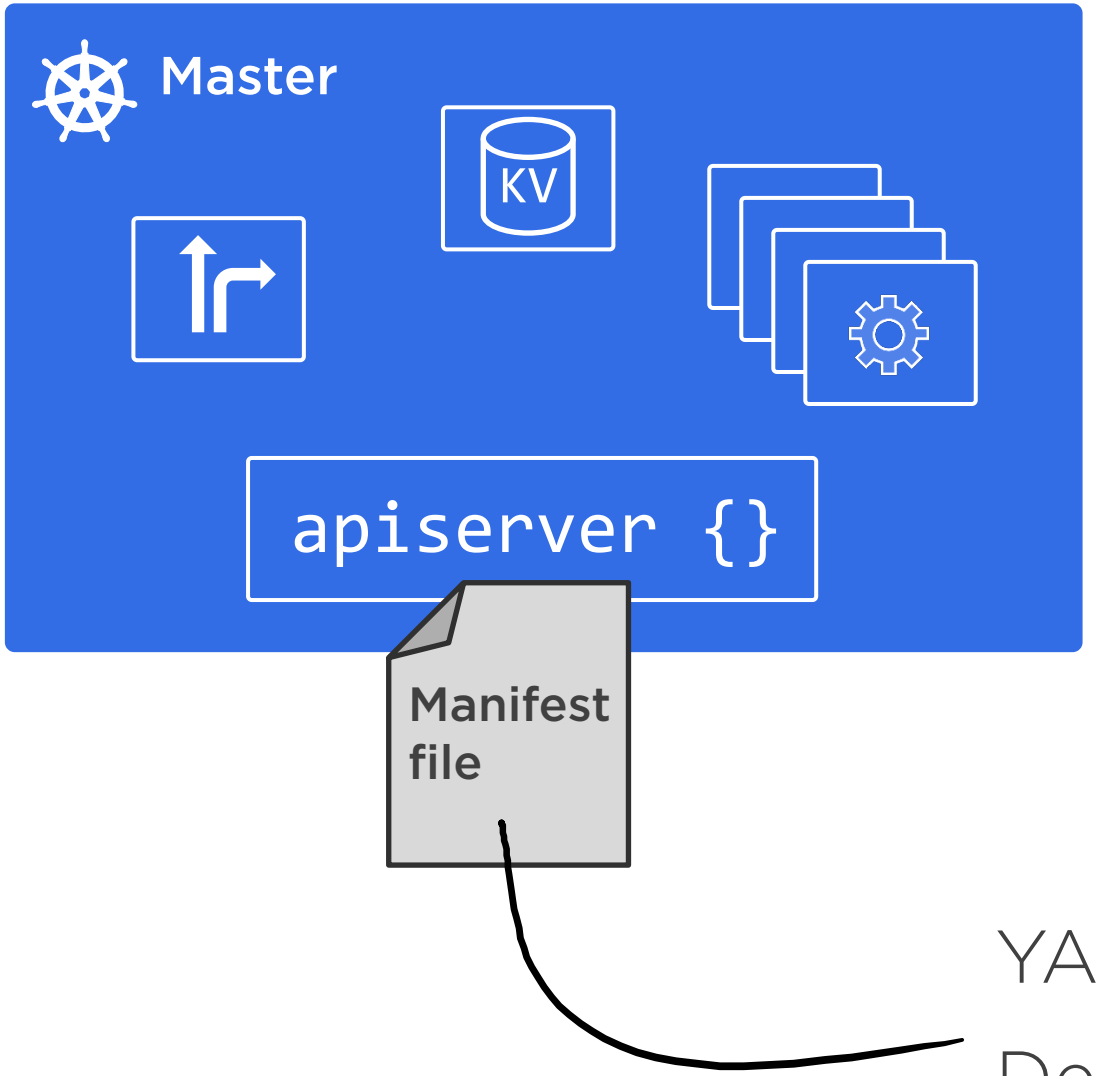
Docker or rkt

kube-proxy

Kubernetes networking

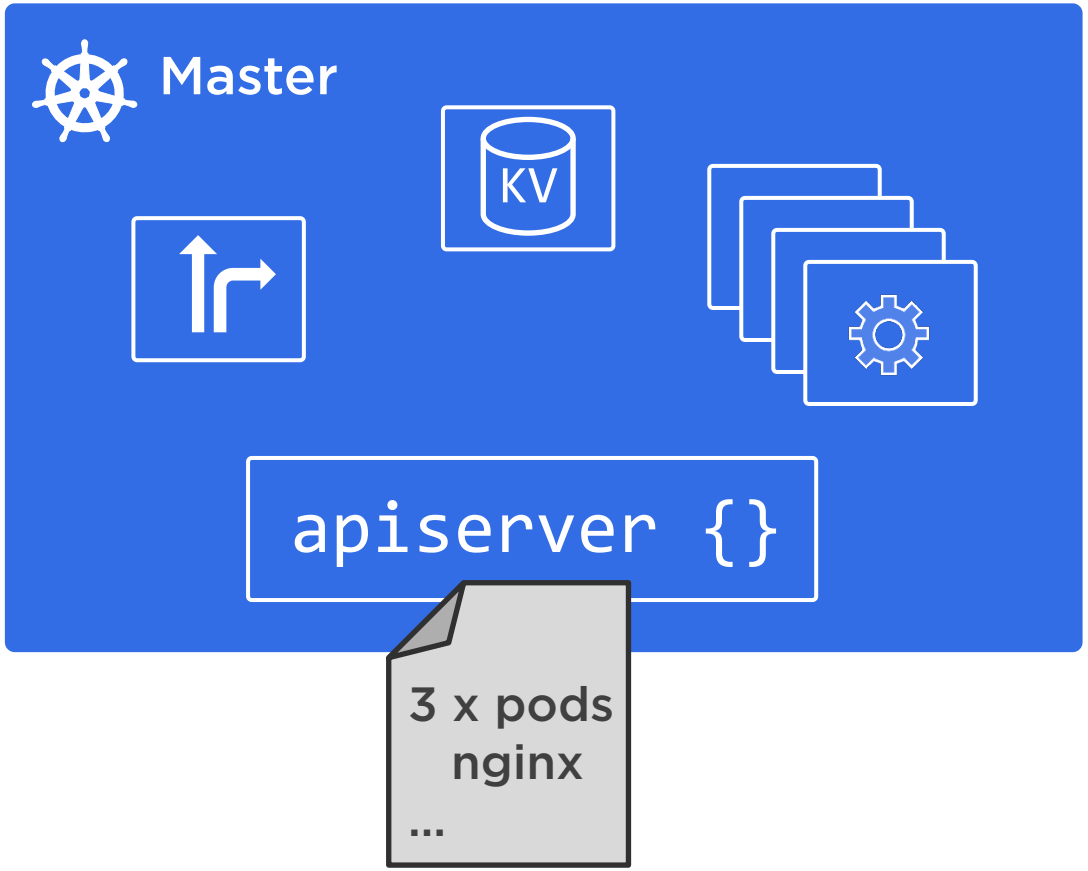


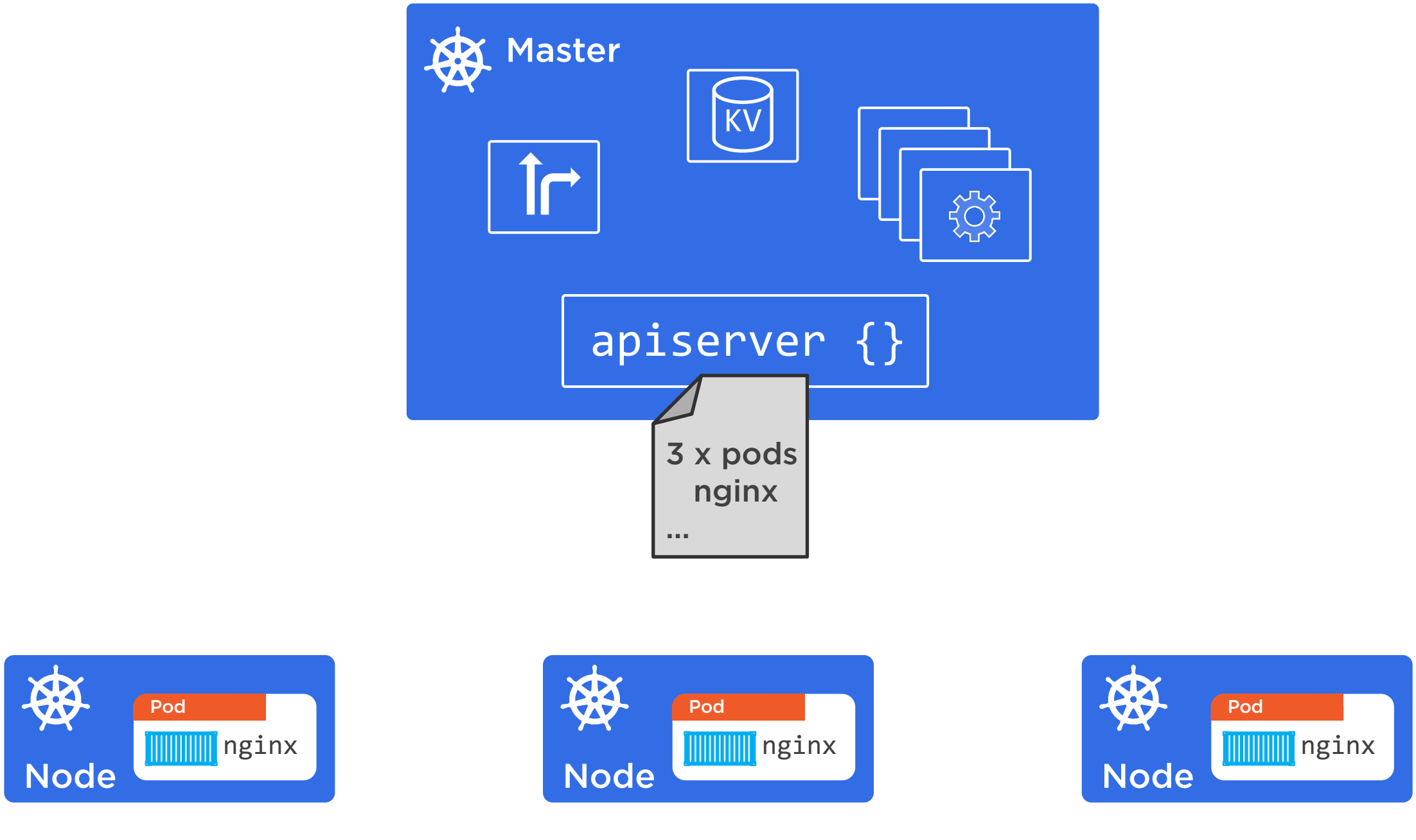
Declarative Model
&
Desired State

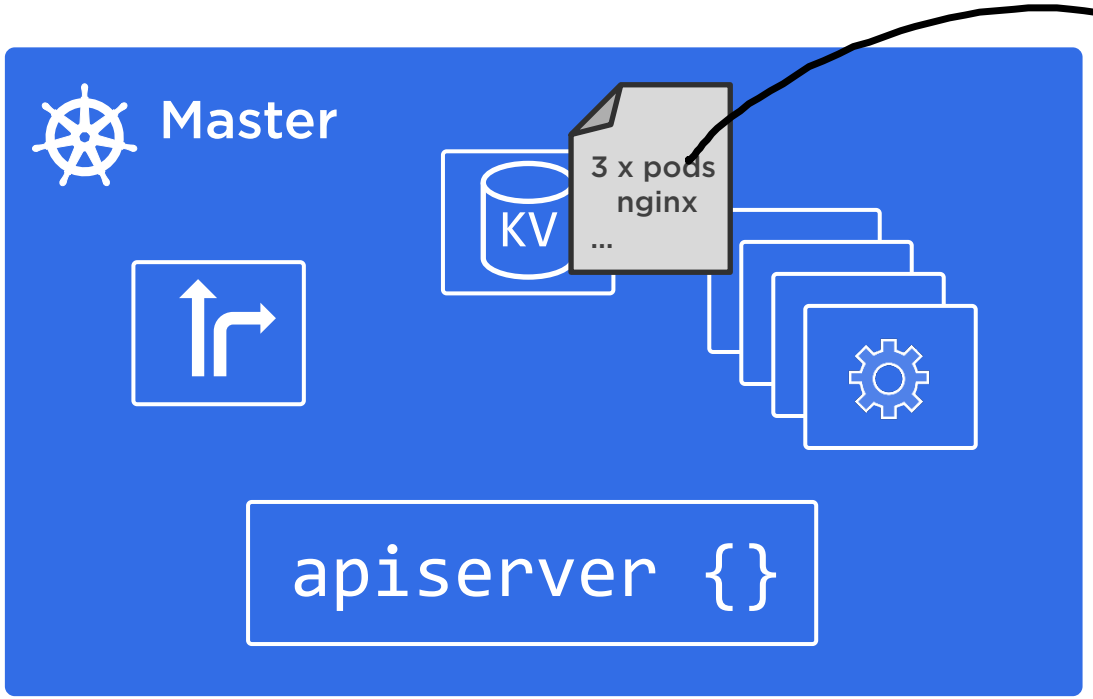


YAML or JSON

Describe **desired state**


Master
KV
apiserver {}
3 x pods
nginx
...


Master
KV
apiserver {}
3 x pods
nginx
...
Node
Pod
nginx
Node
Pod
nginx
Node
Pod
nginx

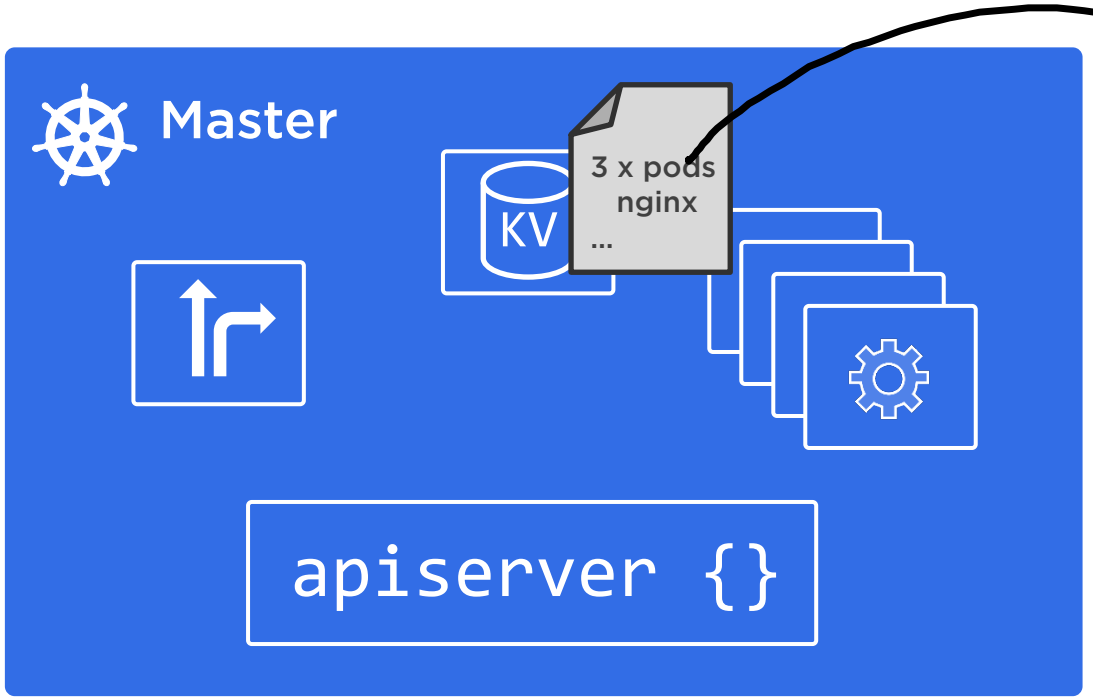


Desired state/
record of intent

- **3 x nginx pods**

Actual state

- **3 x nginx pods**

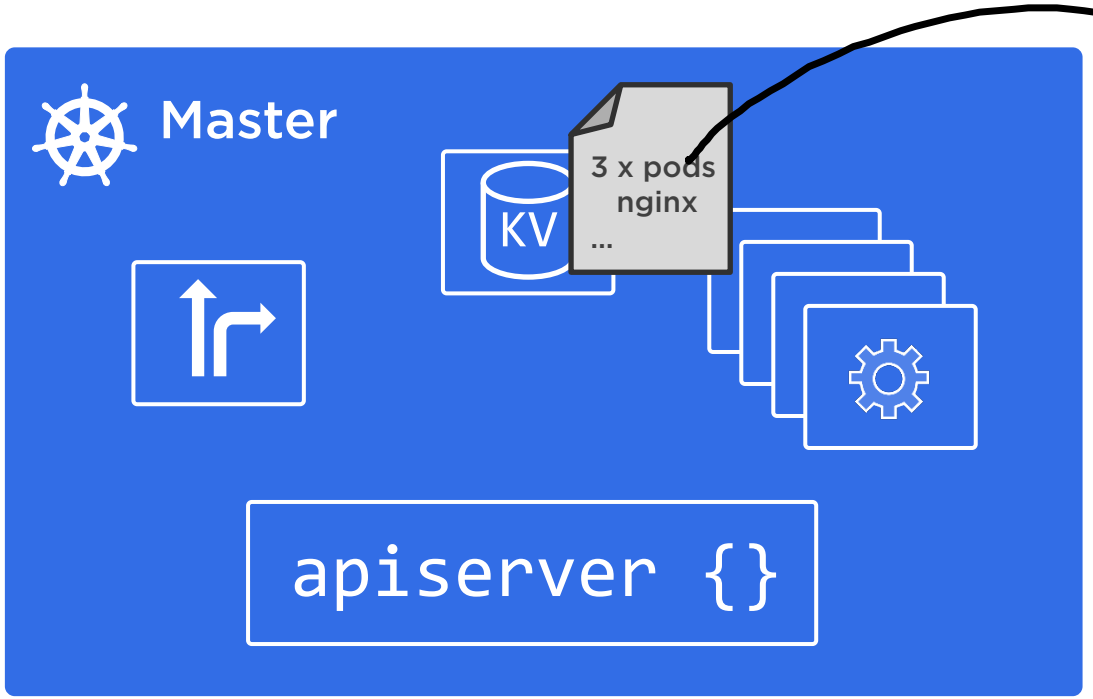


Desired state/
record of intent

- **3 x nginx pods**

Actual state

- **2 x nginx pods**

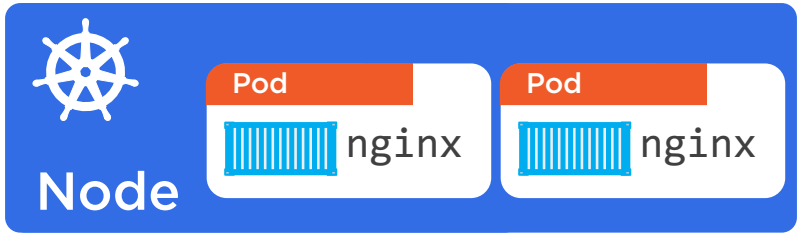


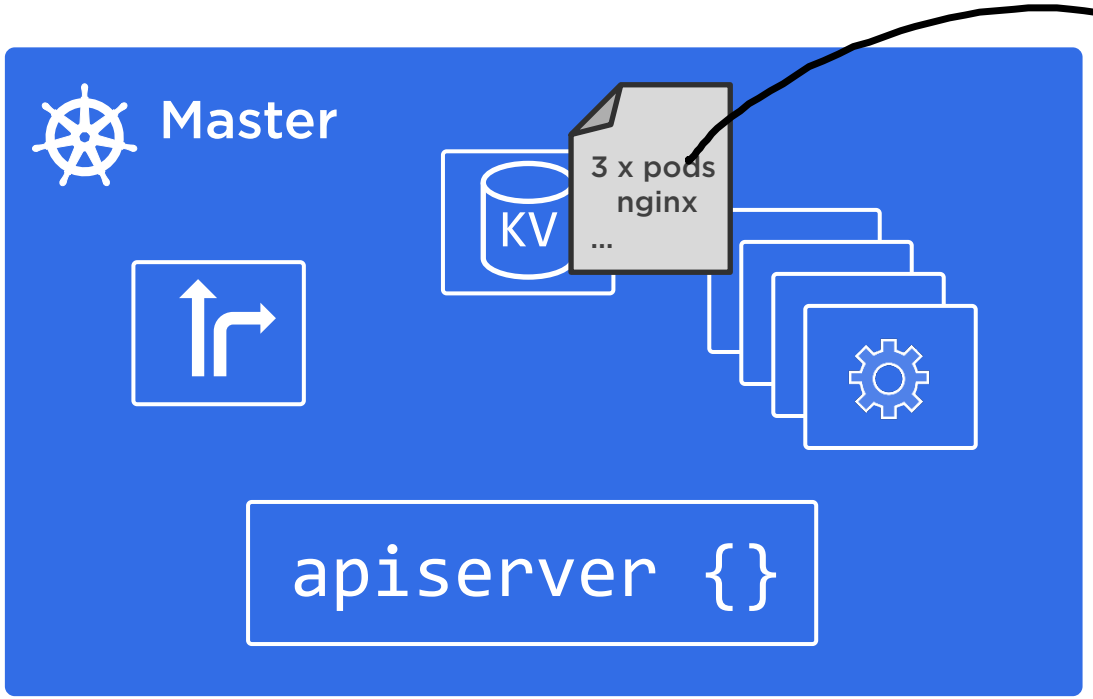
Desired state/
record of intent

- **3 x nginx pods**

Actual state

- **2 x nginx pods**



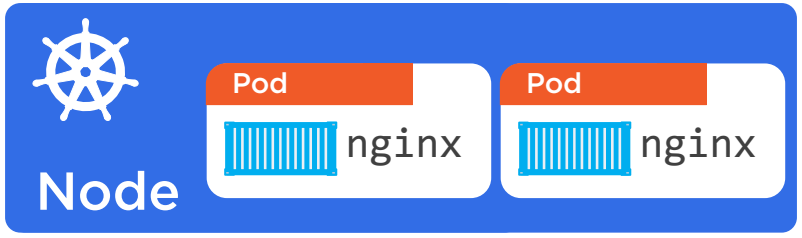


Desired state/
record of intent

- **3 x nginx pods**

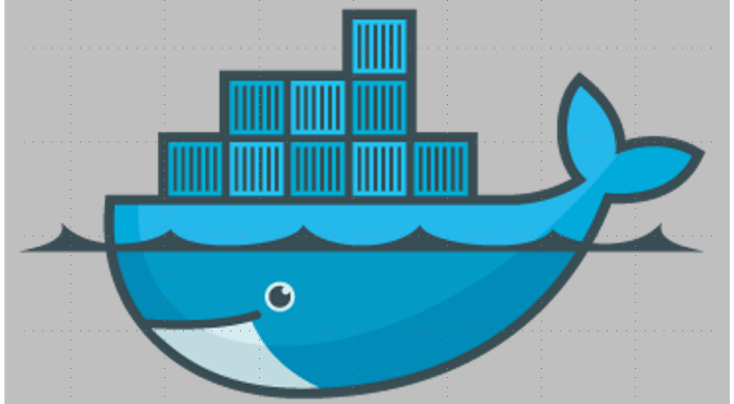
Actual state

- **3 x nginx pods**




Pods

VM

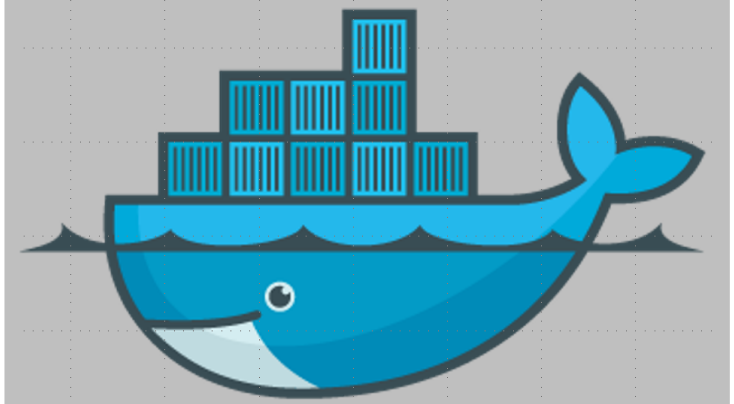


Container

Pod

Atomic units of scheduling

VM



Container

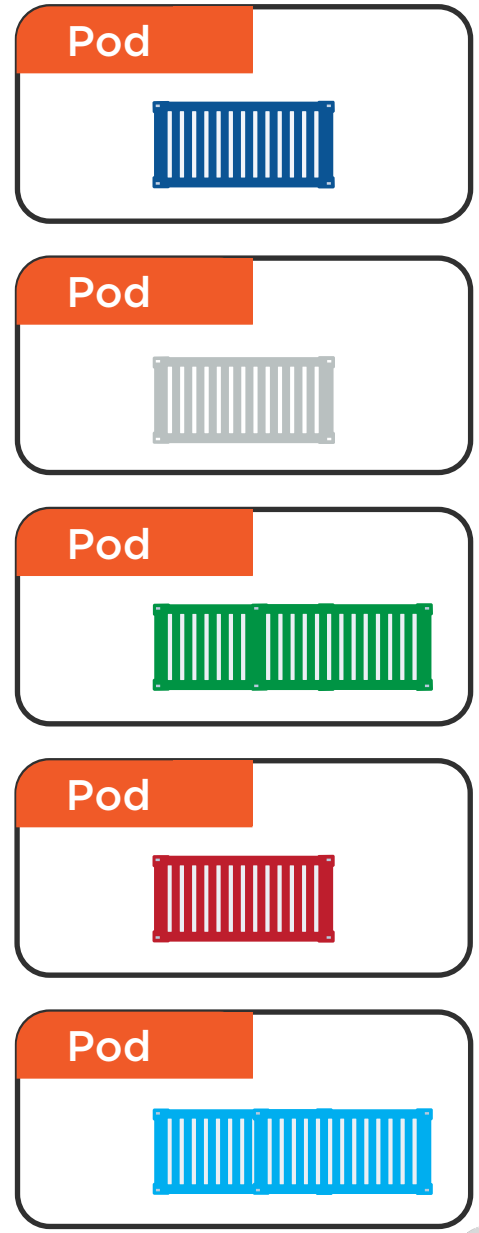
Pod

Atomic units of scheduling

Containers always run inside of pods

Pods can have multiple containers

(advanced use-case)

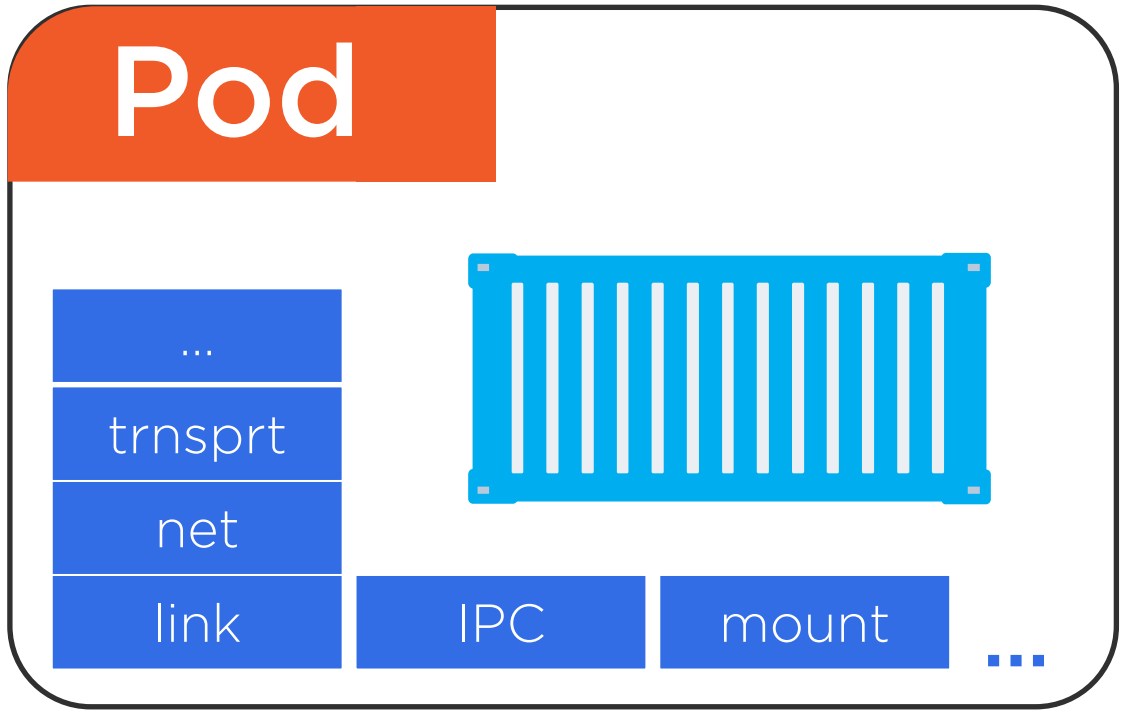


Ring-fenced environment

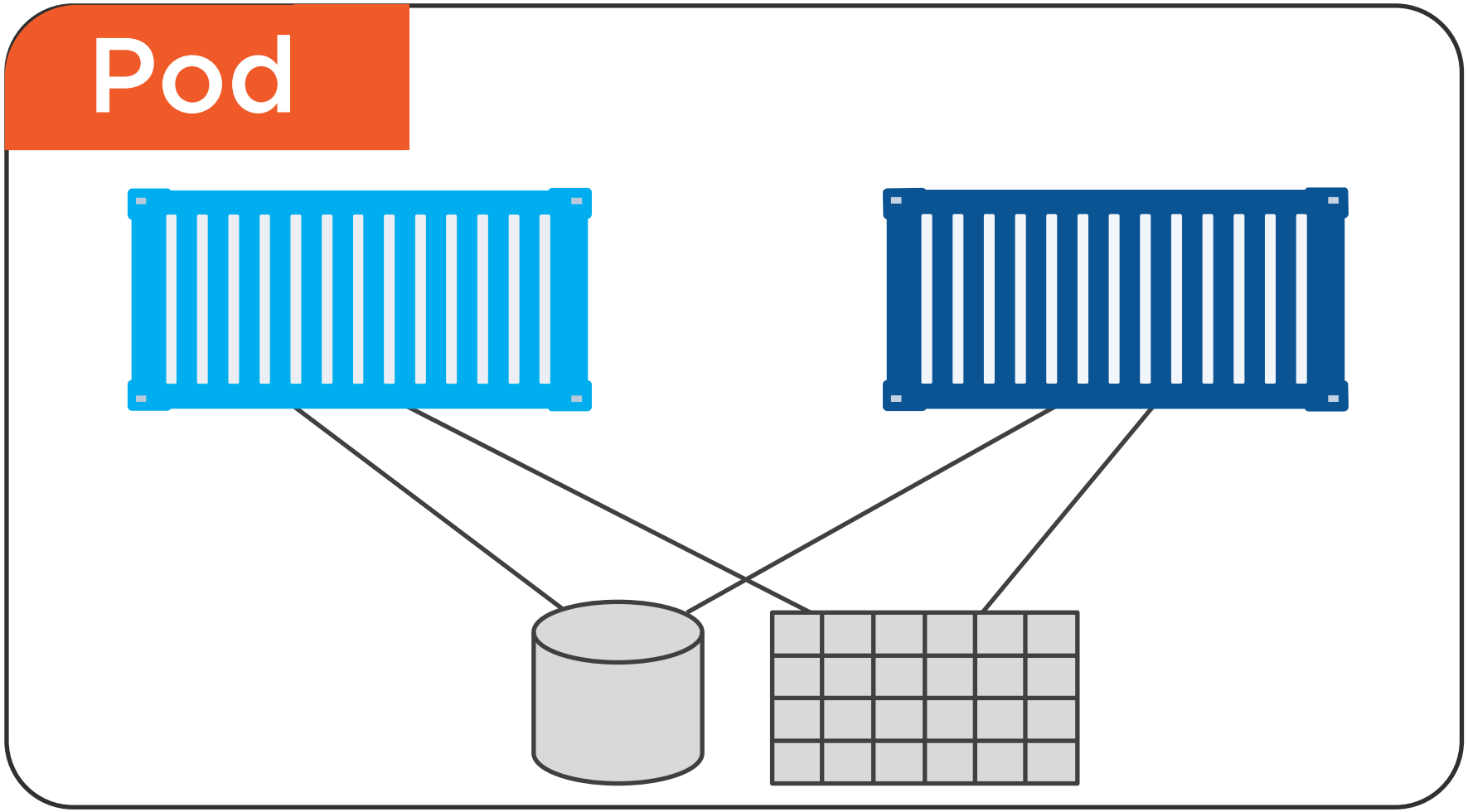
- Network stack
- Kernel namespaces
- …

n containers

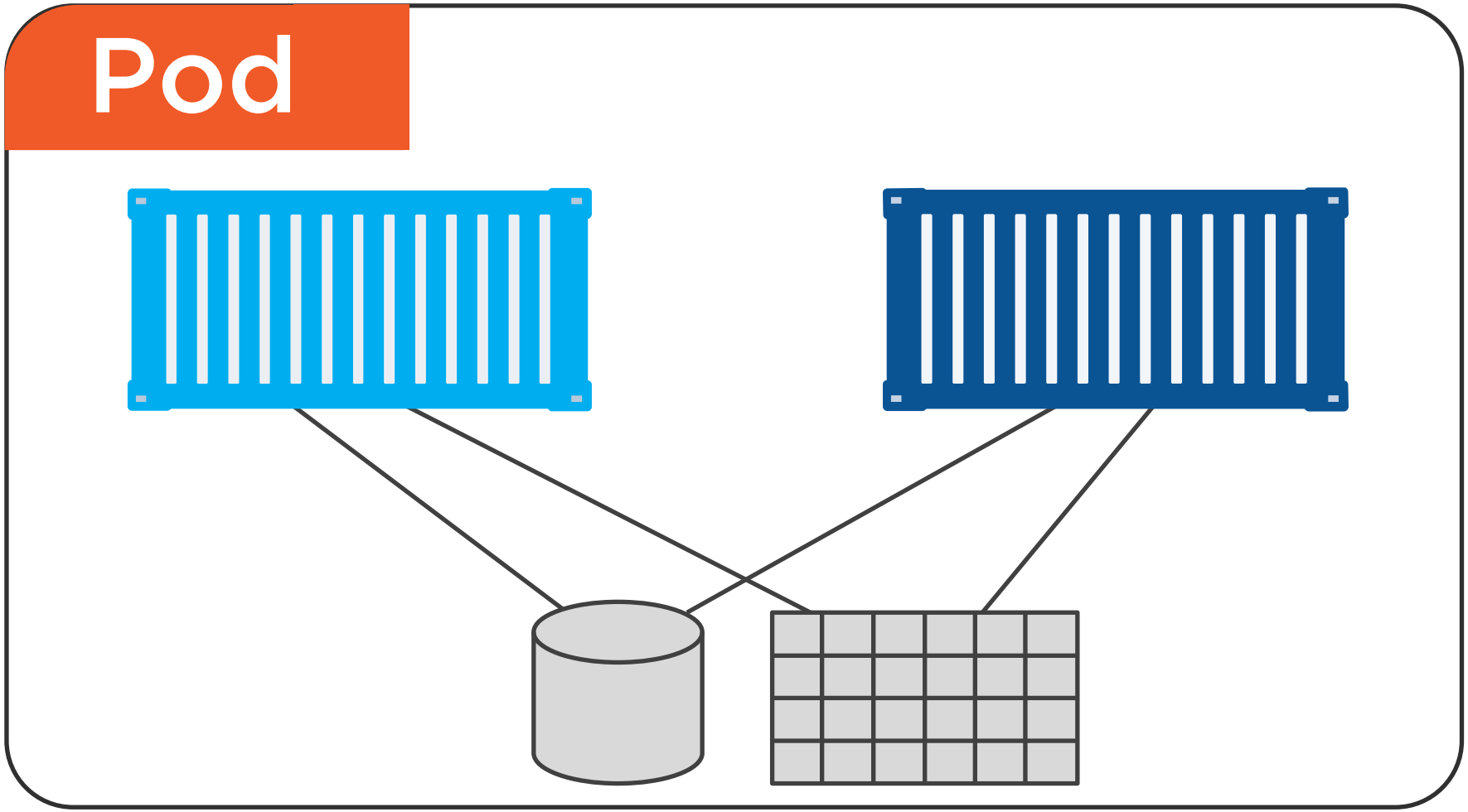
<u>All containers in pod share the pod environment</u>



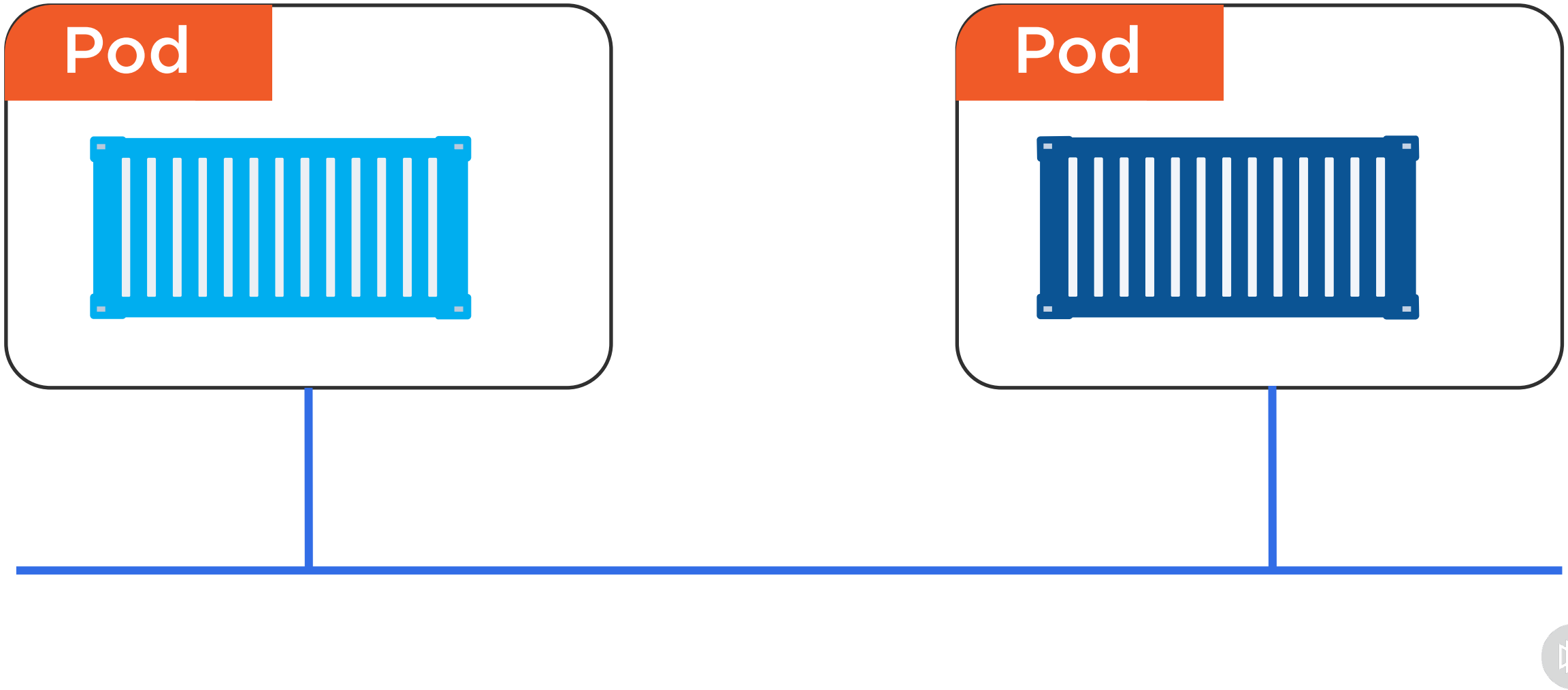
Tight Coupling



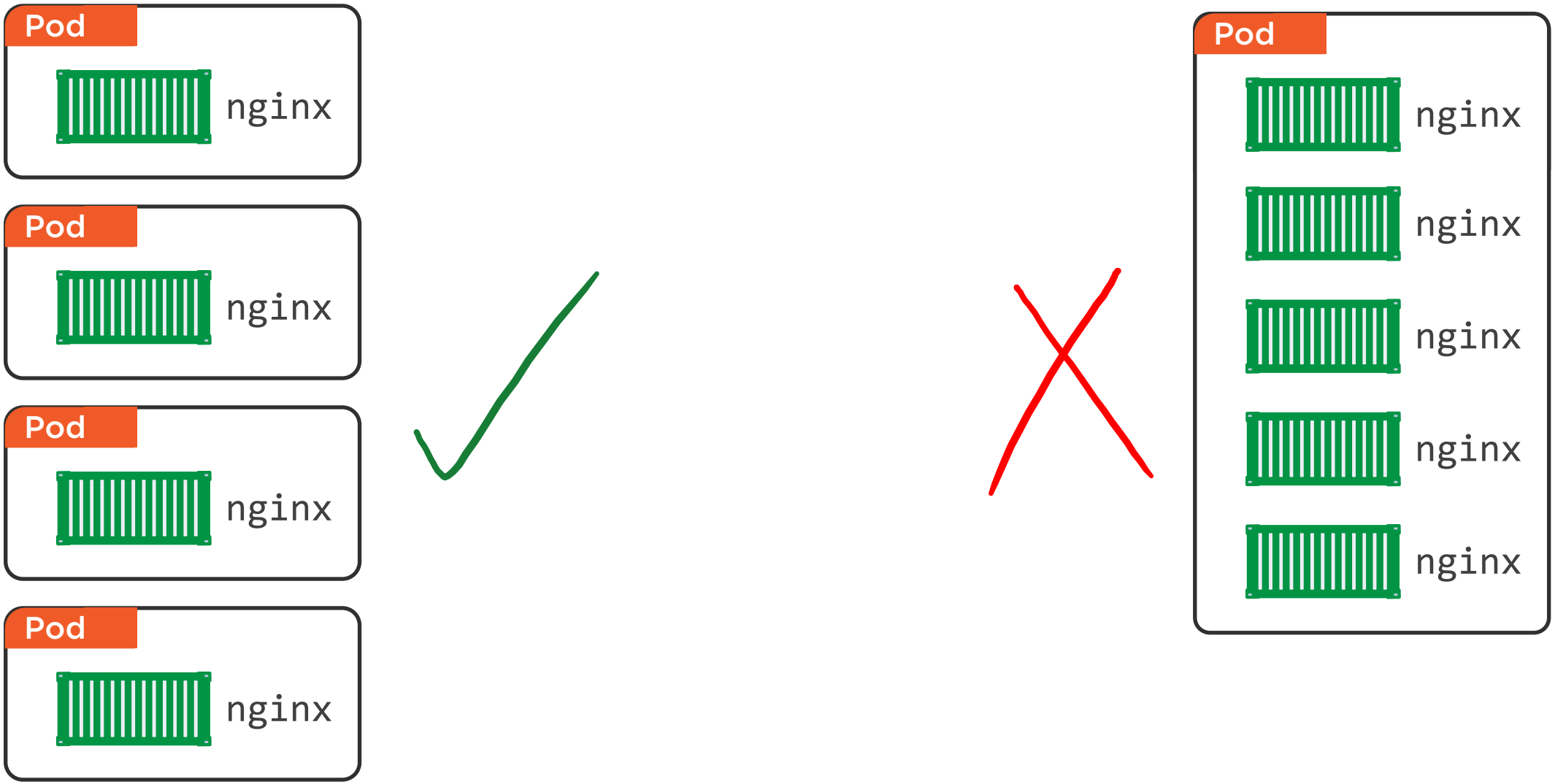
Loose Coupling



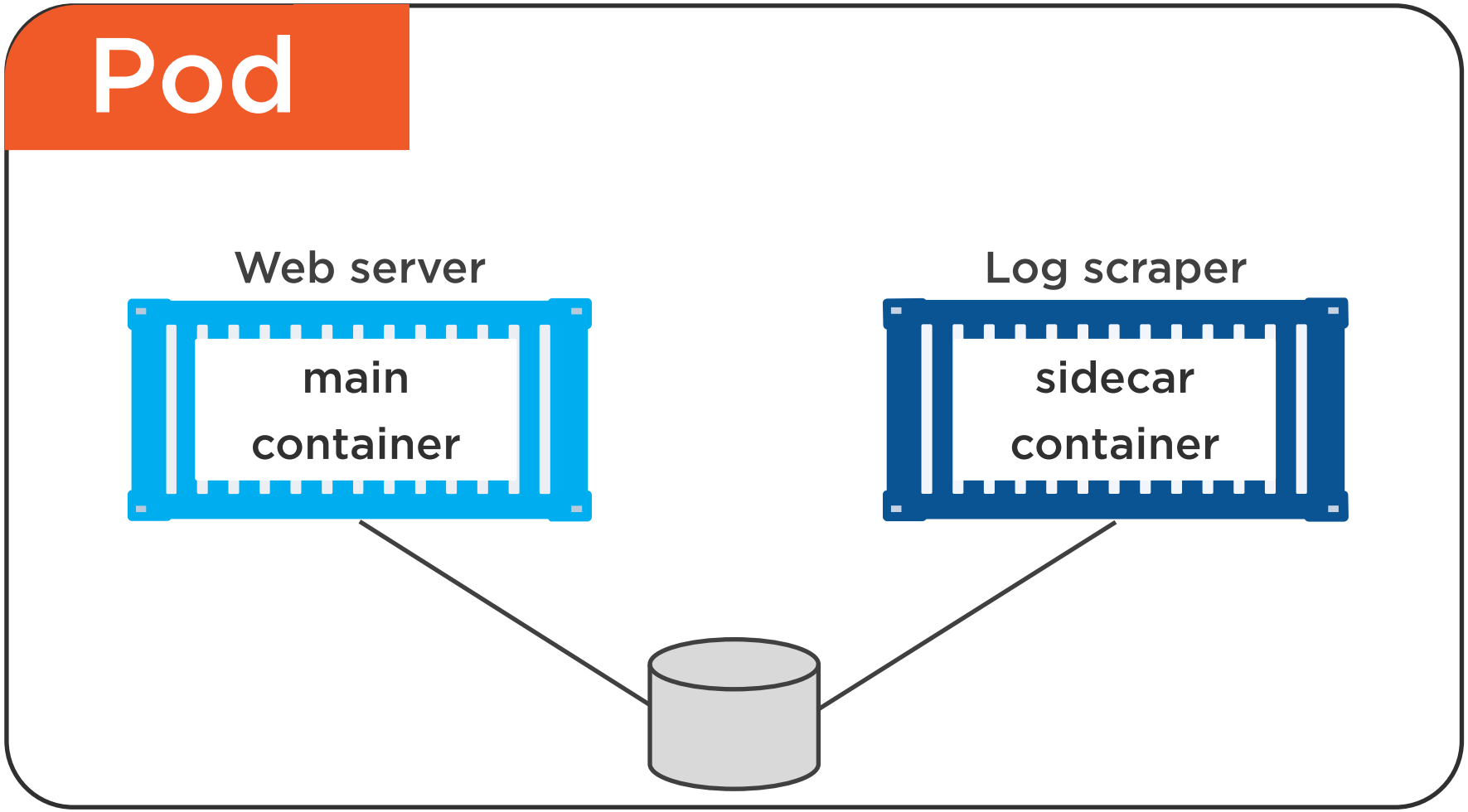
Loose Coupling



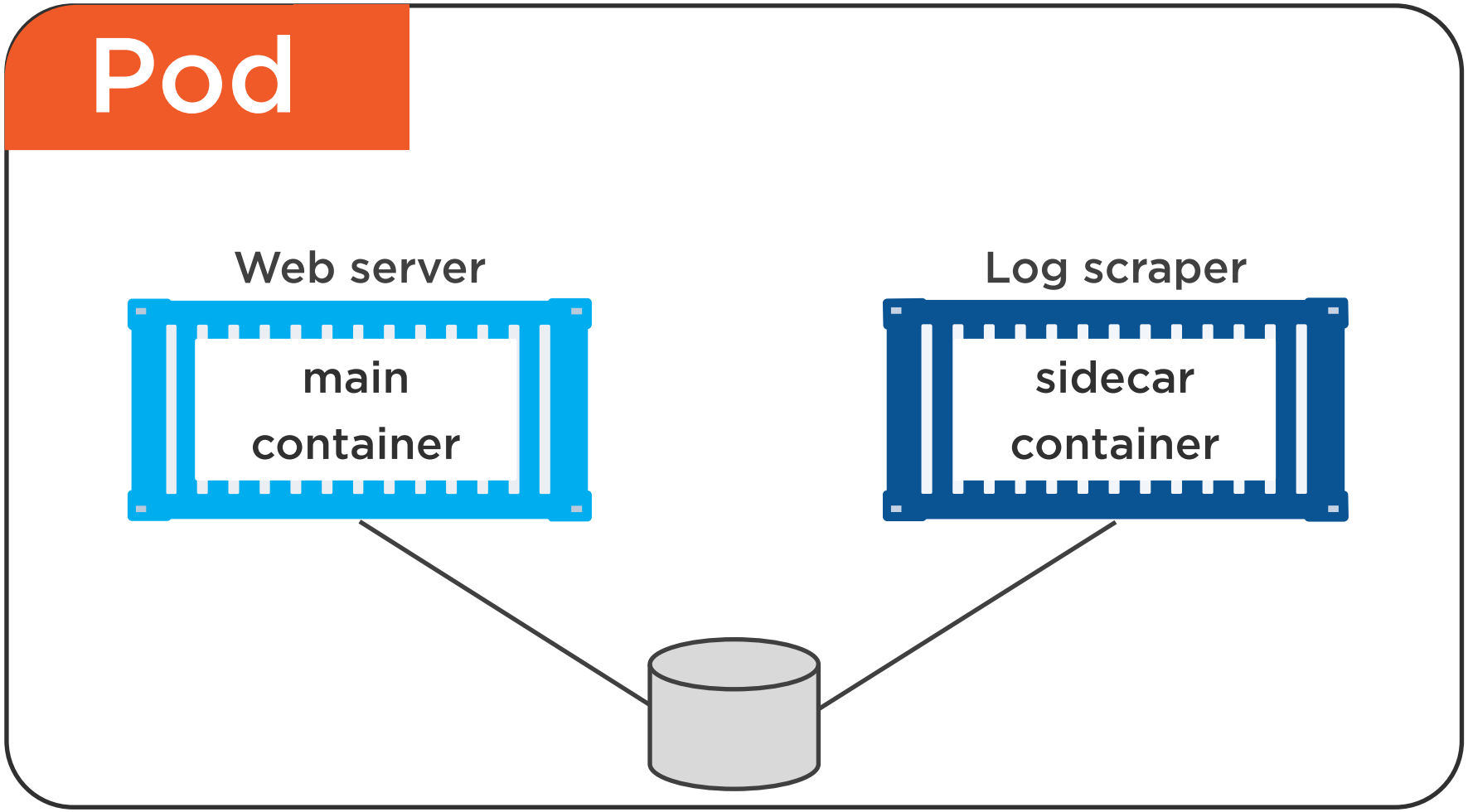
Pods and Scaling



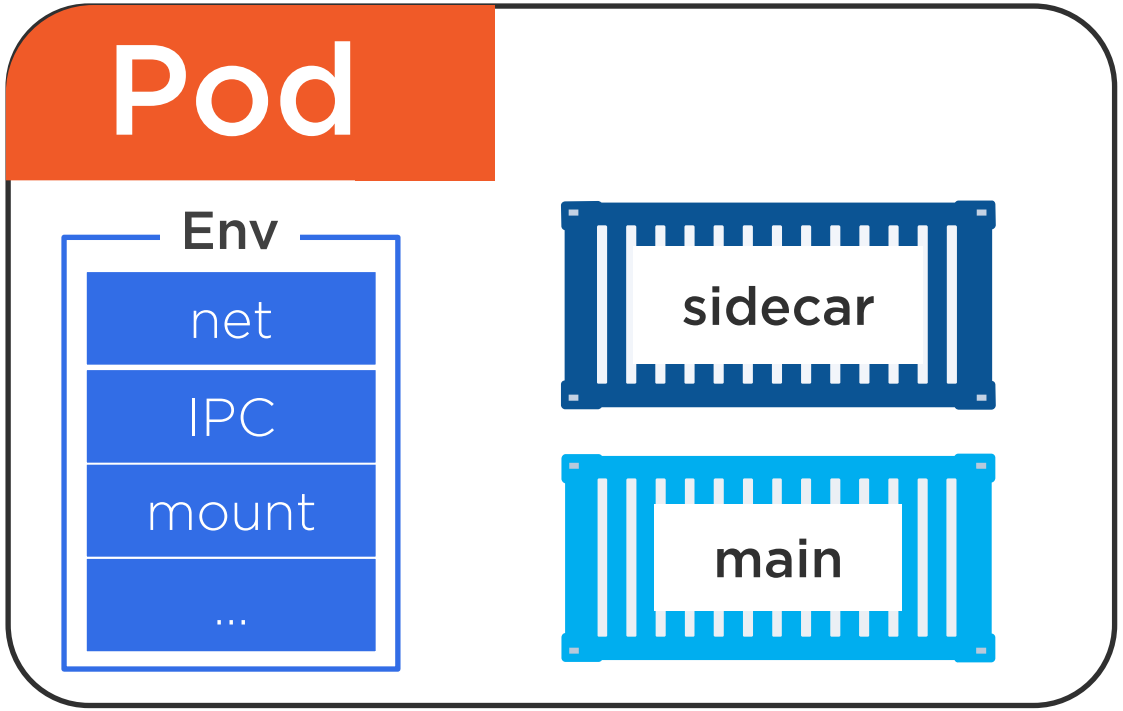
Multi-container Pods



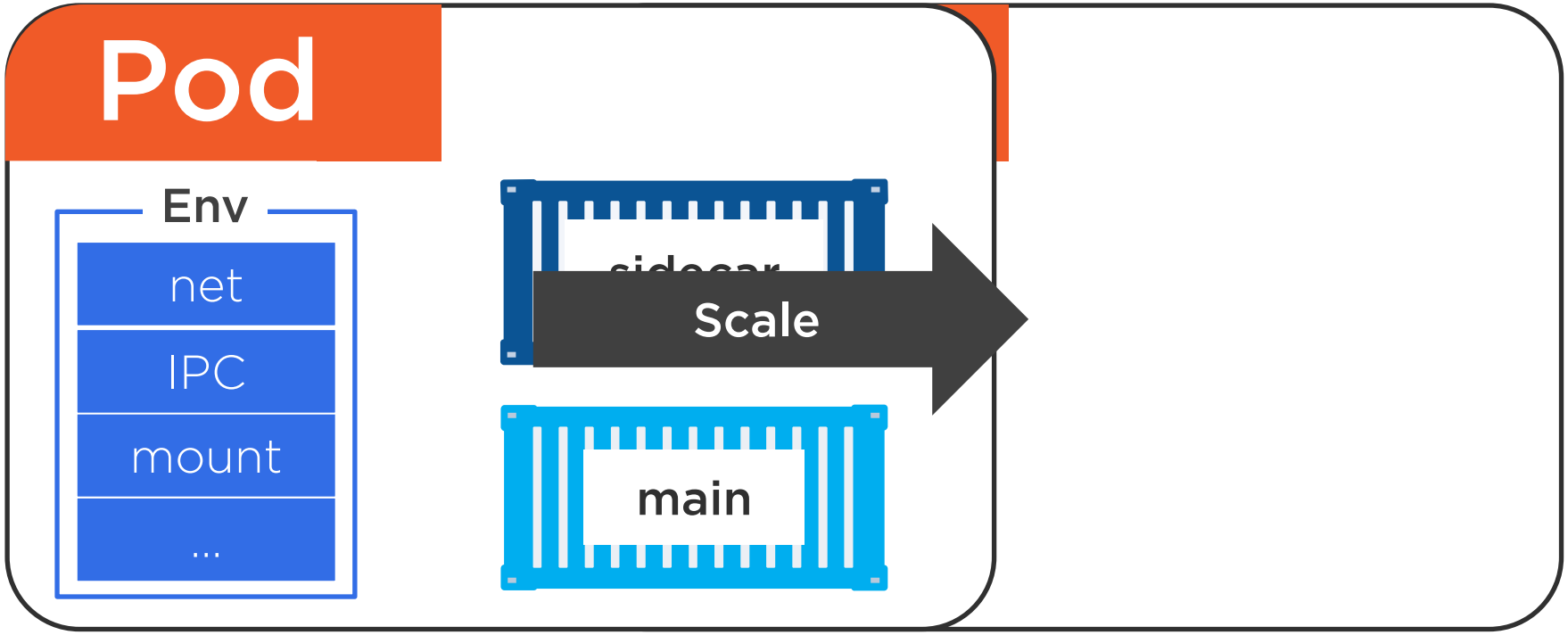
Multi-container Pods



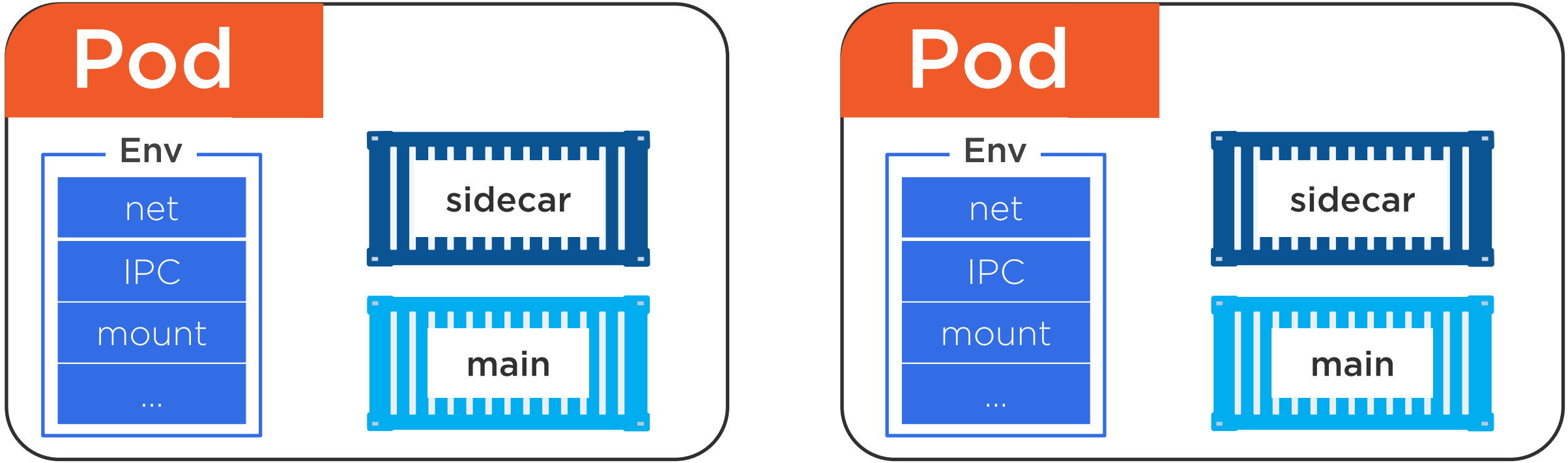
Pods are Atomic



Pods are Atomic



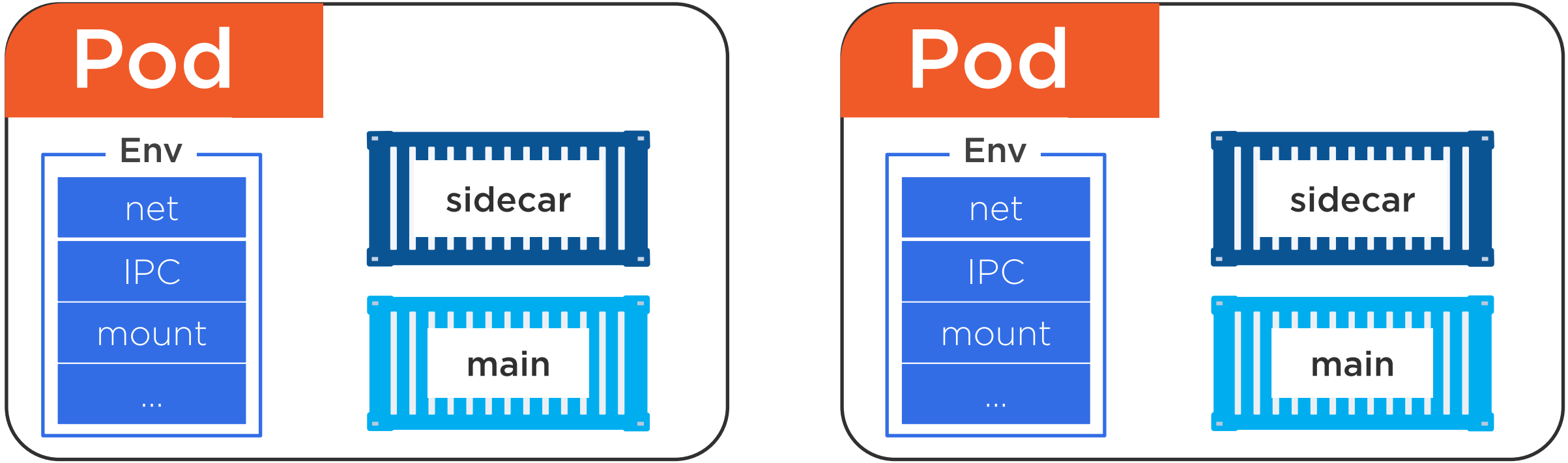
Pods are Atomic



#1 Status:ready

#2 Status:pending

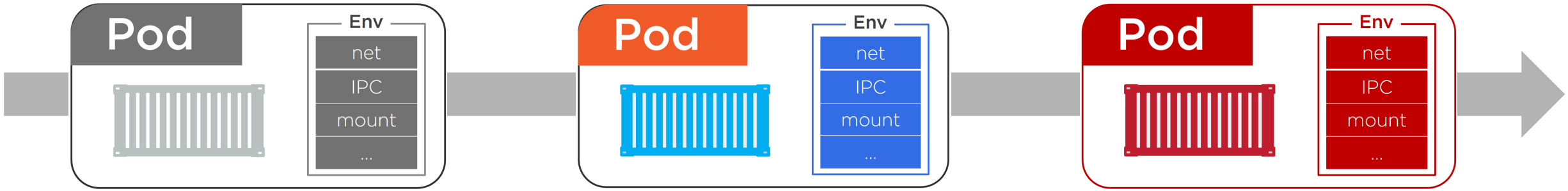
Pods are Atomic



#1 Status:ready

#2 Status:pending

Pod Lifecycle



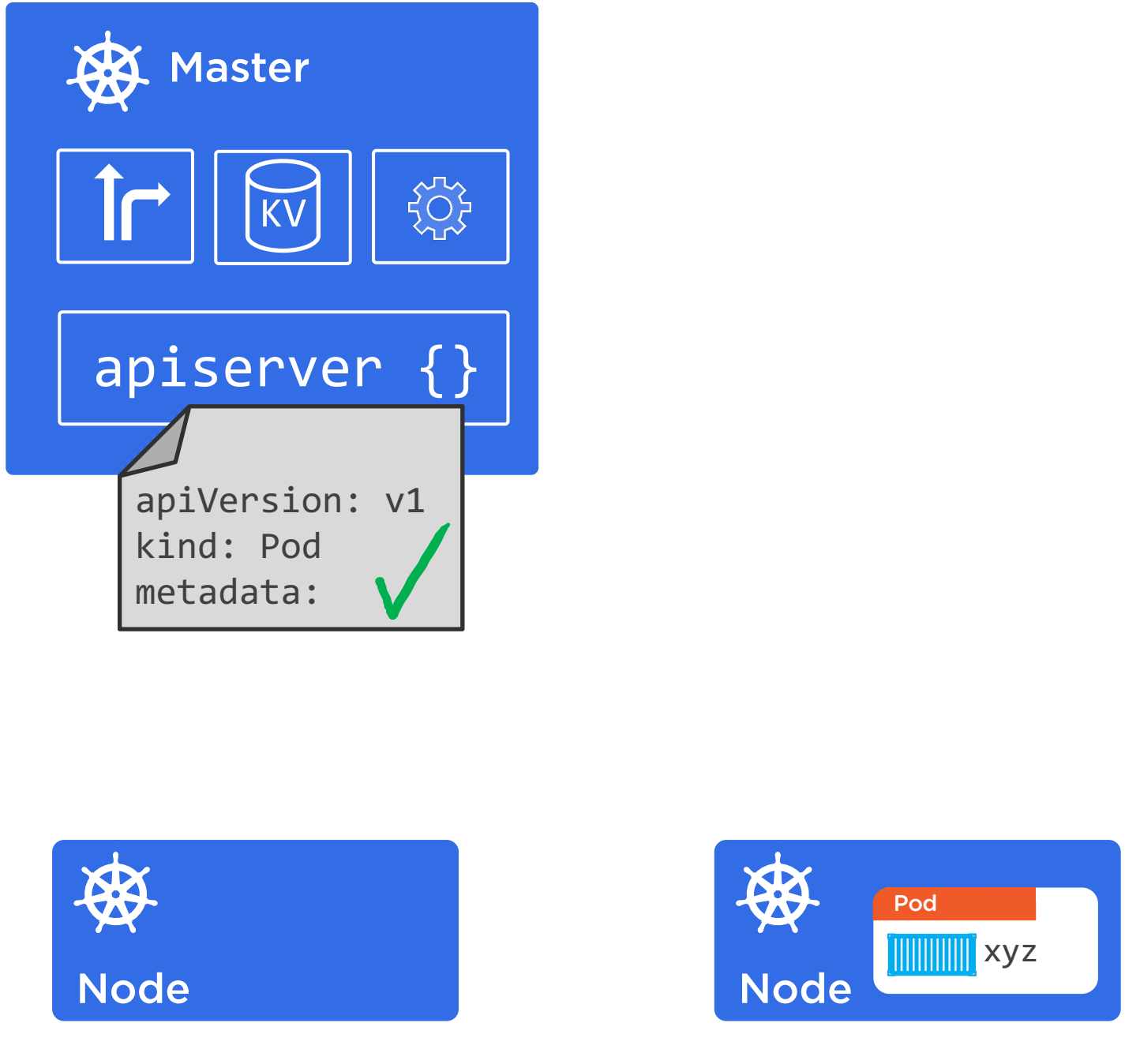
Phase: pending

Phase: running

Phase: succeeded/failed

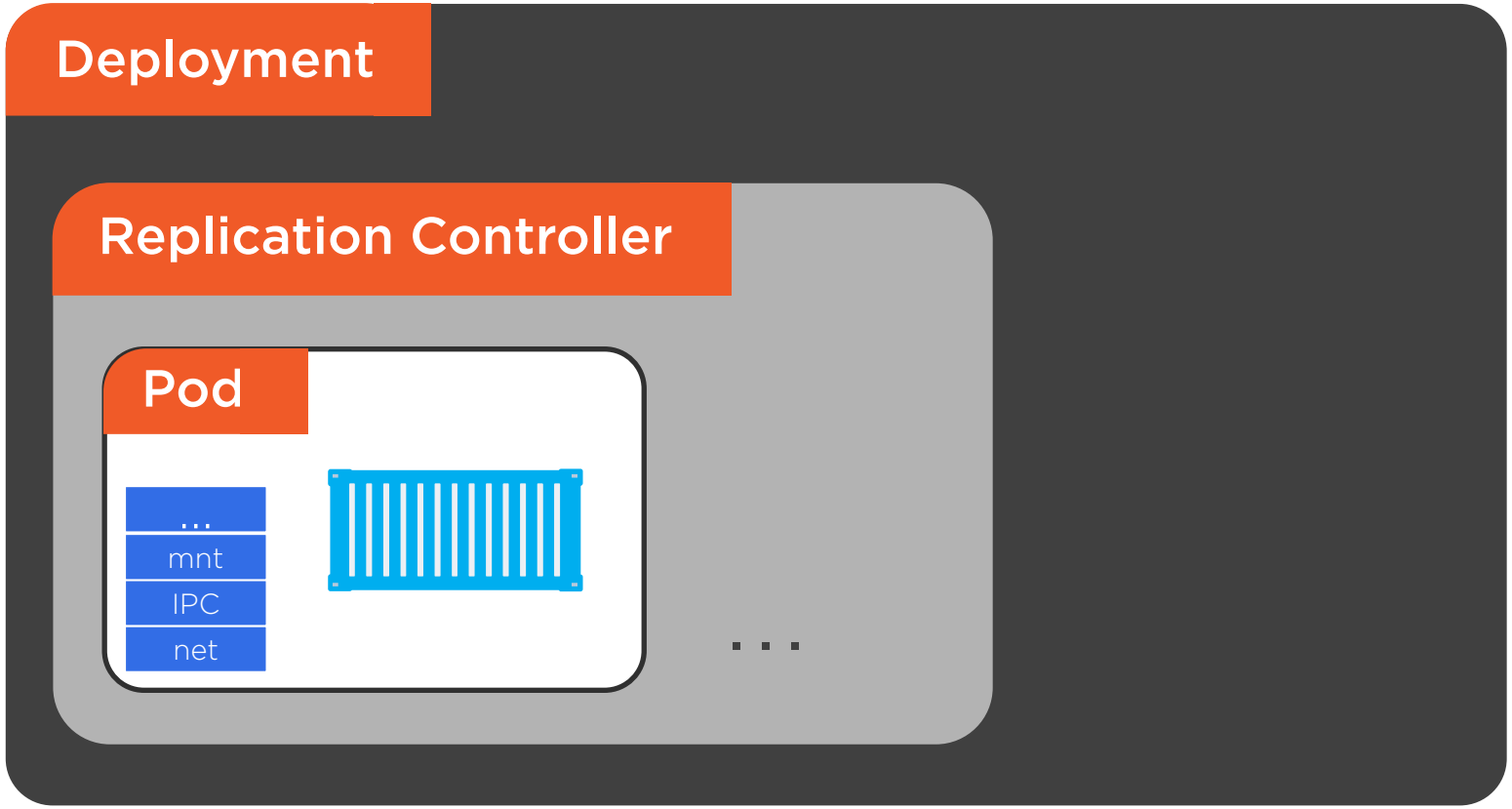
Deploying Pods

Usually via higher level objects

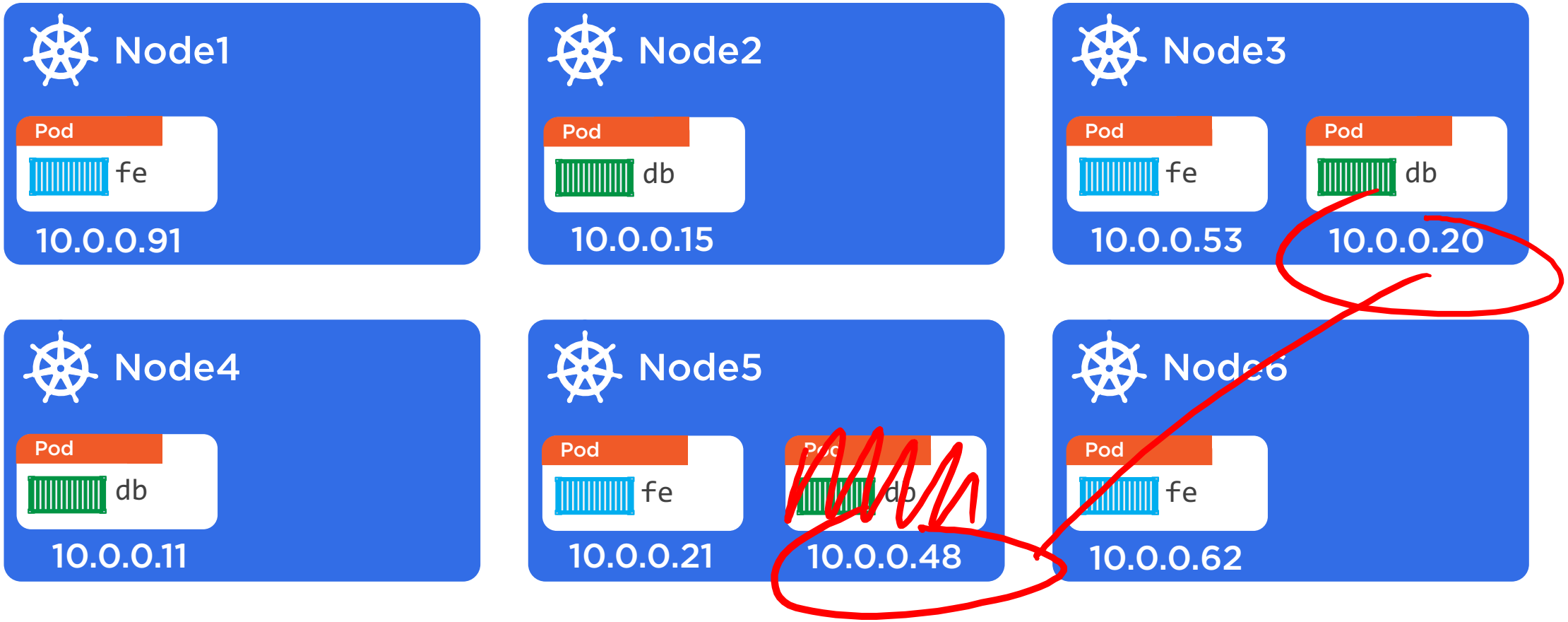

Master
KV
apiserver {}
apiVersion: v1
kind: Pod
metadata:
Node
Node
Pod
xyz
Node

Deploying Pods

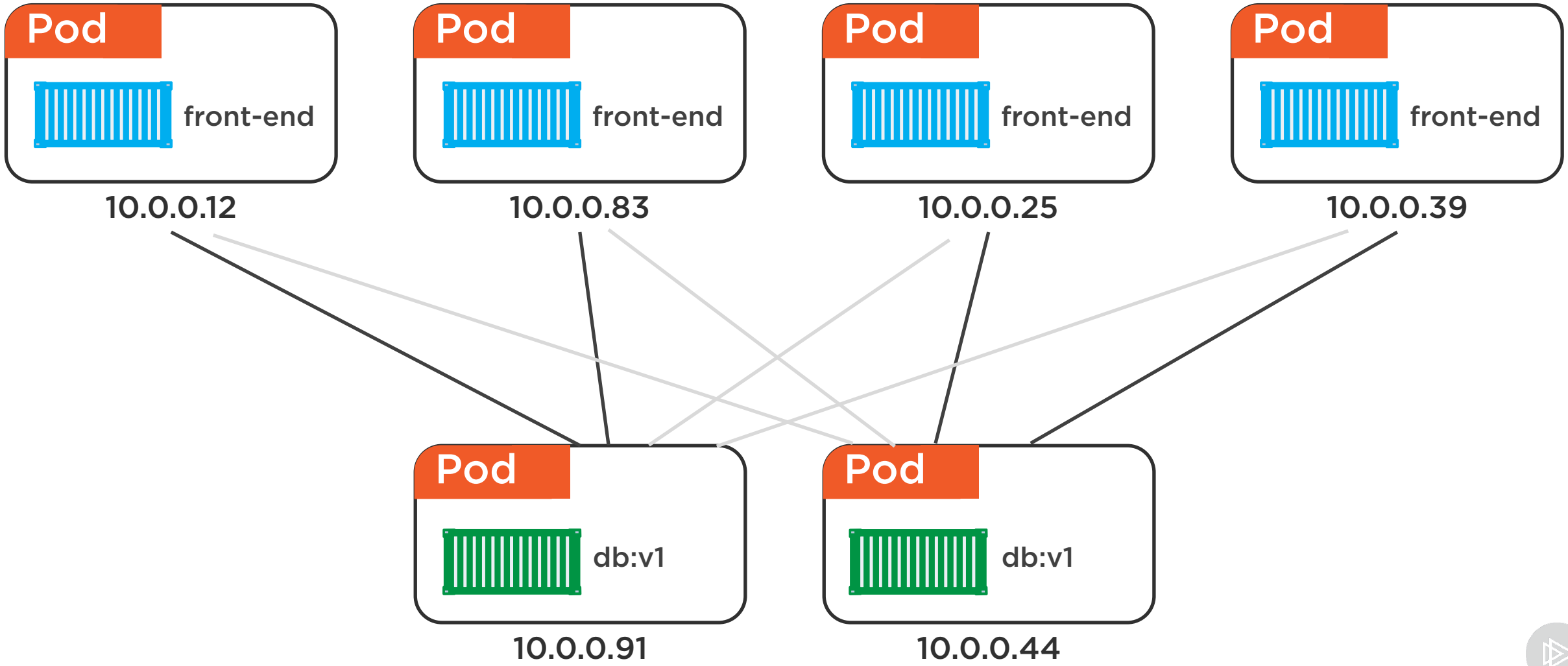
Usually via higher level objects

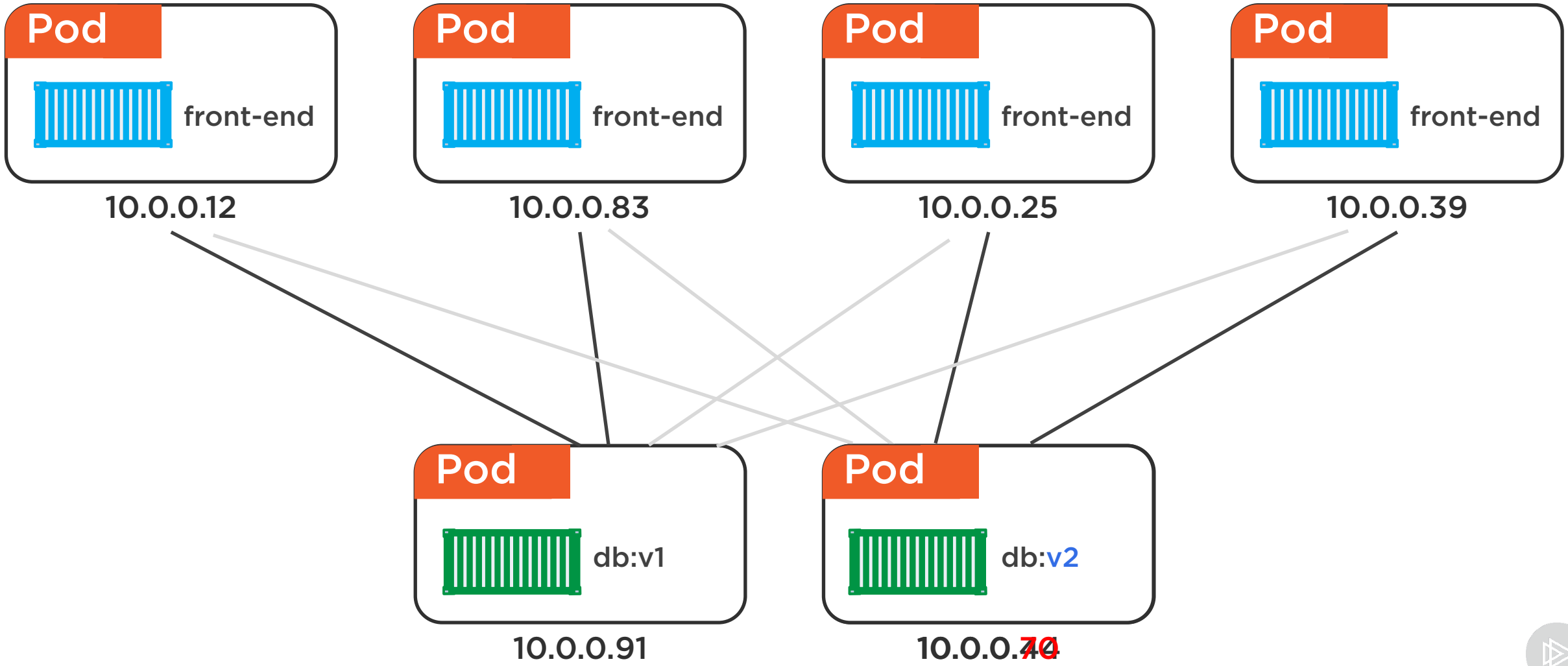


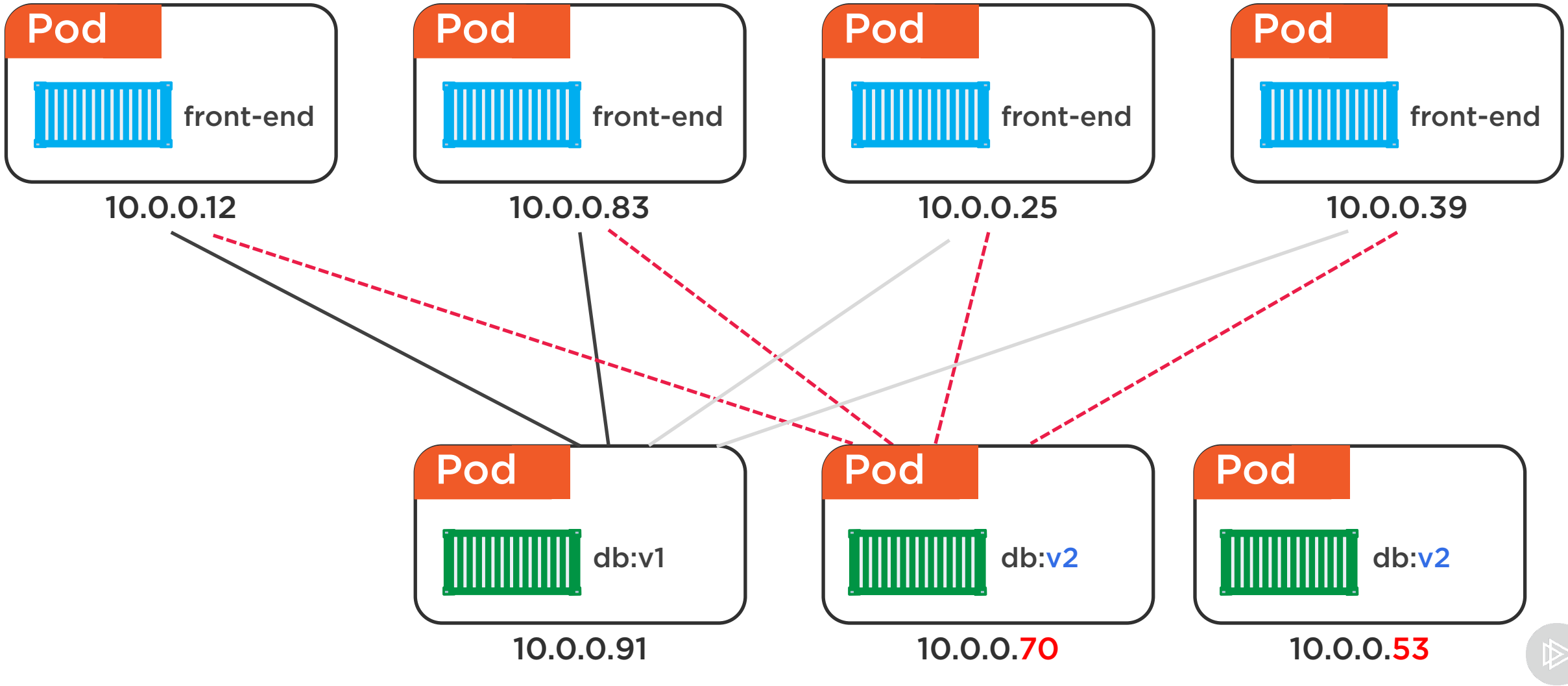
Services

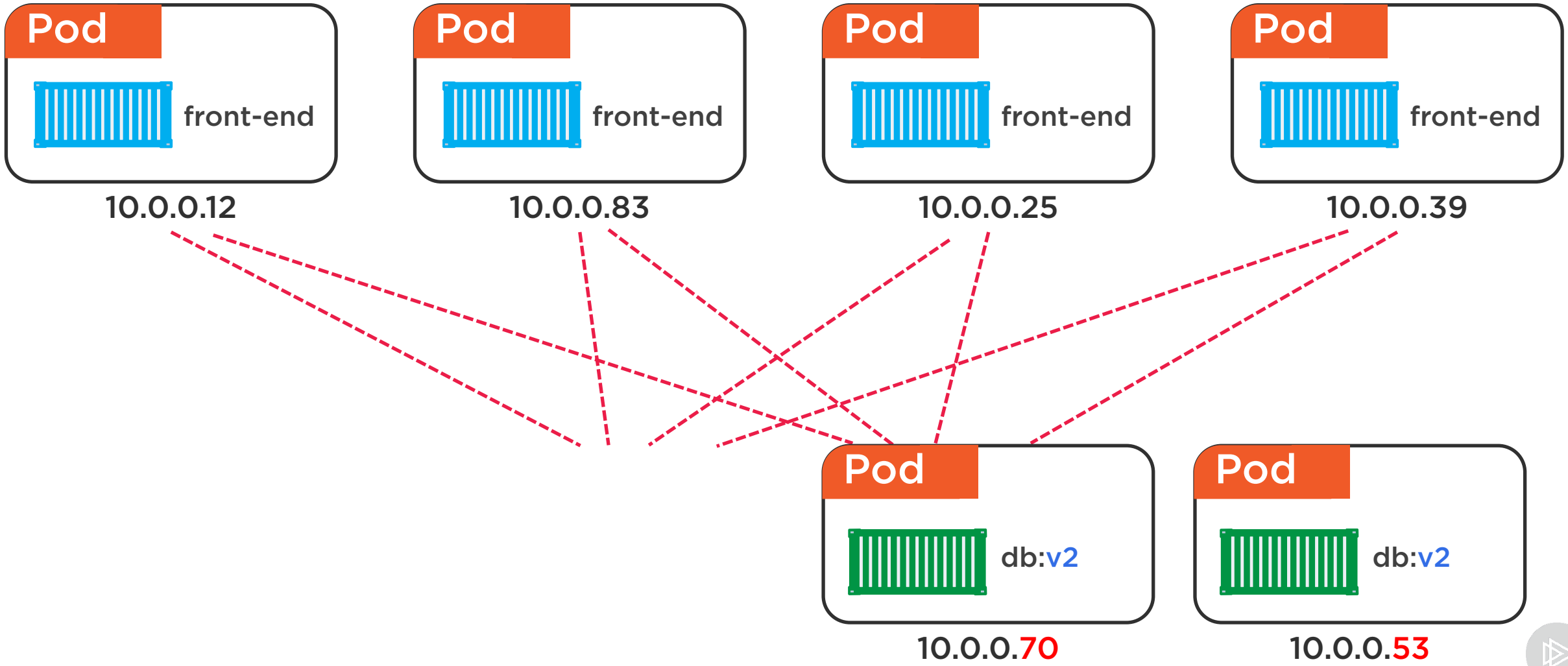


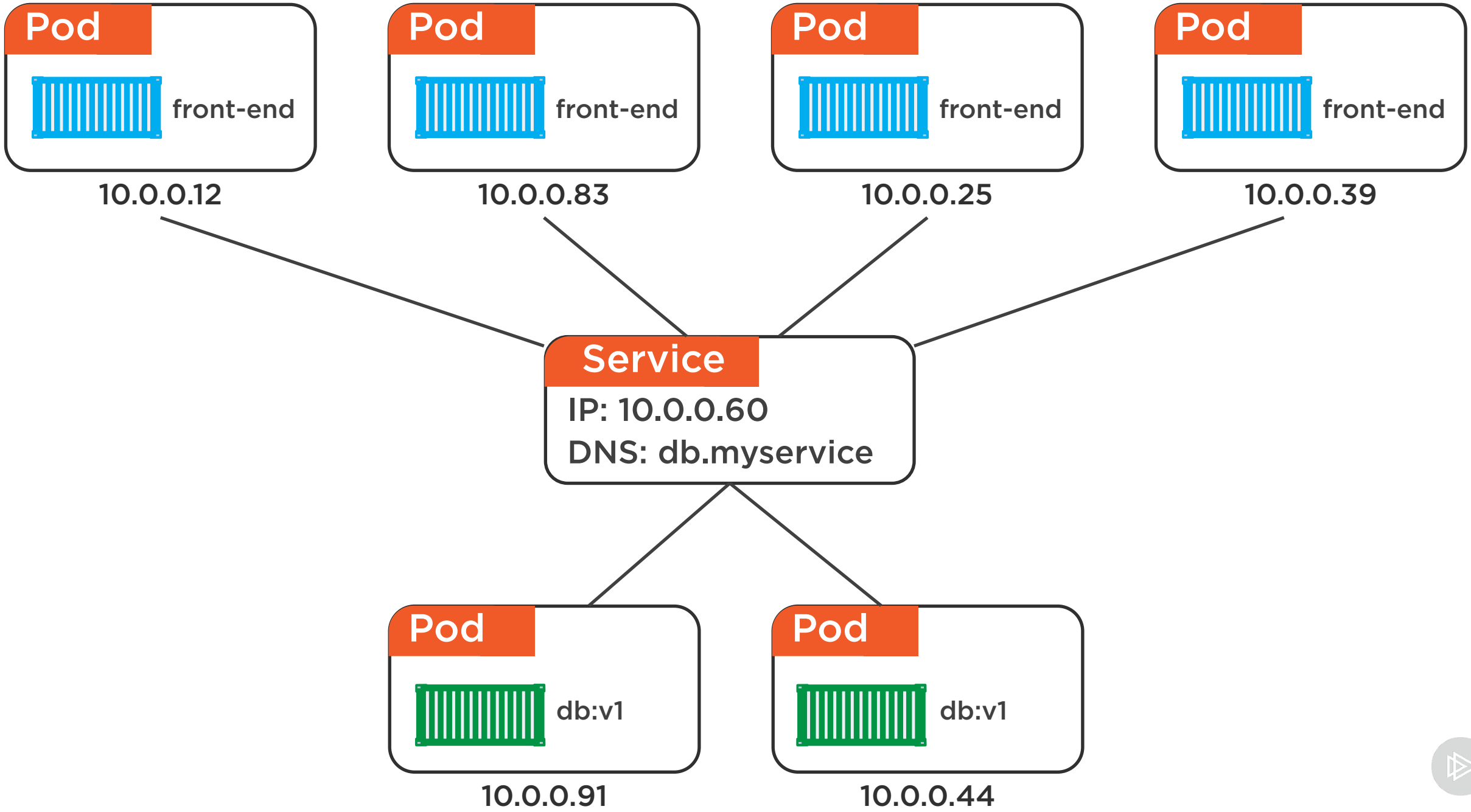
Every **new** pod gets a new IP = IP churn!

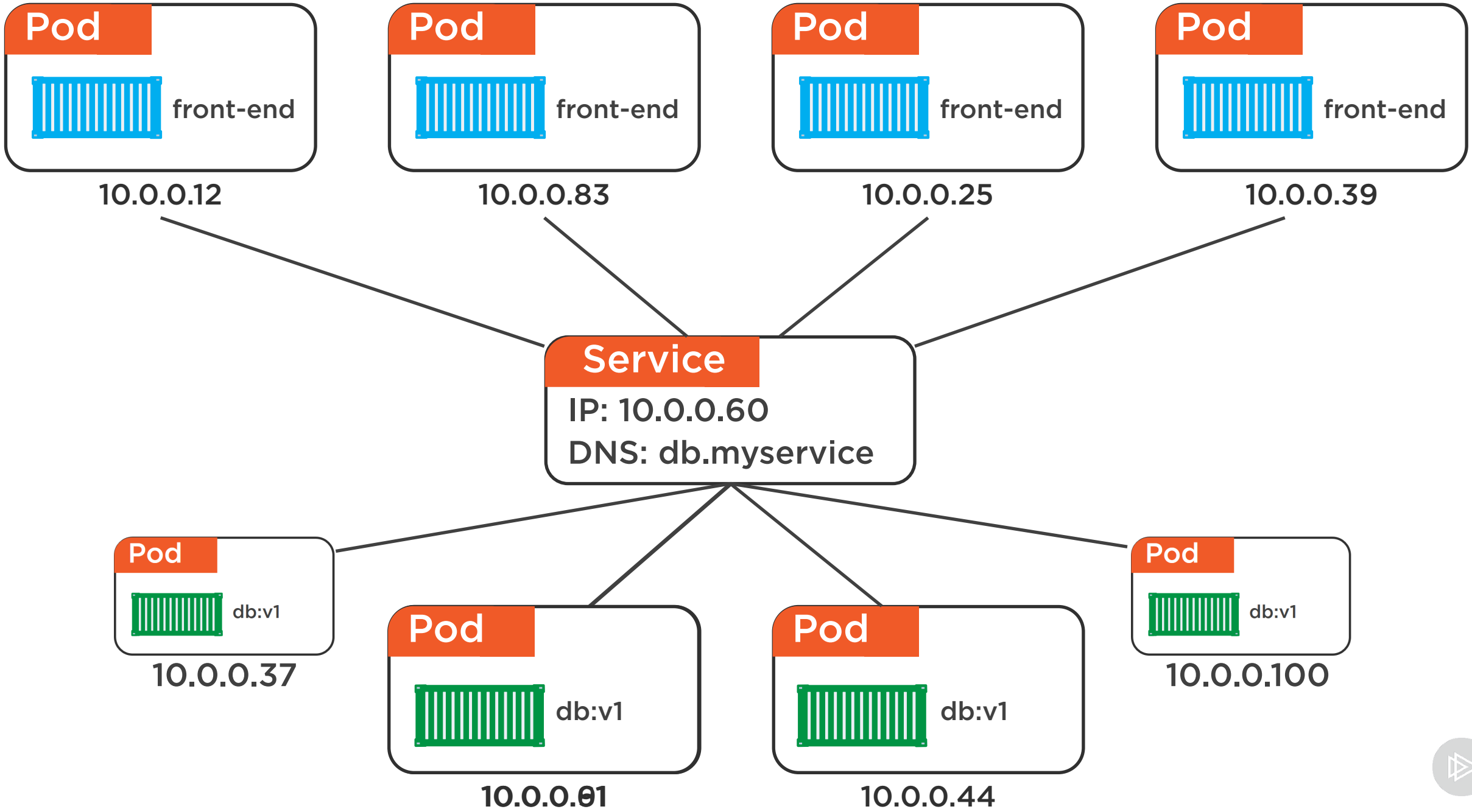

Pod
front-end
10.0.0.12
Pod
front-end
10.0.0.83
Pod
front-end
10.0.0.25
Pod
front-end
10.0.0.39
Pod
db:v1
10.0.0.91
Pod
db:v1
10.0.0.44

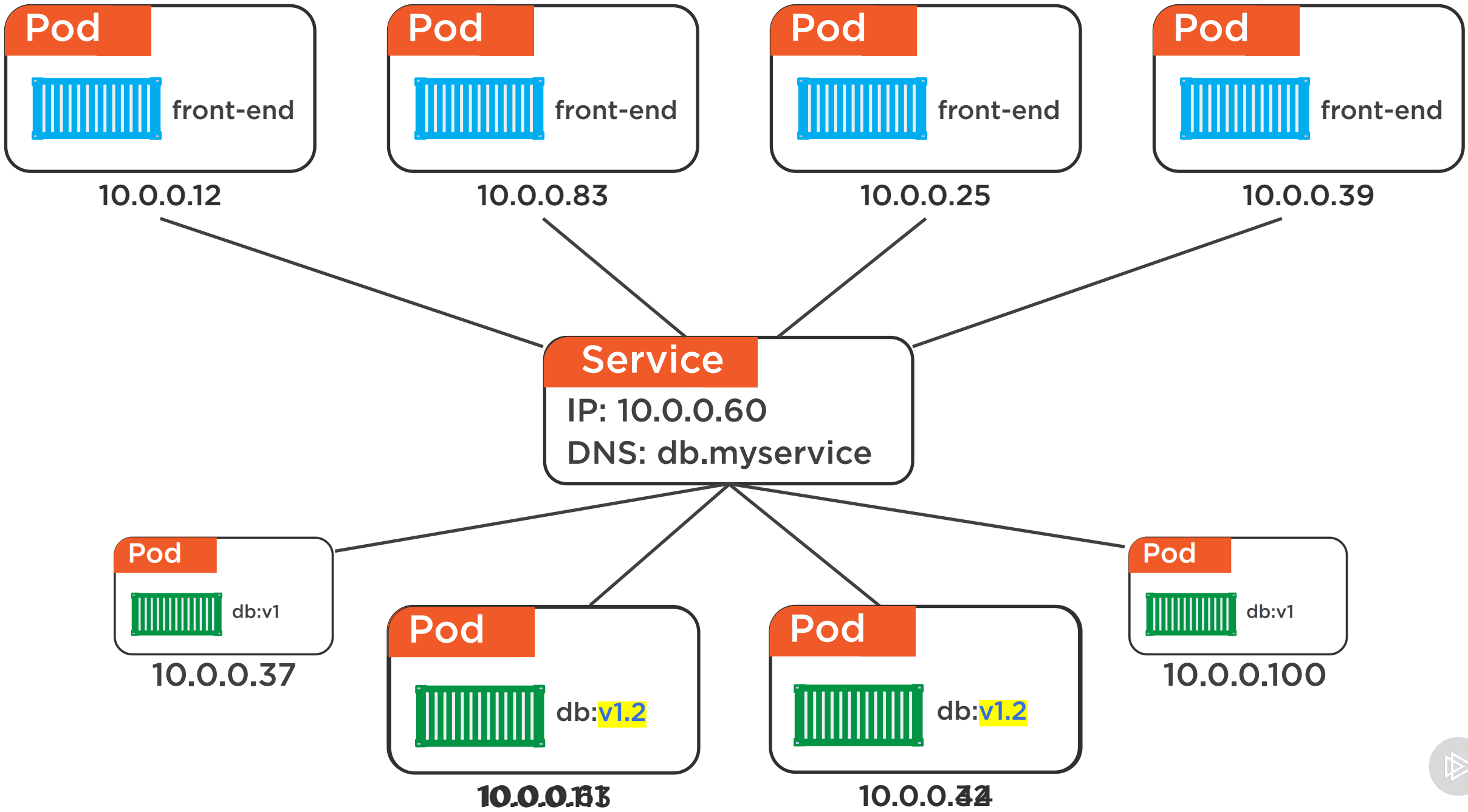

Pod
front-end
10.0.0.12
Pod
front-end
10.0.0.83
Pod
front-end
10.0.0.25
Pod
front-end
10.0.0.39
Pod
db:v1
10.0.0.91
Pod
db:v2
10.0.0.70

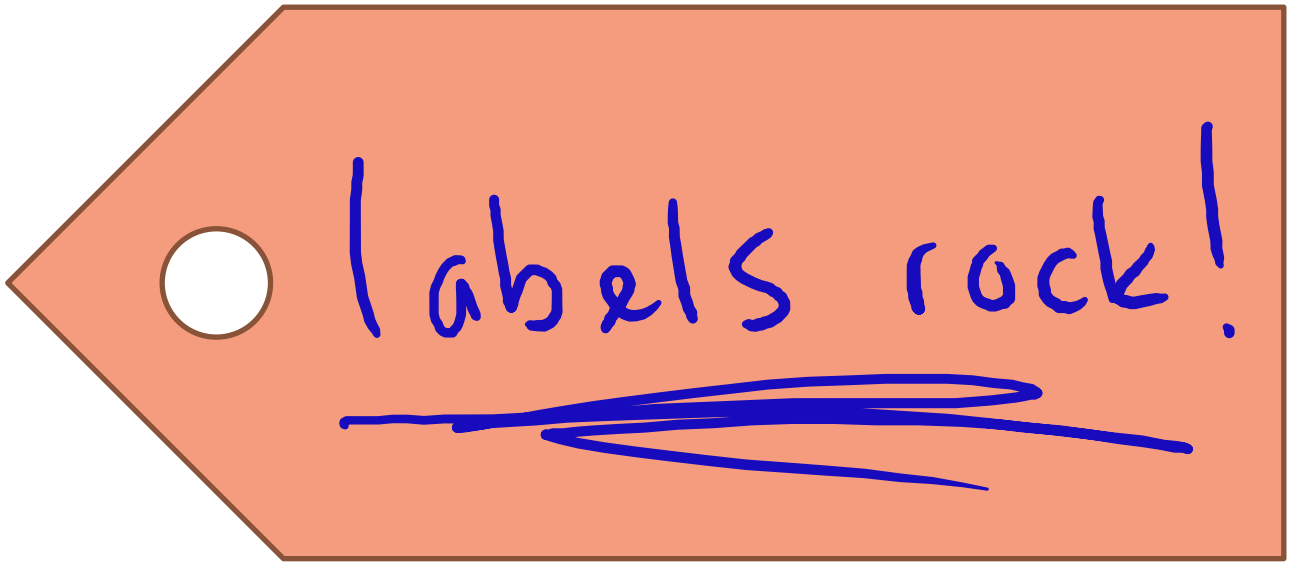

Pod
front-end
10.0.0.12
Pod
front-end
10.0.0.83
Pod
front-end
10.0.0.25
Pod
front-end
10.0.0.39
Pod
db:v1
10.0.0.91
Pod
db:v2
10.0.0.70
Pod
db:v2
10.0.0.53

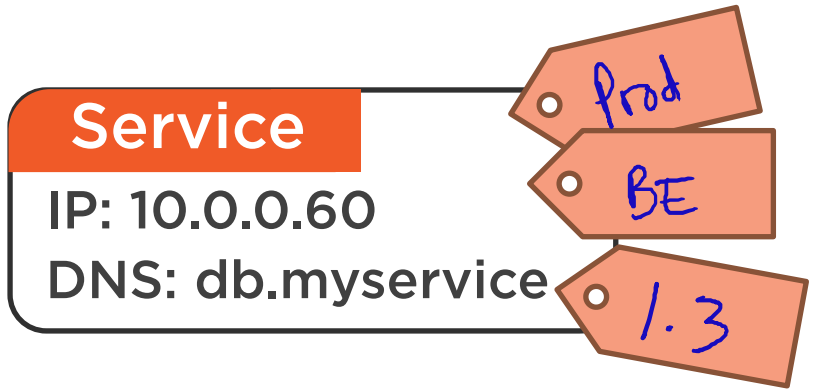

Pod
front-end
10.0.0.12
Pod
front-end
10.0.0.83
Pod
front-end
10.0.0.25
Pod
front-end
10.0.0.39
Pod
db:v2
10.0.0.70
Pod
db:v2
10.0.0.53

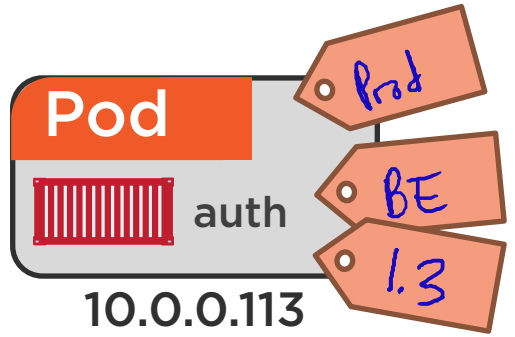

Pod
front-end
10.0.0.12
Pod
front-end
10.0.0.83
Pod
front-end
10.0.0.25
Pod
front-end
10.0.0.39
Service
IP: 10.0.0.60
DNS: db.myservice
Pod
db:v1
10.0.0.91
Pod
db:v1
10.0.0.44

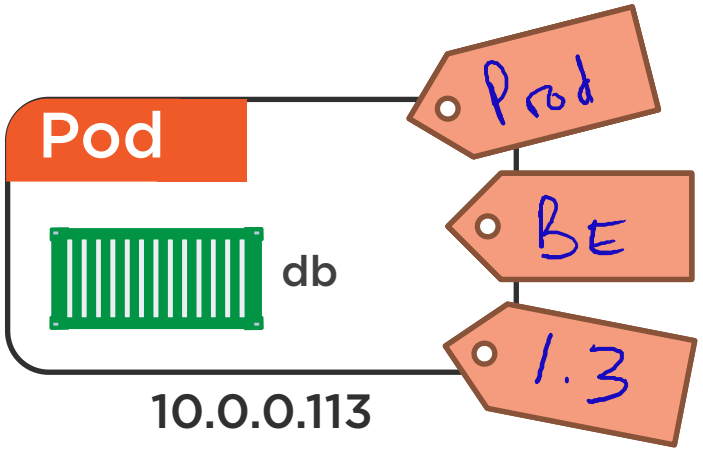

Pod
front-end
10.0.0.12
Pod
front-end
10.0.0.83
Pod
front-end
10.0.0.25
Pod
front-end
10.0.0.39
Service
IP: 10.0.0.60
DNS: db.myservice
Pod
db:v1
10.0.0.37
Pod
db:v1
10.0.0.81
Pod
db:v1
10.0.0.44
Pod
db:v1
10.0.0.100

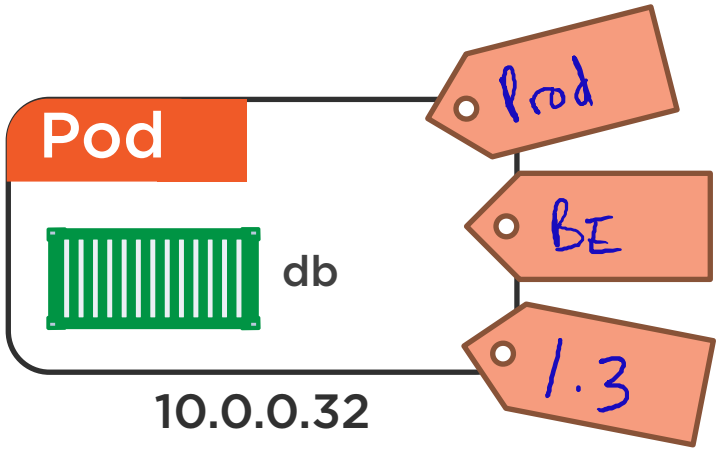

Pod
front-end
10.0.0.12
Pod
front-end
10.0.0.83
Pod
front-end
10.0.0.25
Pod
front-end
10.0.0.39
Service
IP: 10.0.0.60
DNS: db.myservice
Pod
db:v1
10.0.0.37
Pod
db:v1.2
Pod
db:v1.2
Pod
db:v1
10.0.0.100

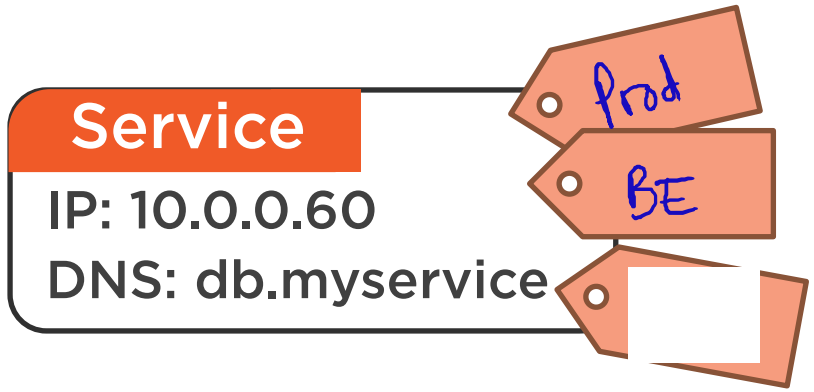

labels rock!

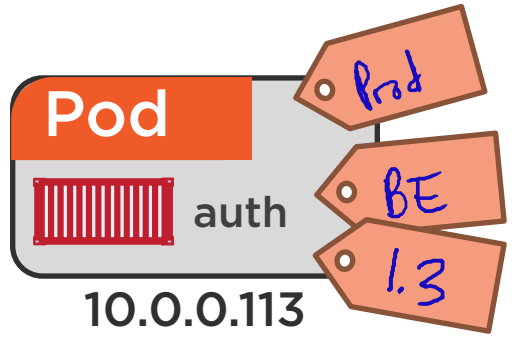

Service
IP: 10.0.0.60
DNS: db.myservice
Prod
BE
1.3

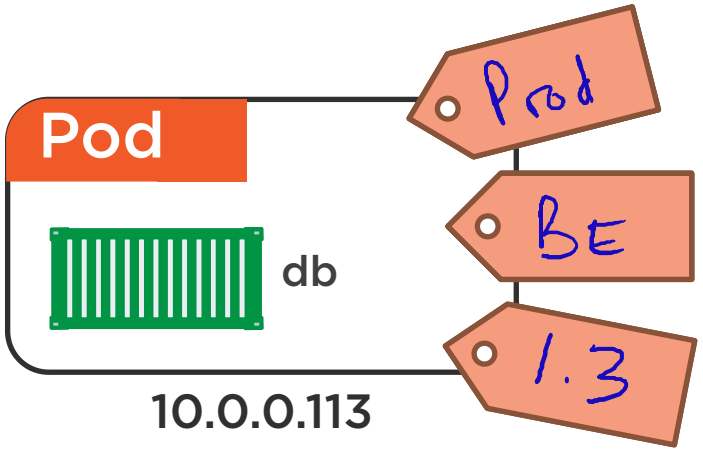

Pod
auth
10.0.0.113
Prod
BE
1.3

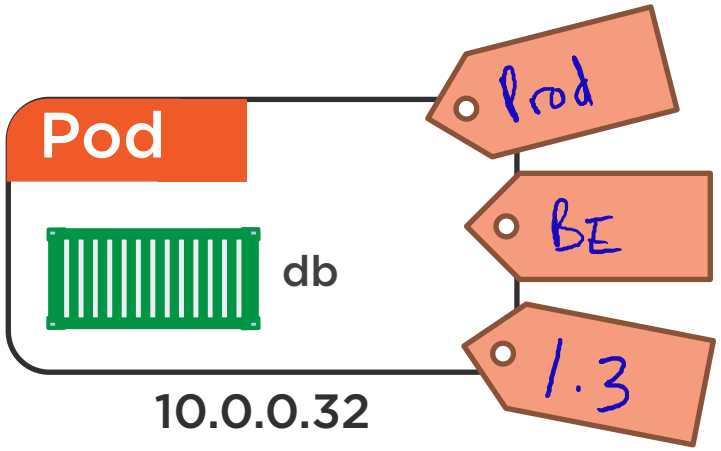

Pod
db
10.0.0.113
Prod
BE
1.3

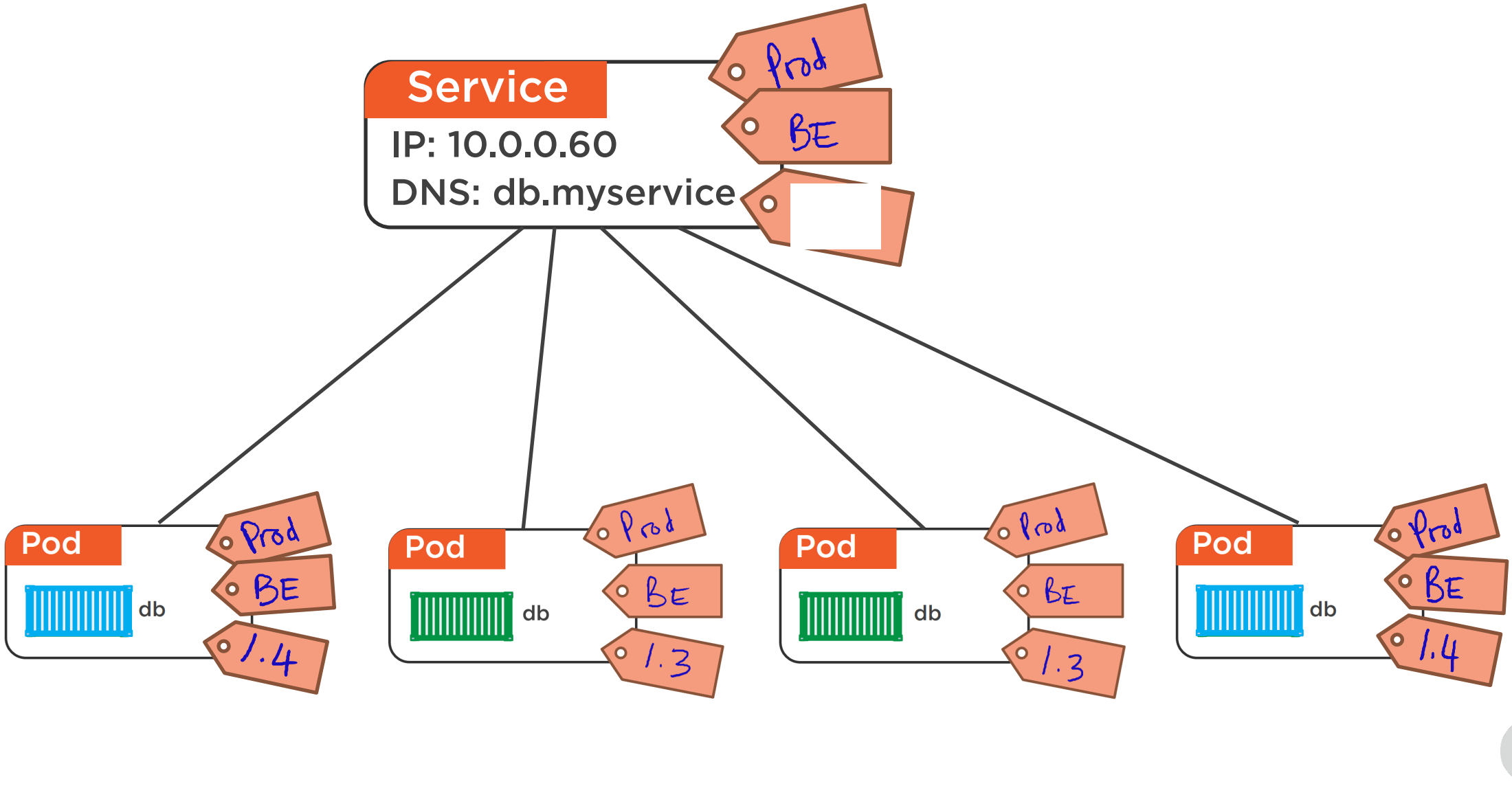

Pod
db
10.0.0.32
Prod
BE
1.3

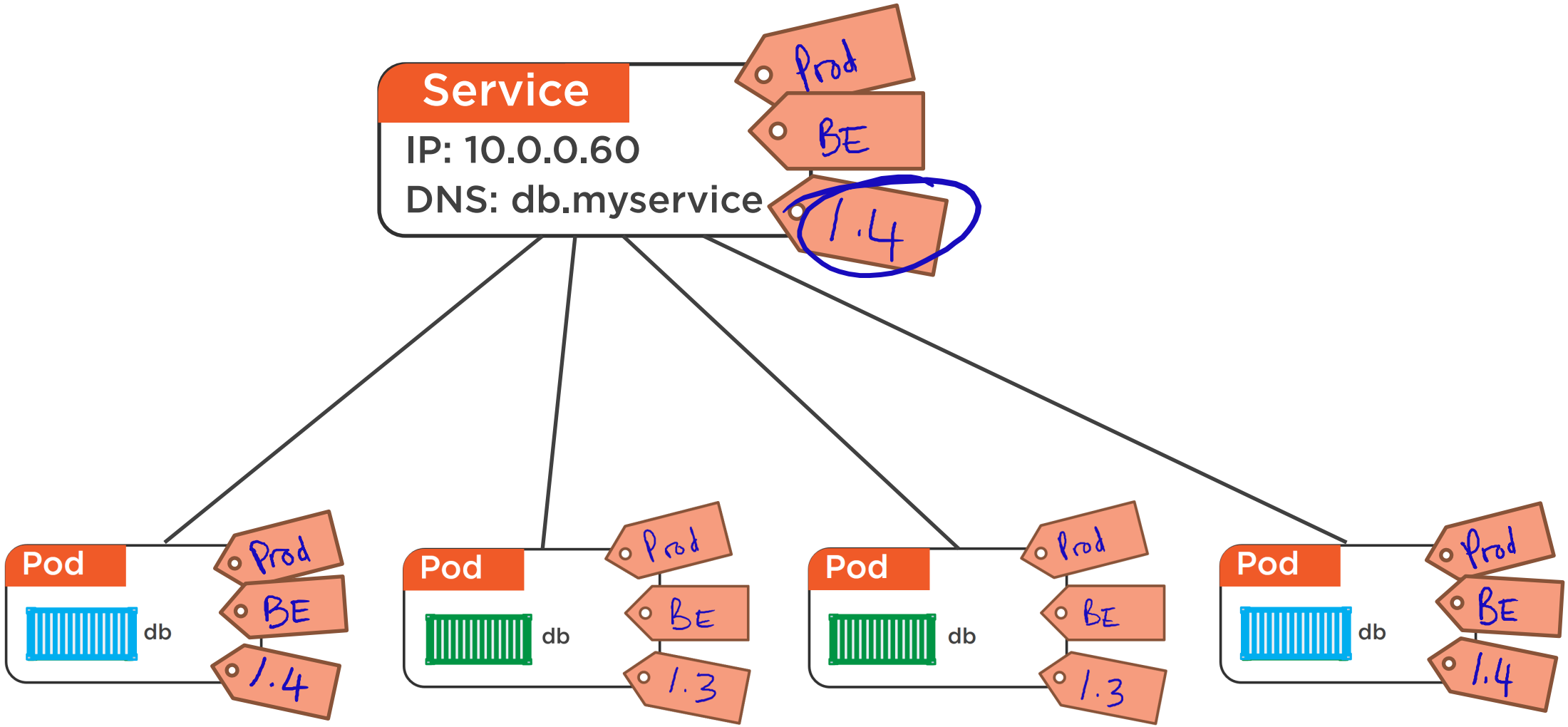

Service
IP: 10.0.0.60
DNS: db.myservice
Prod
BE


Pod
auth
10.0.0.113
Prod
BE
1.3

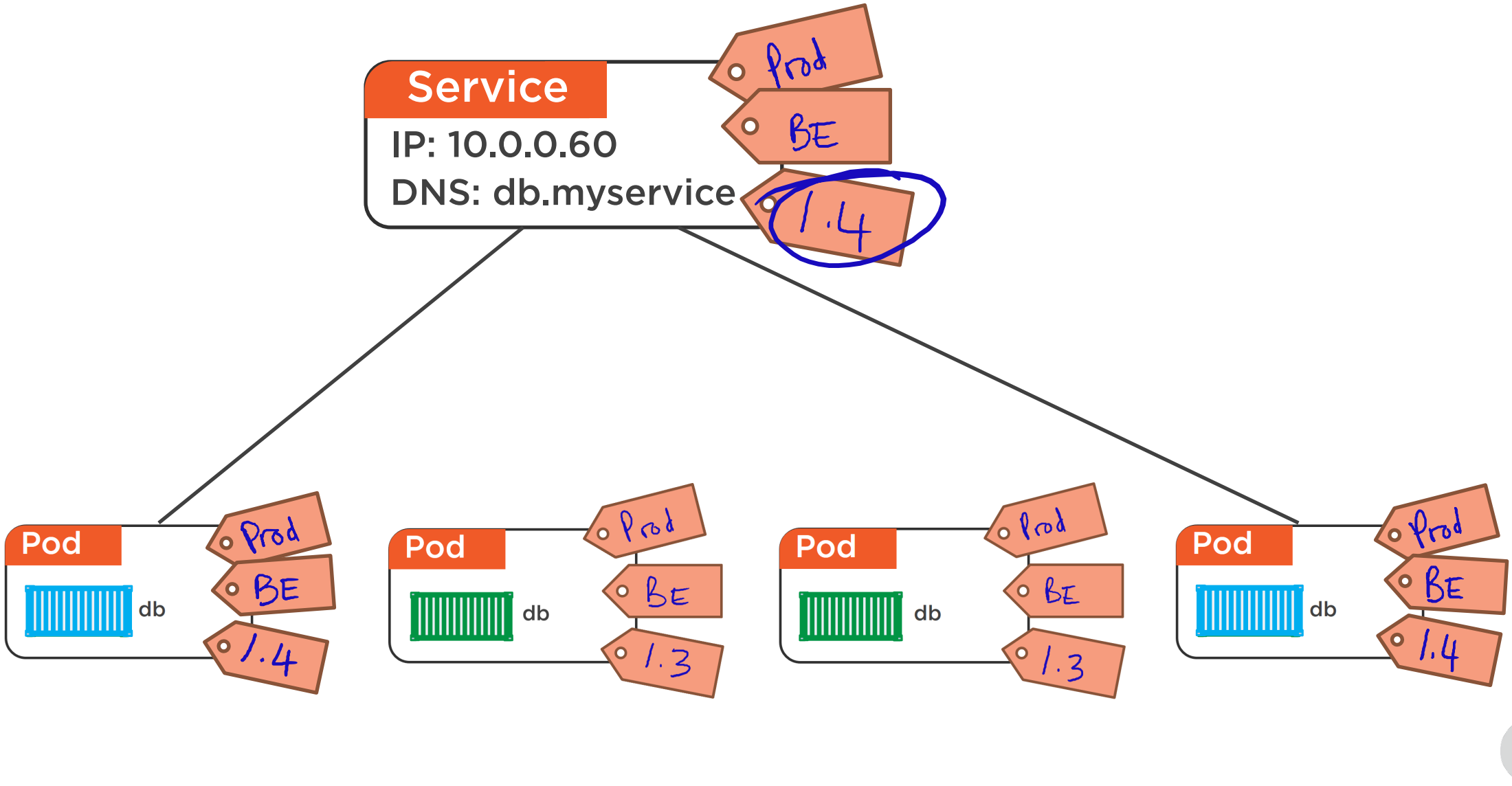

Pod
db
10.0.0.113
Prod
BE
1.3


Pod
db
10.0.0.32
Prod
BE
1.3


Service
IP: 10.0.0.60
DNS: db.myservice
Prod
BE
Pod
db
Prod
BE
1.4
Pod
db
Prod
BE
1.3
Pod
db
Prod
BE
1.3
Pod
db
Prod
BE
1.4

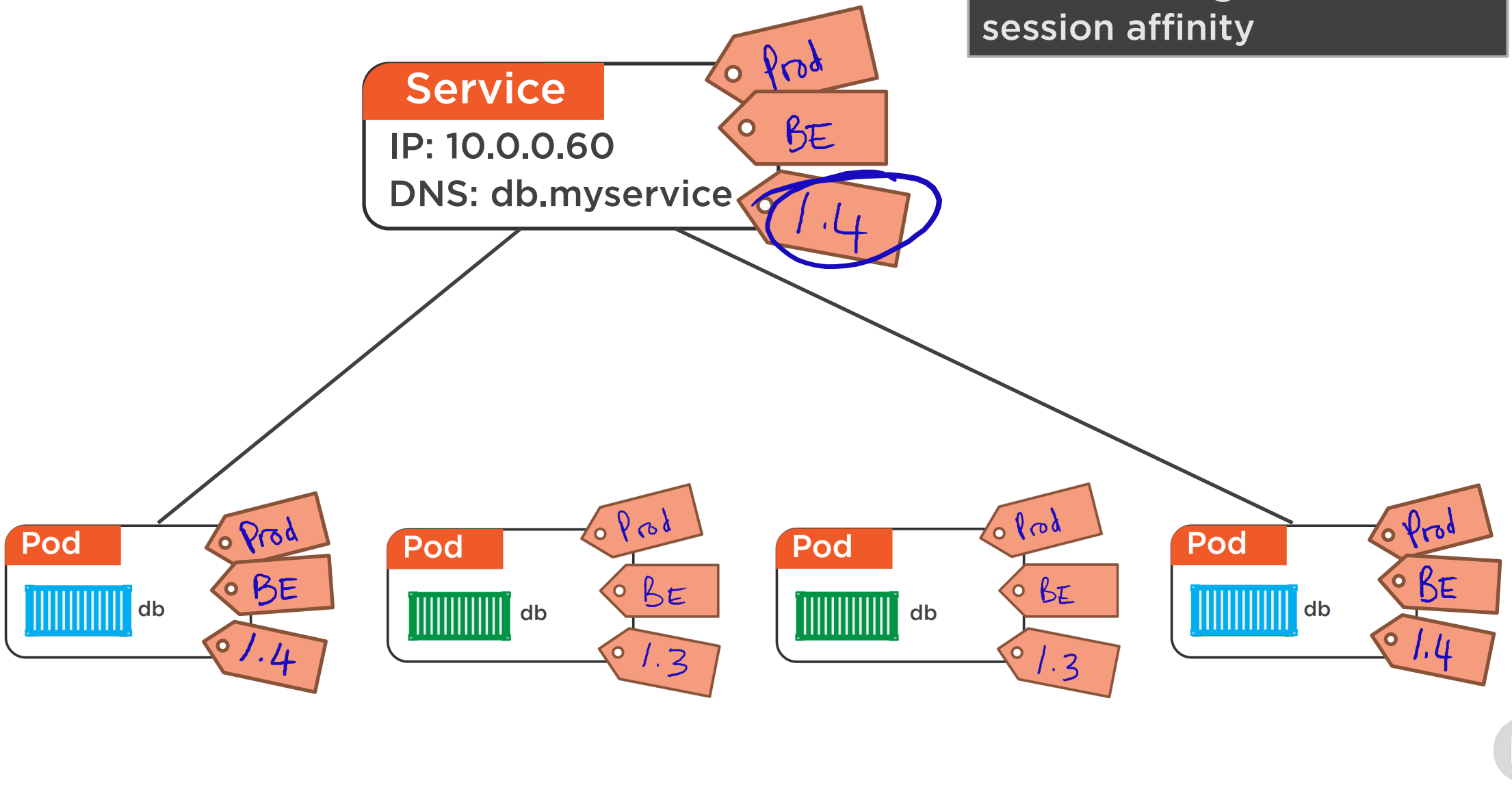

Service
IP: 10.0.0.60
DNS: db.myservice
Prod
BE
1.4
Pod
db
Prod
BE
1.4
Pod
db
Prod
BE
1.3
Pod
db
Prod
BE
1.3
Pod
db
Prod
BE
1.4

Only send to healthy pods



Only send to healthy pods

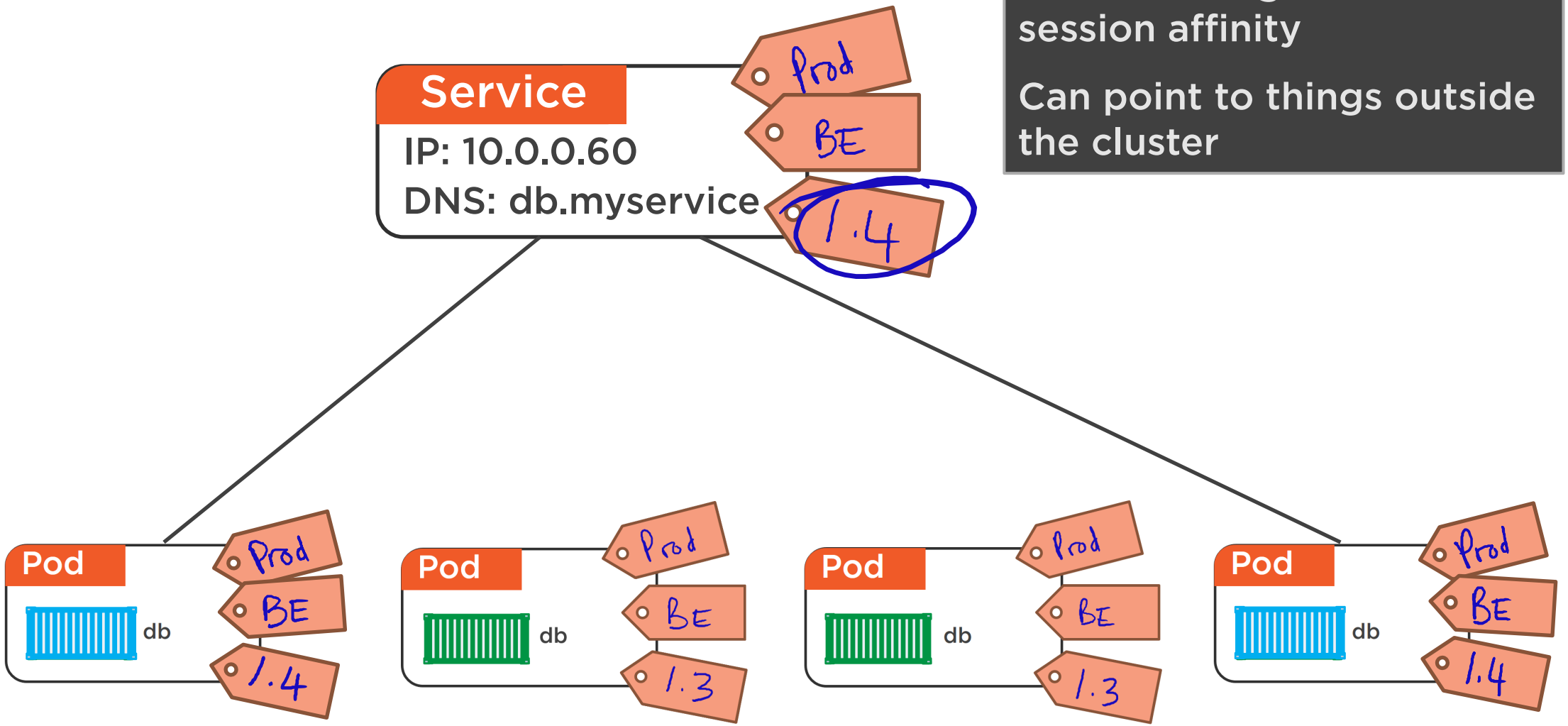
Can be configured for session affinity



Only send to healthy pods

Can be configured for session affinity

Can point to things outside the cluster

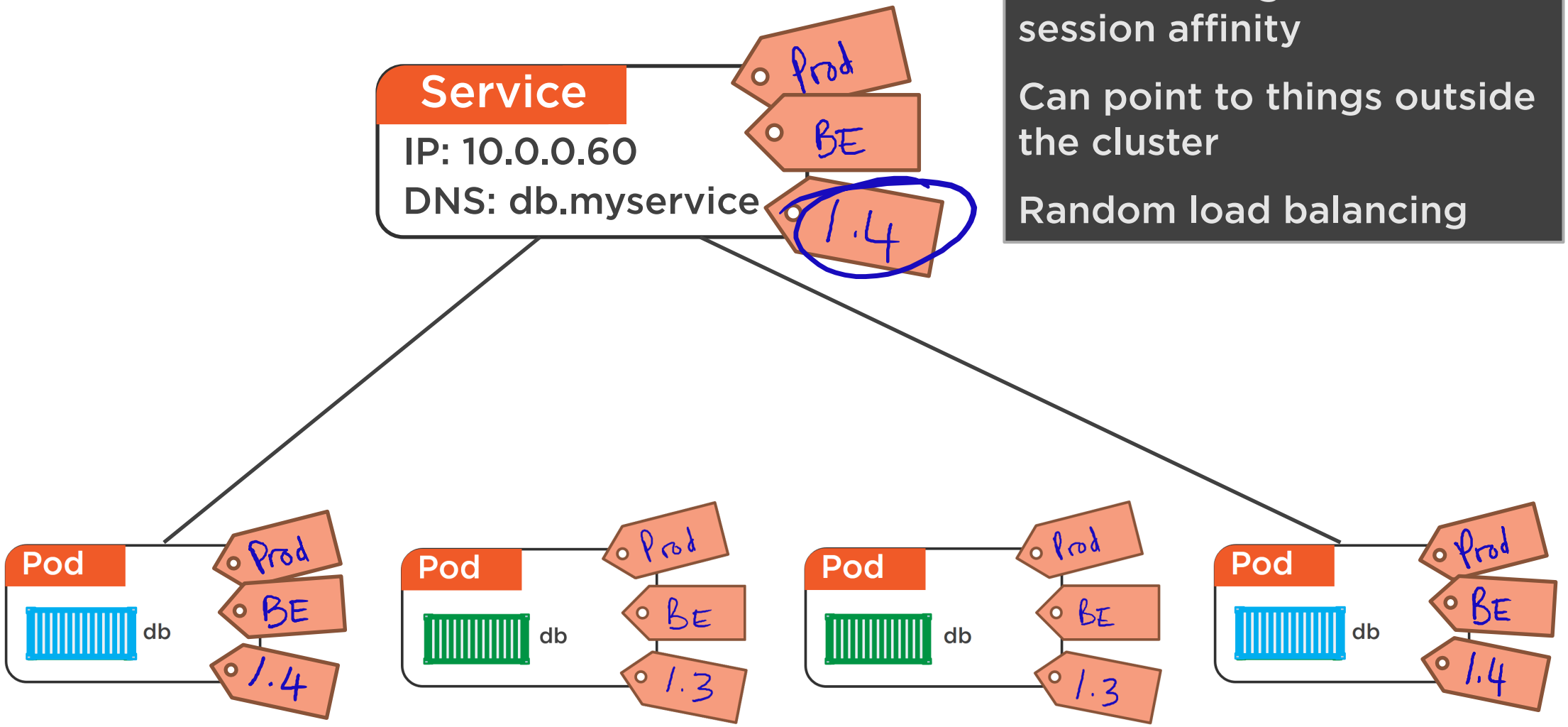


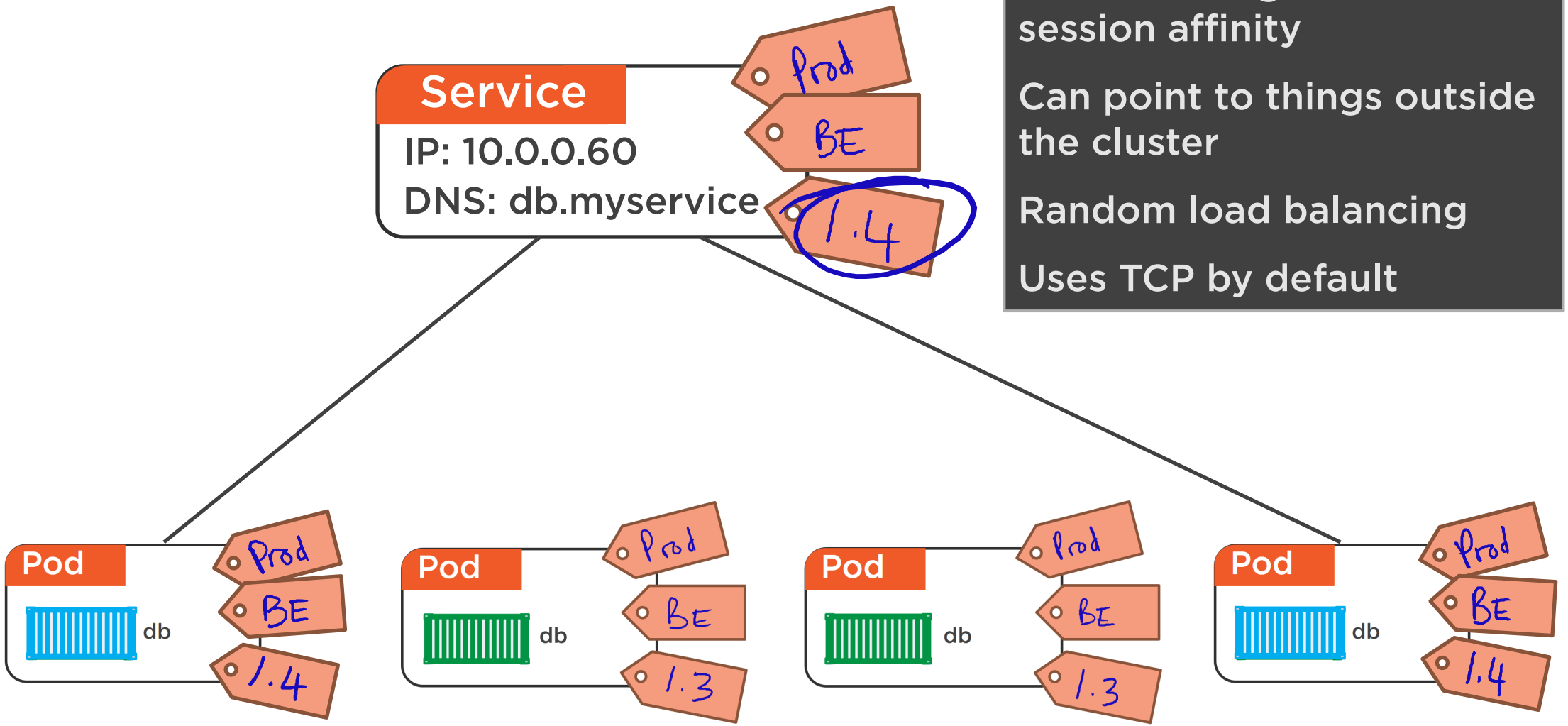
Only send to healthy pods

Can be configured for session affinity

Can point to things outside the cluster

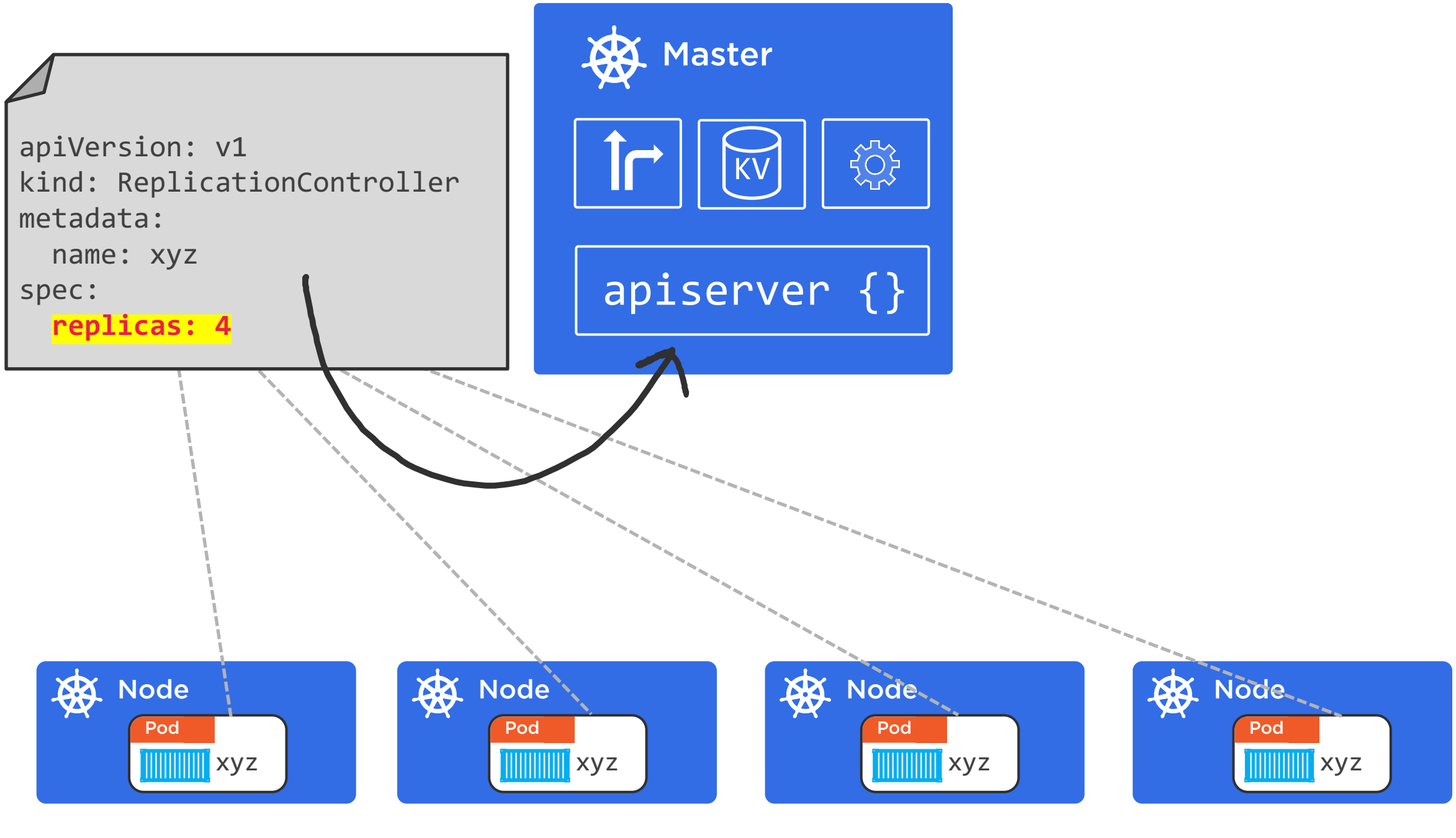
Random load balancing

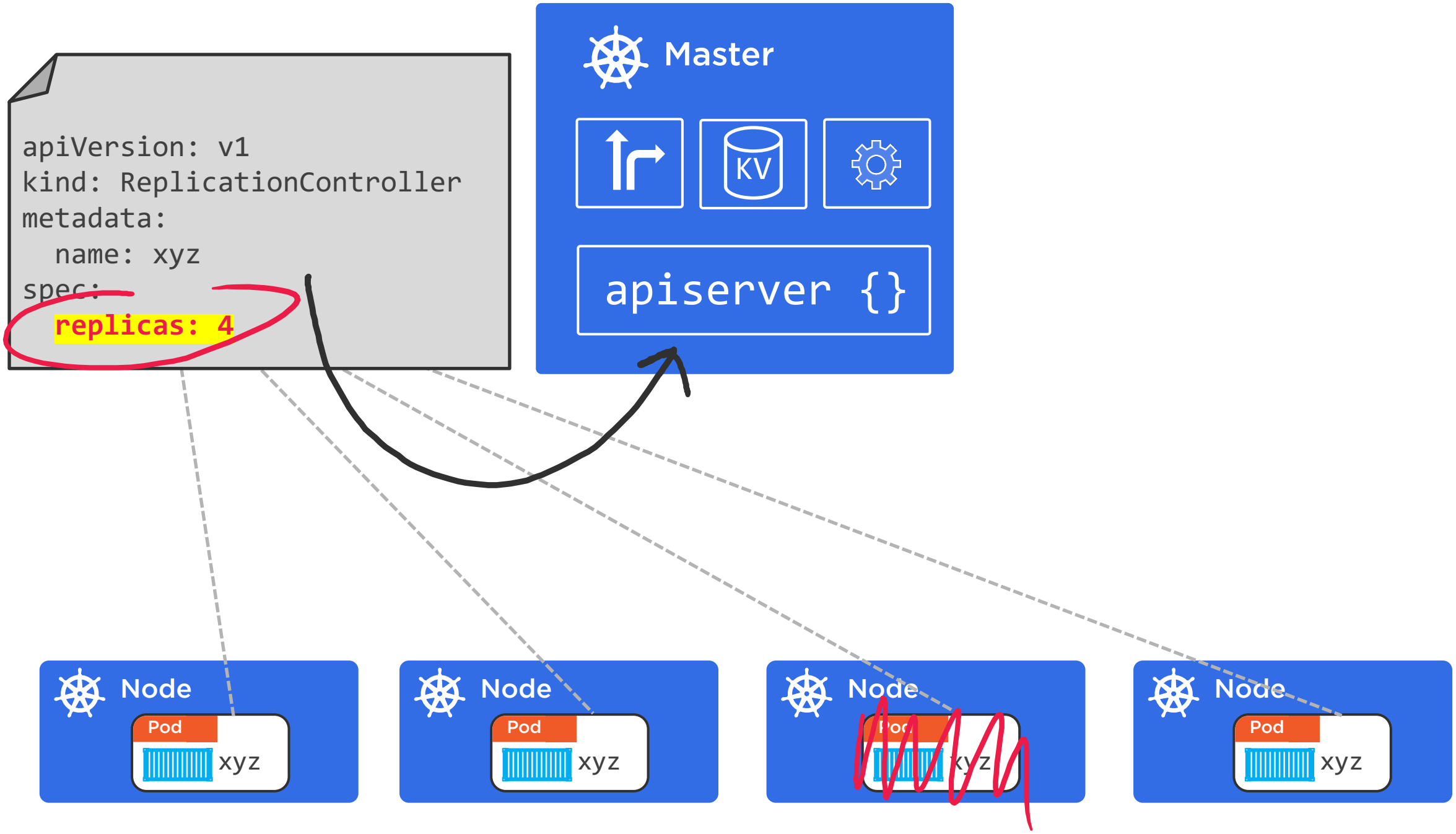


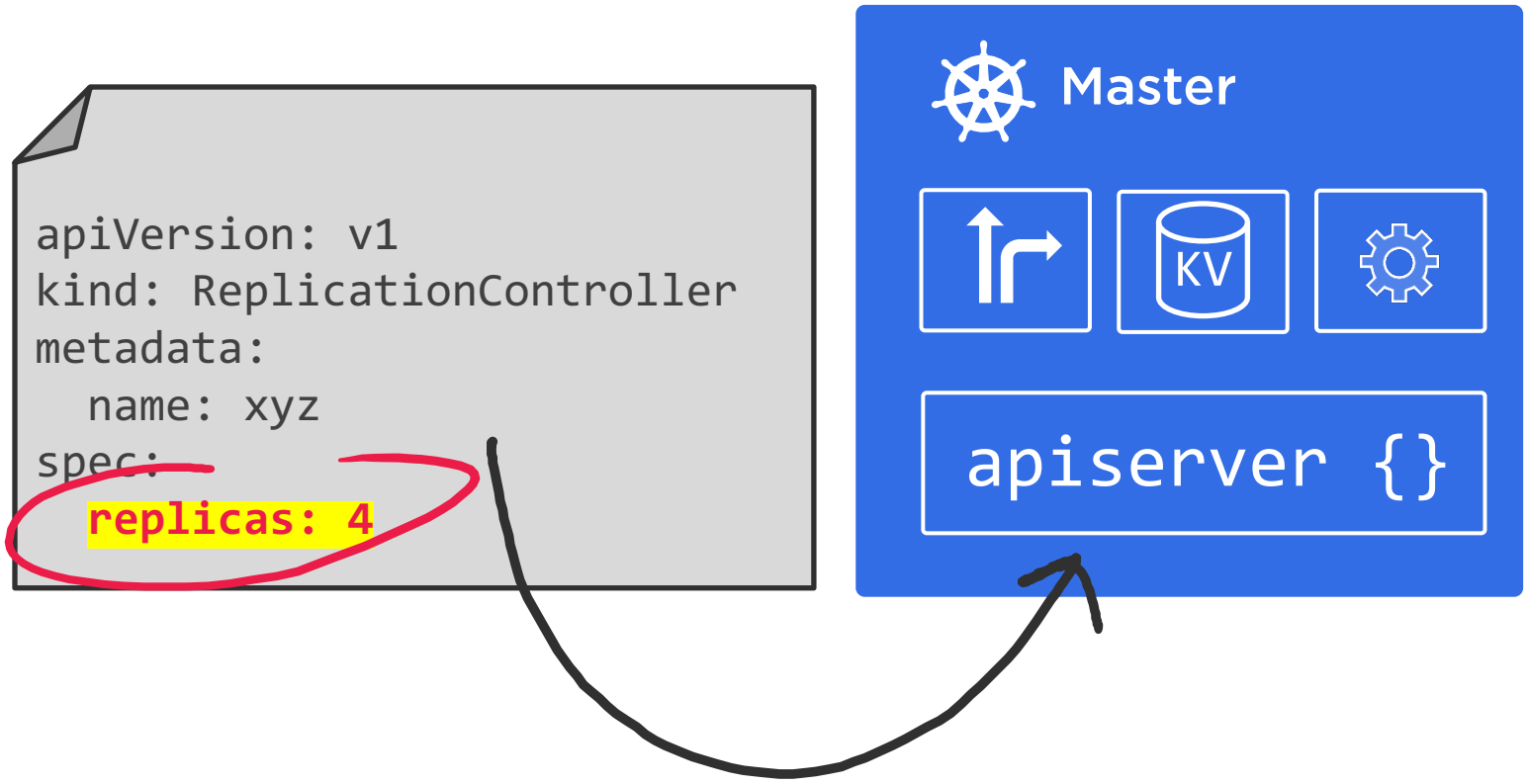


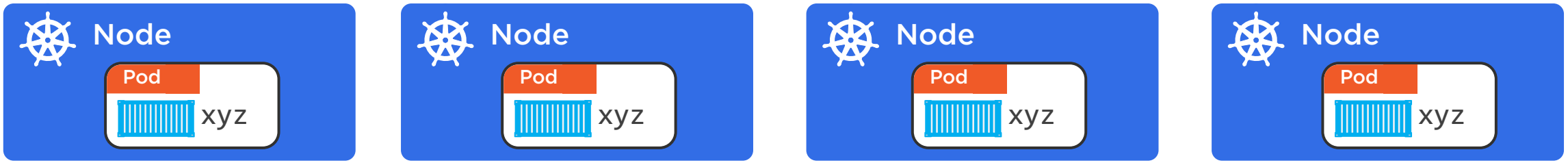
- Only send to healthy pods
- Can be configured for session affinity
- Can point to things outside the cluster
- Random load balancing
- Uses TCP by default

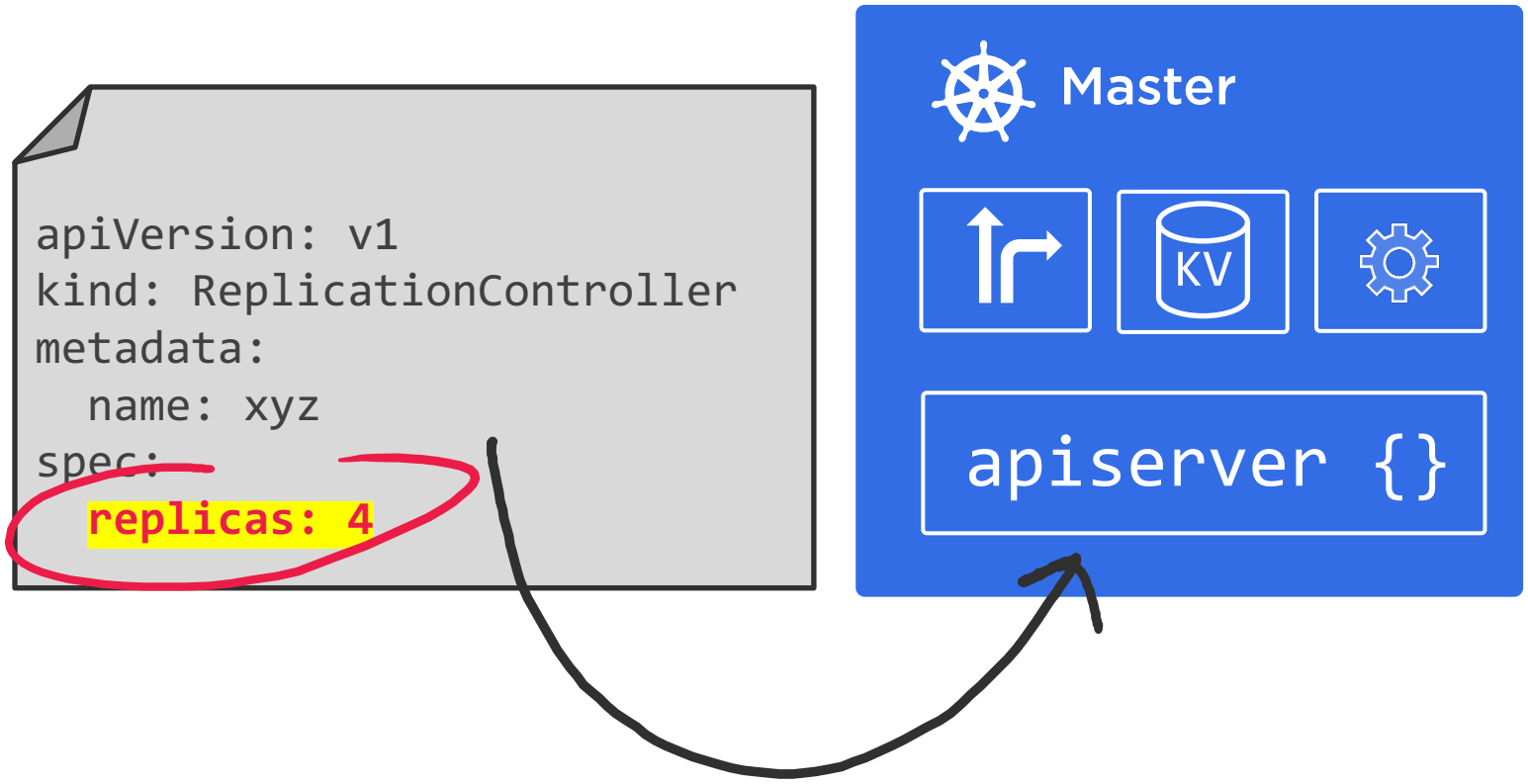
Deployments

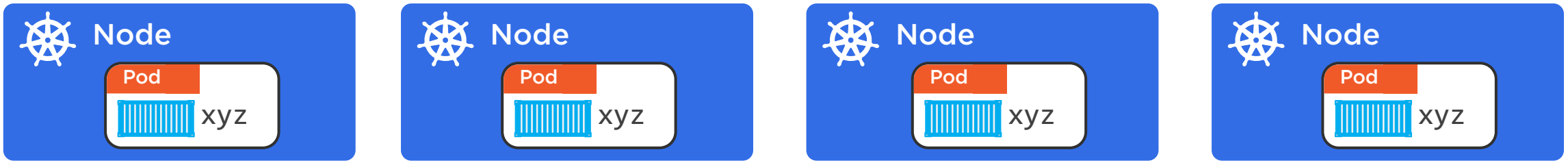

apiVersion: v1
kind: ReplicationController
metadata:
name: xyz
spec:
replicas: 4
Master
KV
apiserver {}
Node
Pod
xyz
Node
Pod
xyz
Node
Pod
xyz
Node
Pod
xyz

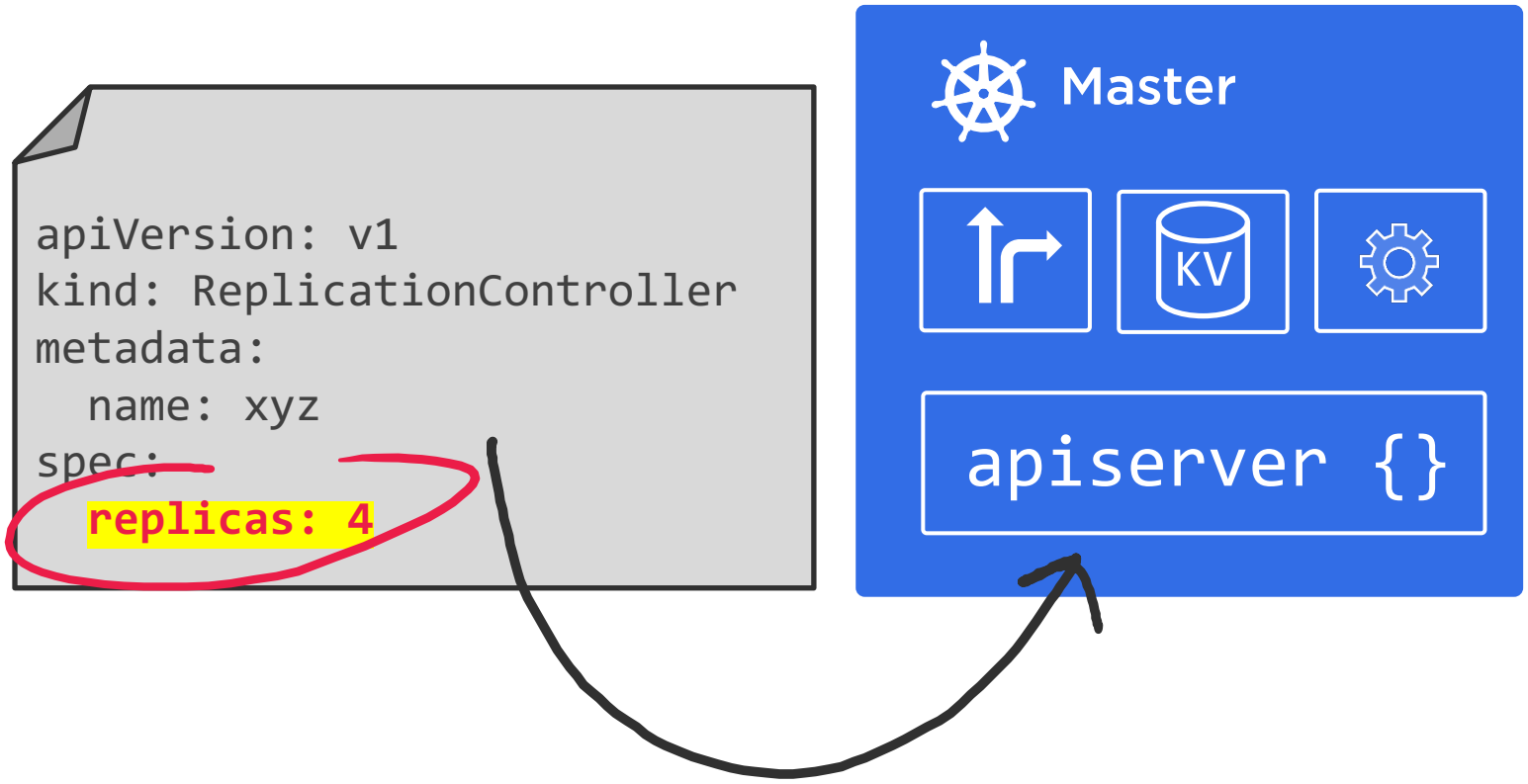

apiVersion: v1
kind: ReplicationController
metadata:
name: xyz
spec:
replicas: 4
Master
KV
apiserver {}
Node
Pod
xyz
Node
Pod
xyz
Node
Pod
xyz
Node
Pod
xyz

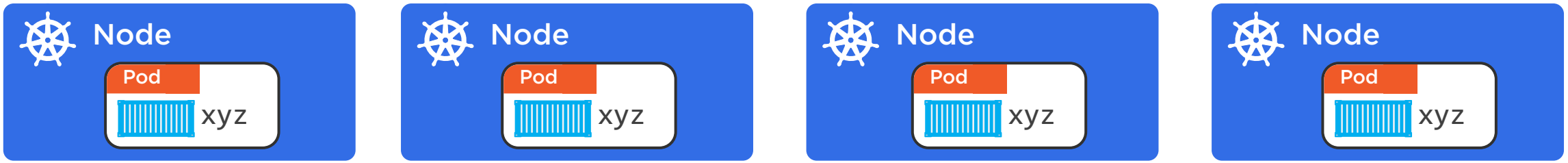

apiVersion: v1
kind: ReplicationController
metadata:
name: xyz
spec:
replicas: 4
Master
KV
apiserver {}


Node
Pod
xyz
Node
Pod
xyz
Node
Pod
xyz
Node
Pod
xyz


apiVersion: v1
kind: ReplicationController
metadata:
name: xyz
spec:
replicas: 4
Master
KV
apiserver {}


Node
Pod
xyz
Node
Pod
xyz
Node
Pod
xyz
Node
Pod
xyz


apiVersion: v1
kind: ReplicationController
metadata:
name: xyz
spec:
replicas: 4
Master
KV
apiserver {}


Node
Pod
xyz
Node
Pod
xyz
Node
Pod
xyz
Node
Pod
xyz

REST objects

Self documenting

Deployed via YAML or JSON manifests

Spec-once deploy-many

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: xyz
spec:
  replicas: 4
```

Simple rolling updates and rollbacks

Add features to Replication Controllers (Replica Sets) RC v2

Versioned

Deployed via the `apiserver`

REST objects

Self documenting

Deployed via YAML or JSON manifests

Spec-once deploy-many

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: xyz
spec:
  replicas: 4
```

Simple rolling updates and rollbacks

Add features to Replication Controllers (Replica Sets) RC v2

Versioned

Deployed via the `apiserver`

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: xyz
spec:
  replicas: 4
```

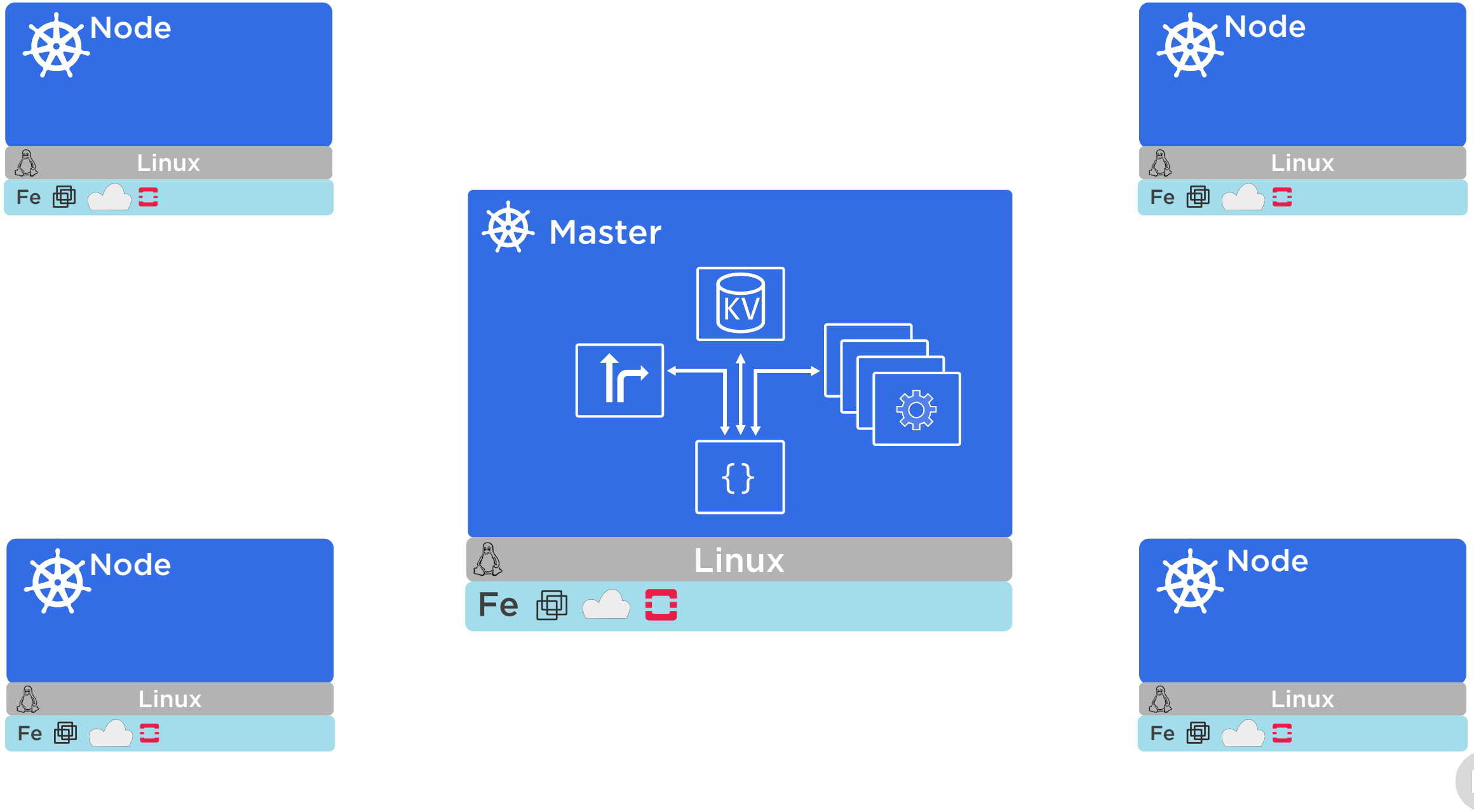
Simple rolling updates and rollbacks

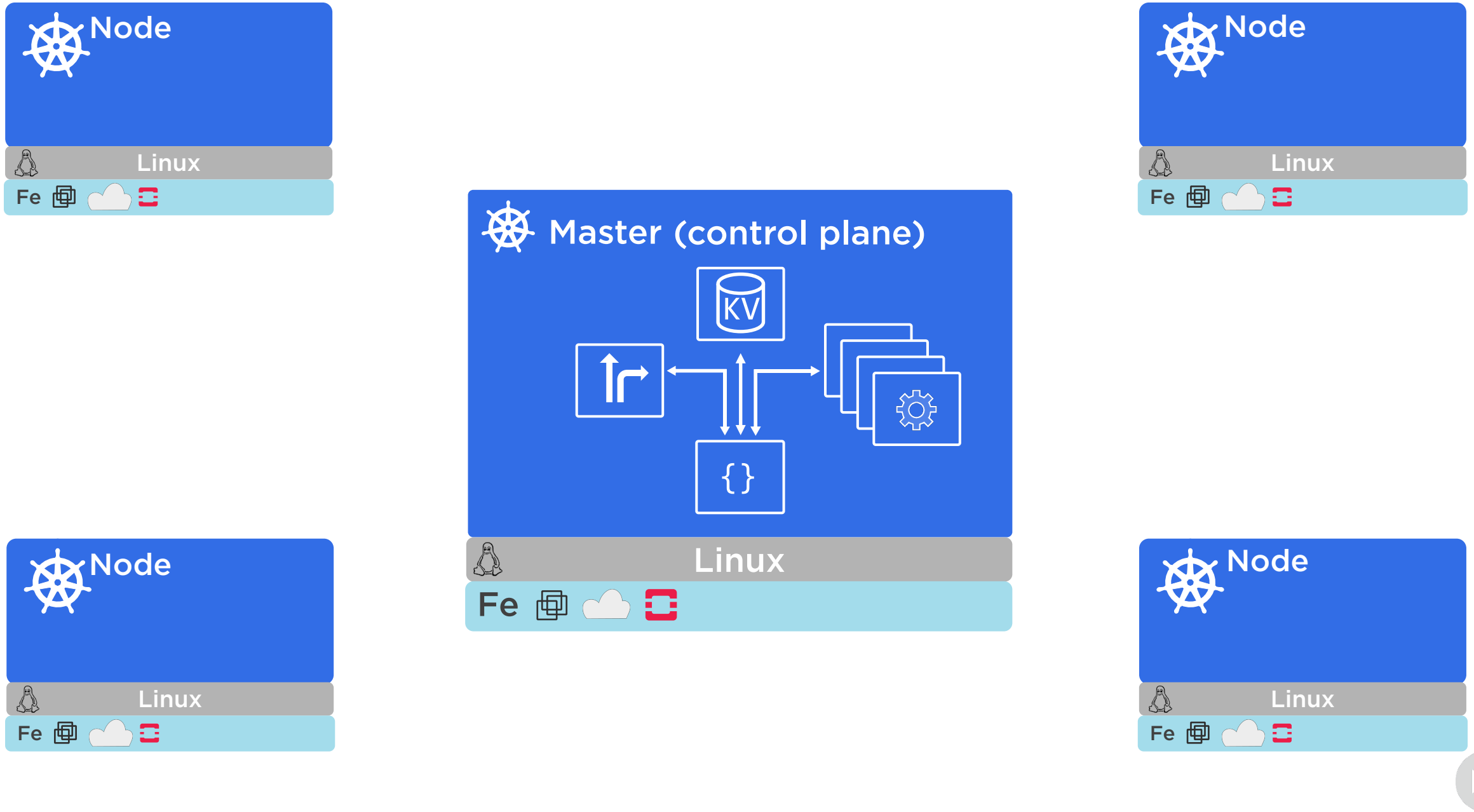
Multiple concurrent versions
- Blue-green deployments
- Canary releases

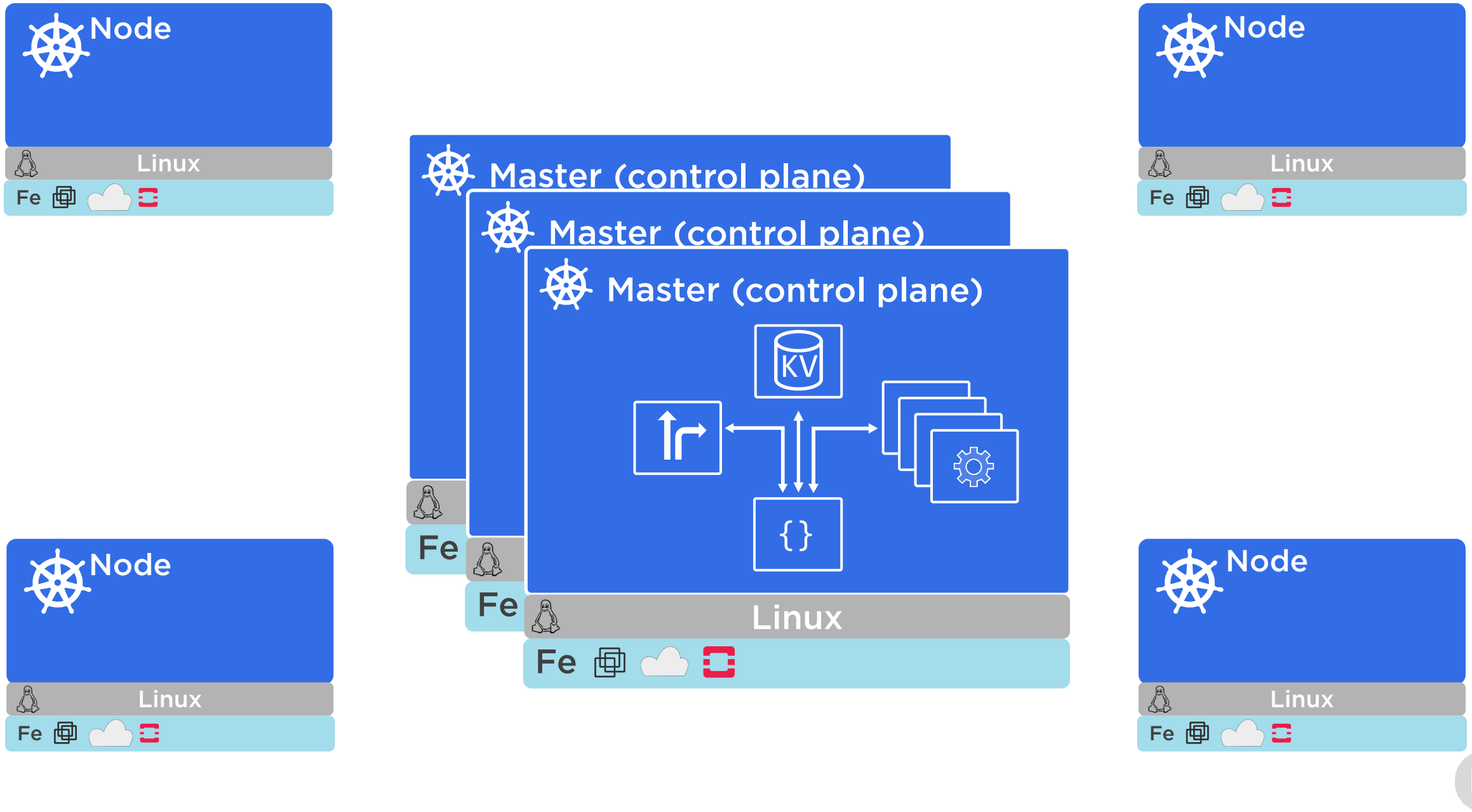
Simple versioned rollbacks

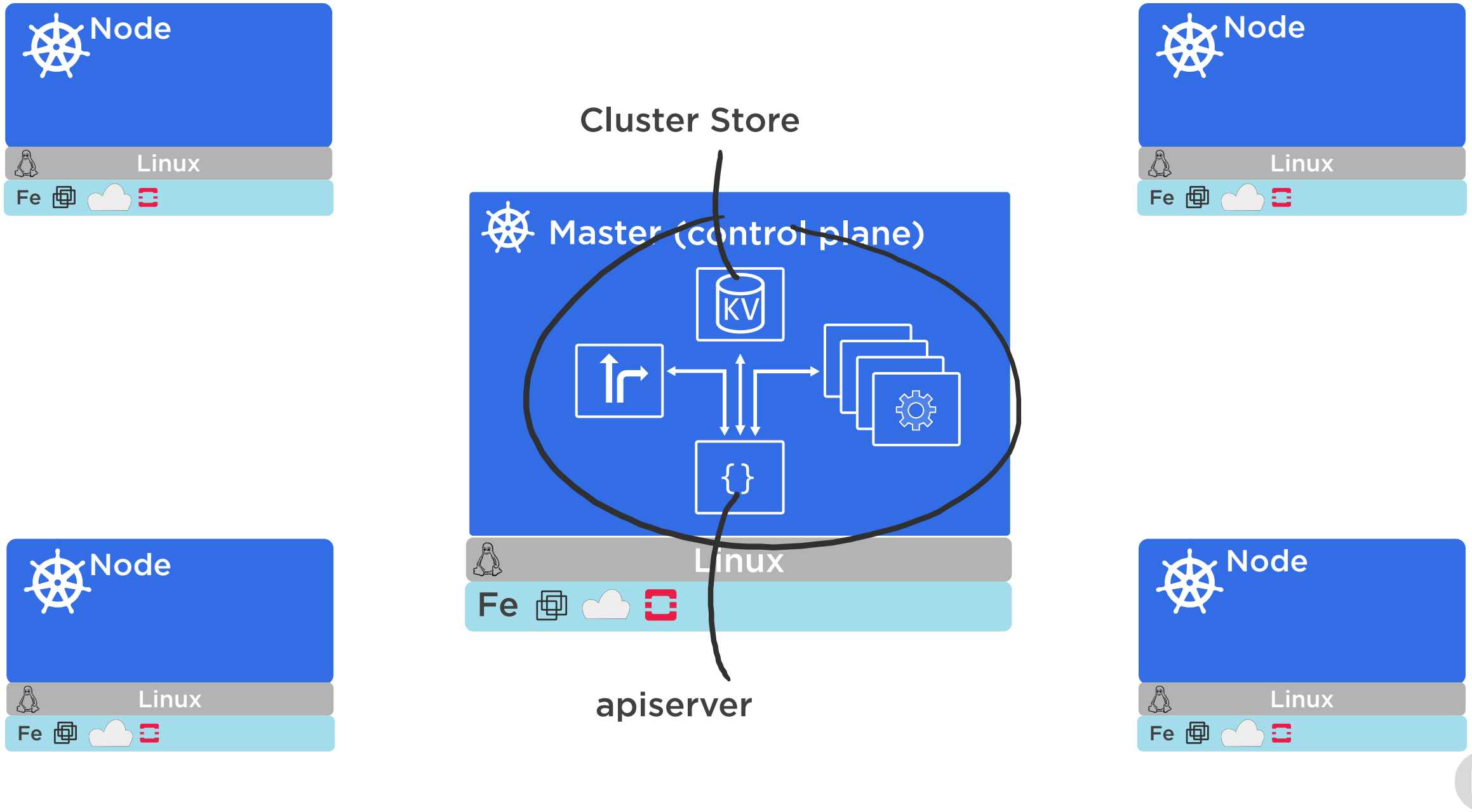
Bringing it home!

containerized apps!


Node
Linux
Fe
Node
Linux
Fe
Master
KV
{}
Linux
Fe
Node
Linux
Fe
Node
Linux
Fe


Node
Linux
Fe
Node
Linux
Fe
Master (control plane)
KV
{}
Linux
Fe
Node
Linux
Fe
Node
Linux
Fe


Node
Linux
Fe
Node
Linux
Fe
Master (control plane)
Master (control plane)
Master (control plane)
KV
{}
Fe
Fe
Linux
Fe
Node
Linux
Fe
Node
Linux
Fe


Node
Linux
Fe
Node
Linux
Fe
Cluster Store
Master (control plane)
KV
{}
Linux
Fe
apiserver
Node
Linux
Fe
Node
Linux
Fe

Node

Linux

Fe

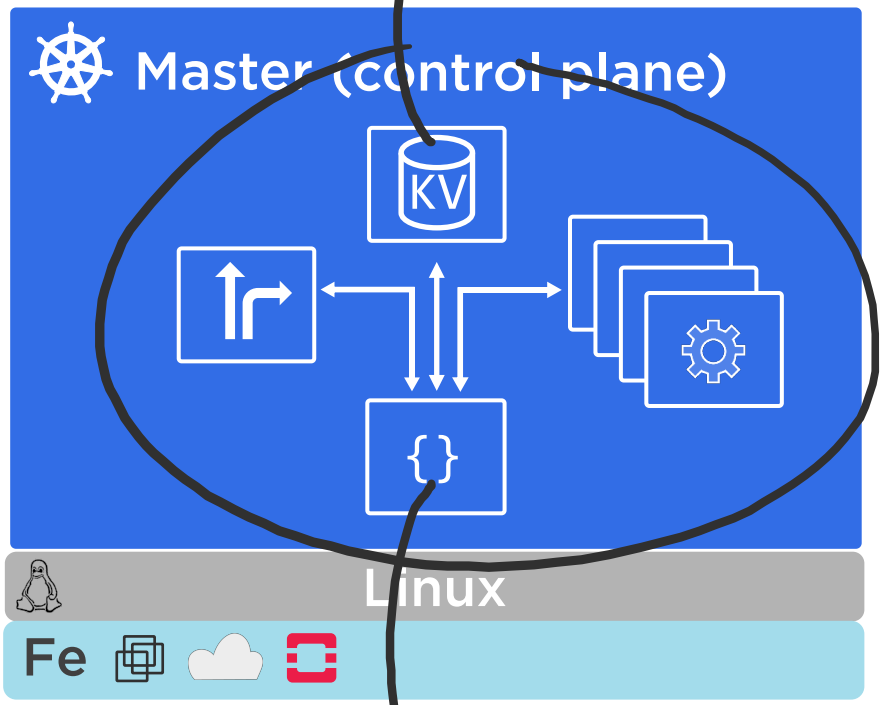
Node

Linux

Fe

Cluster Store

- Cluster state and config
- Stateful



Node

Linux

Fe

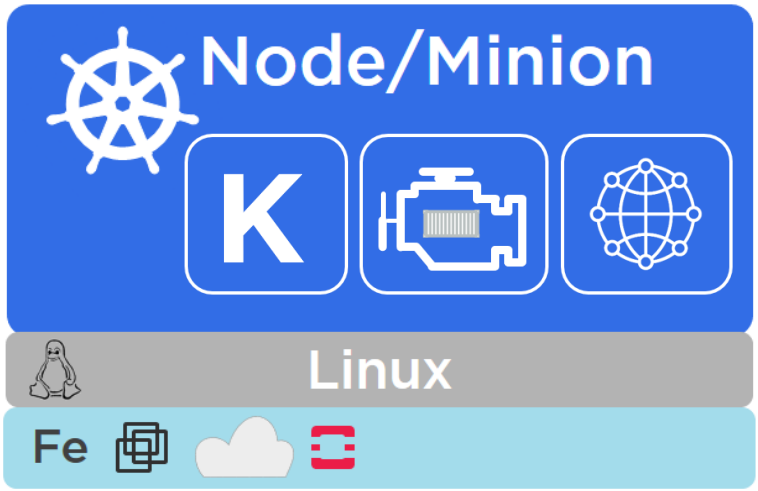
Node

Linux

Fe

apiserver

- Front-end to control plane



Kubelet

Main Kubernetes agent

Container engine

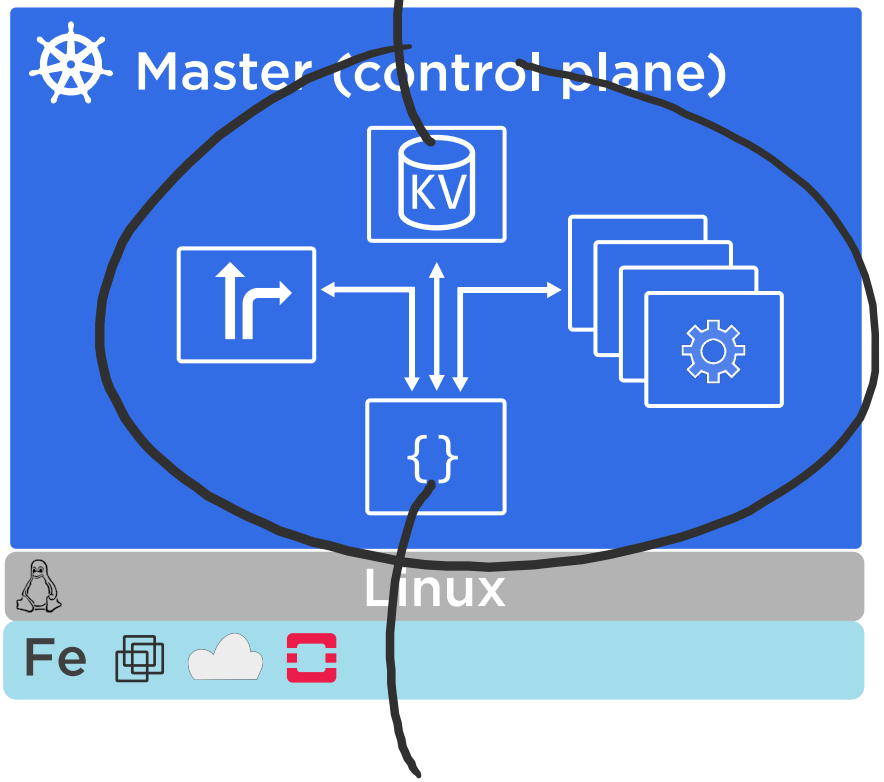
Docker or rkt

kube-proxy

Kubernetes networking

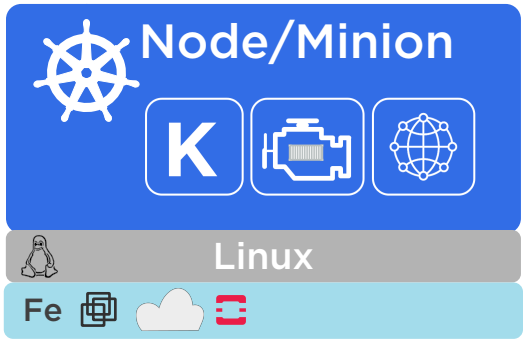
Cluster Store

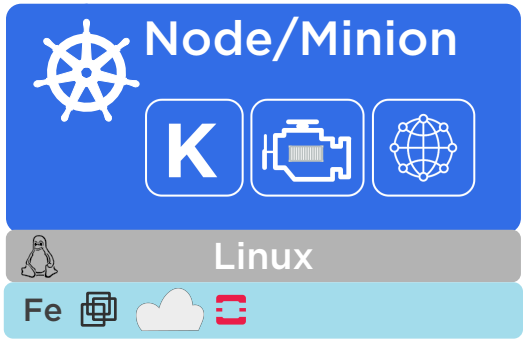
- Cluster state and config
- Stateful



apiserver

- Front-end to control plane





Objects in the K8s API

Pods : Atomic unit of scheduling...

Replication Controllers : Scale pods, desired state etc...

Deployments : RC + rolling updates, rollbacks...

Services : Stable networking...

Coming up next...

Installing Kubernetes