Kubernetes Architecture

Module Outline

Big picture view

Masters

Nodes

Pods

Services

Deployments

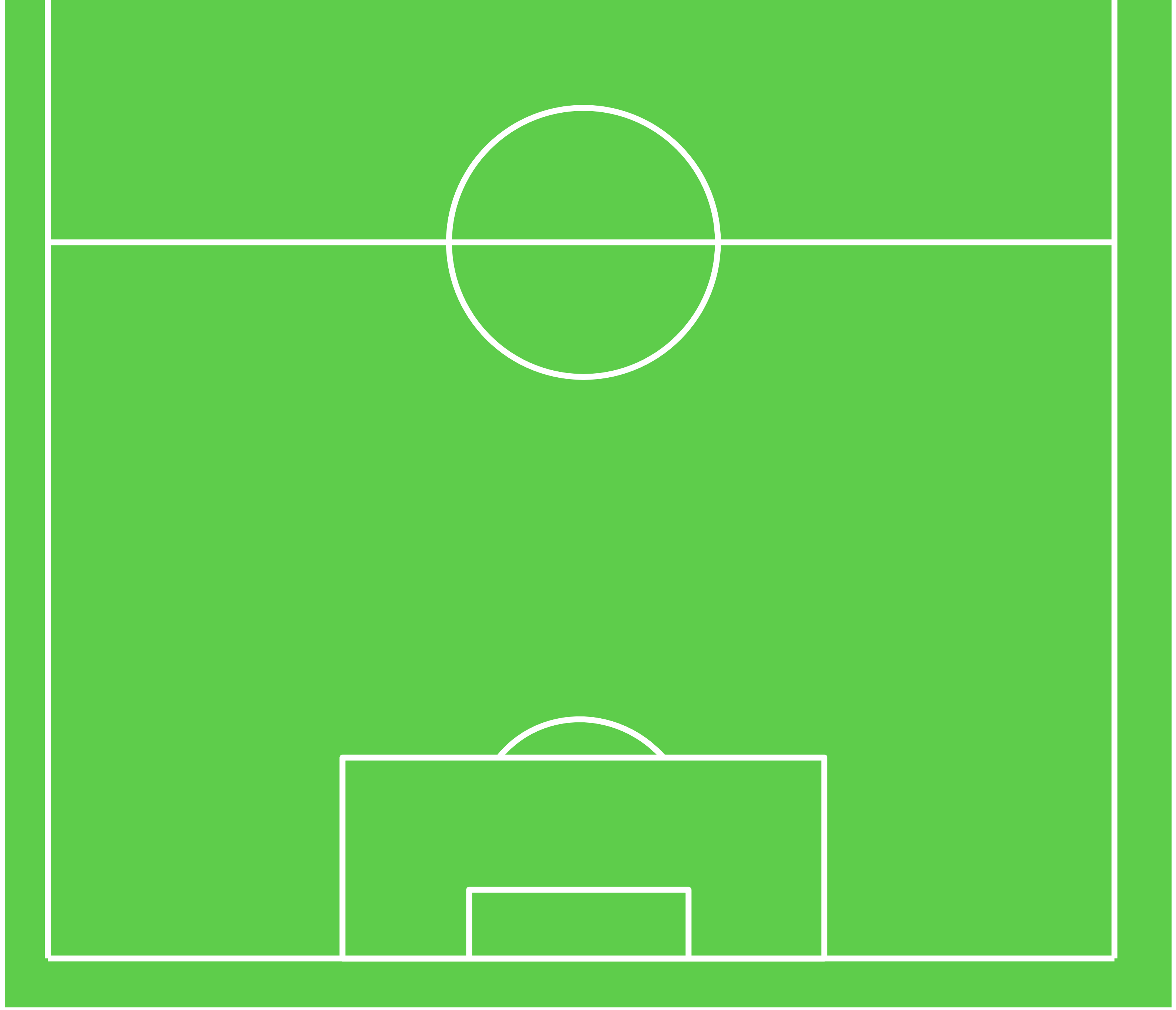
Recap

Kubernetes

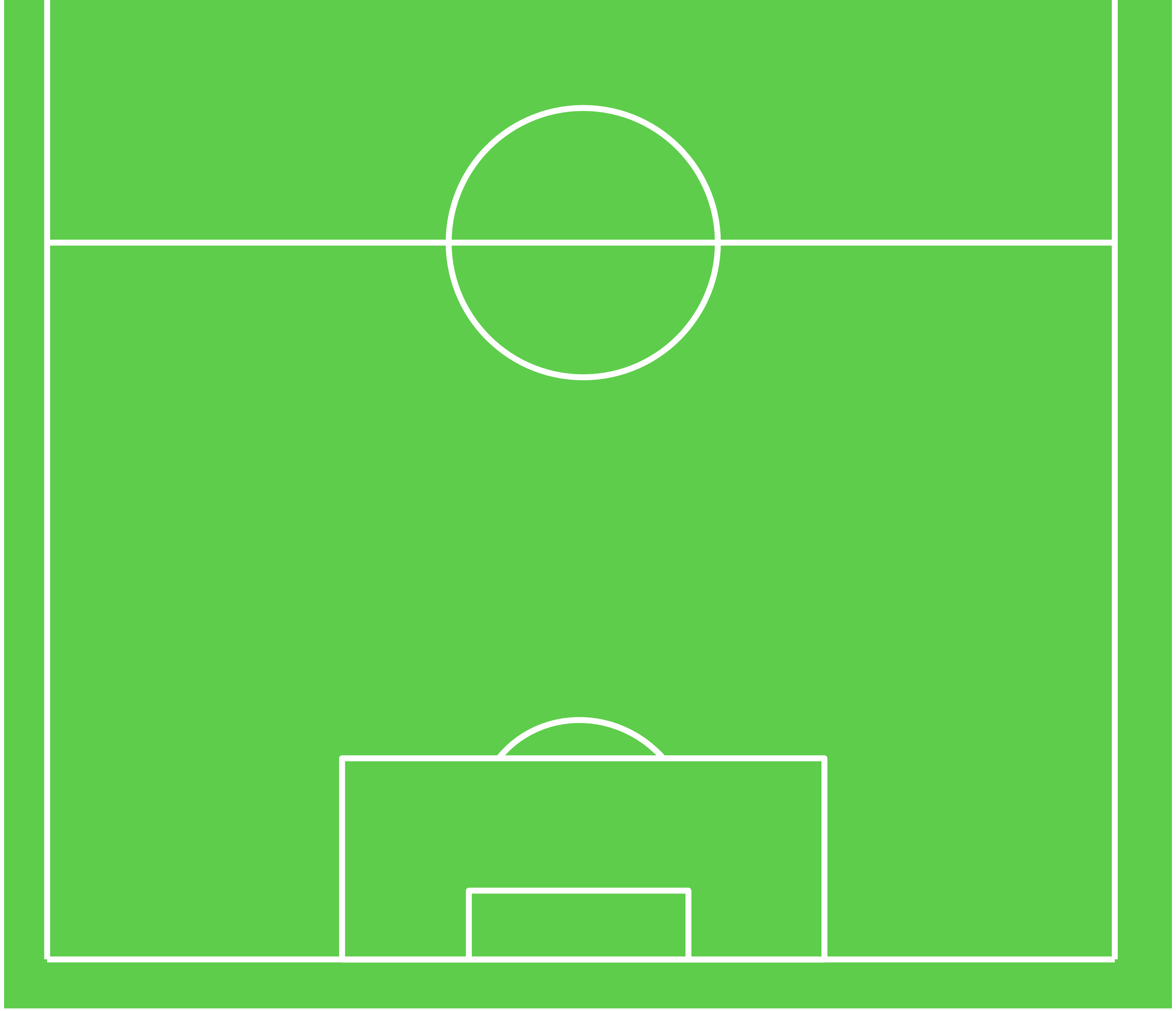
Big Picture View

Team

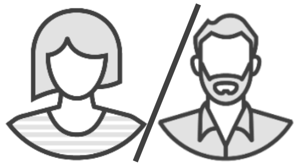
Manager (coach)



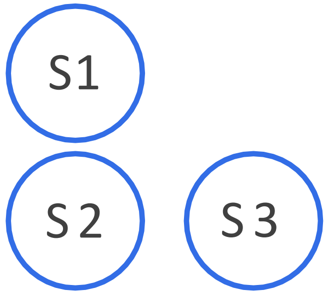
Team

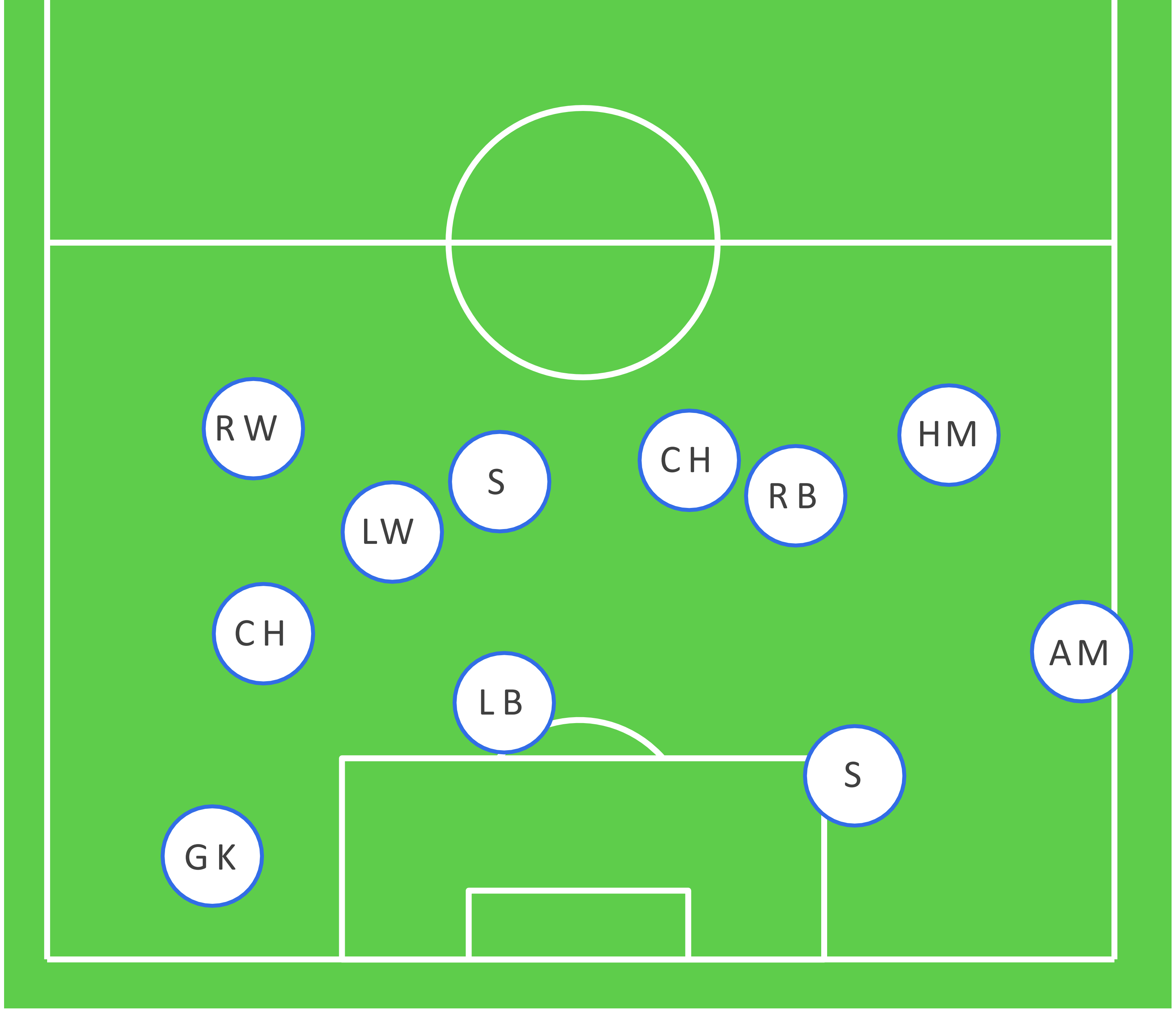


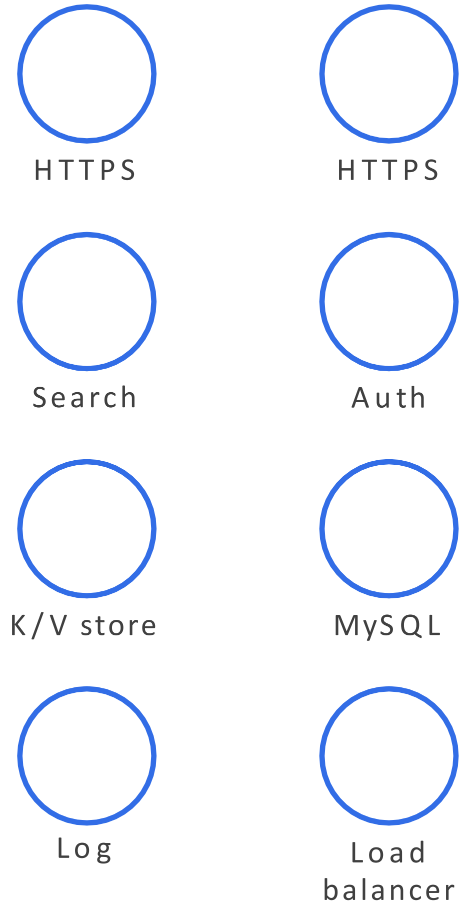
Team

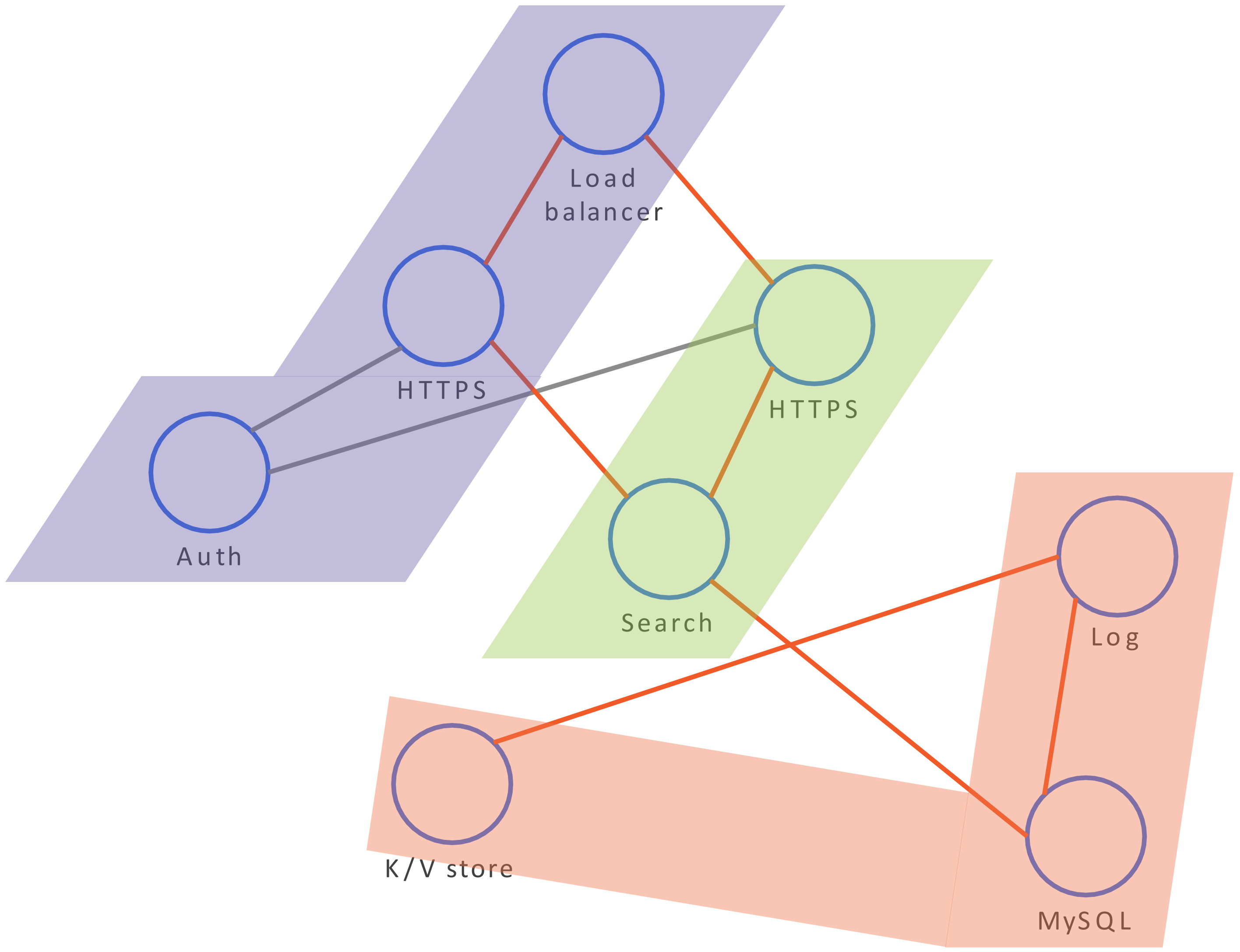


Manager (coach)

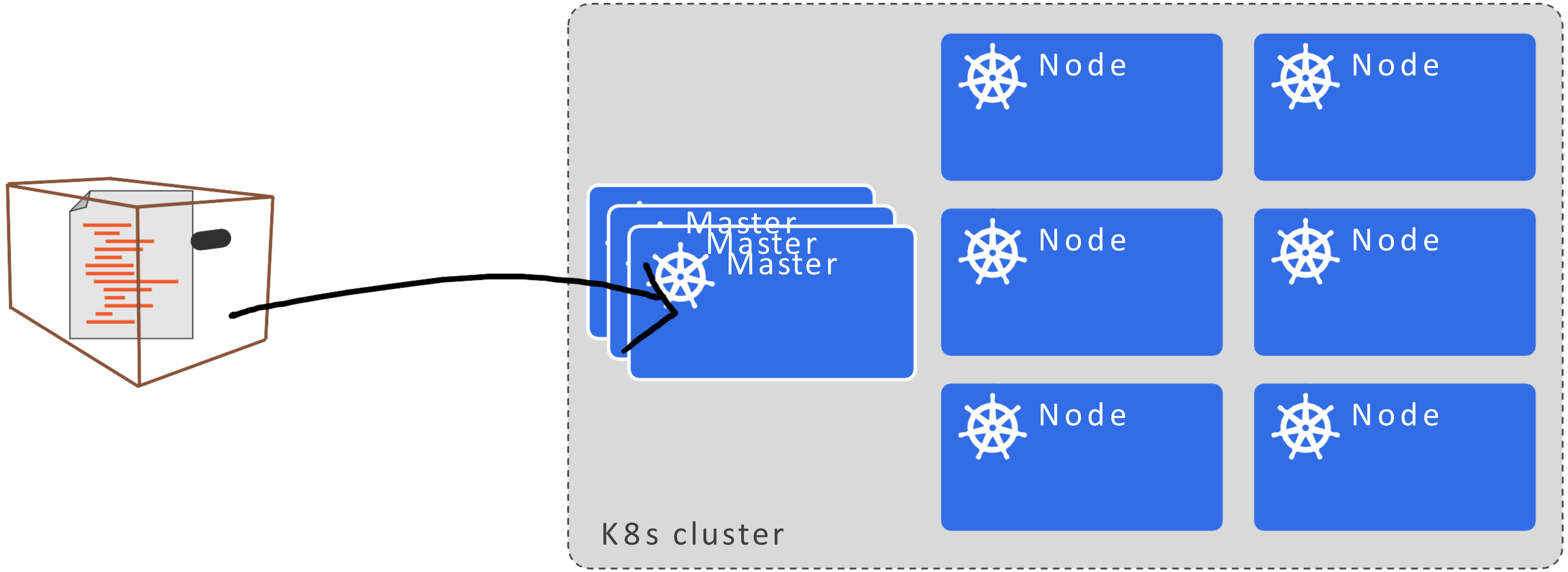




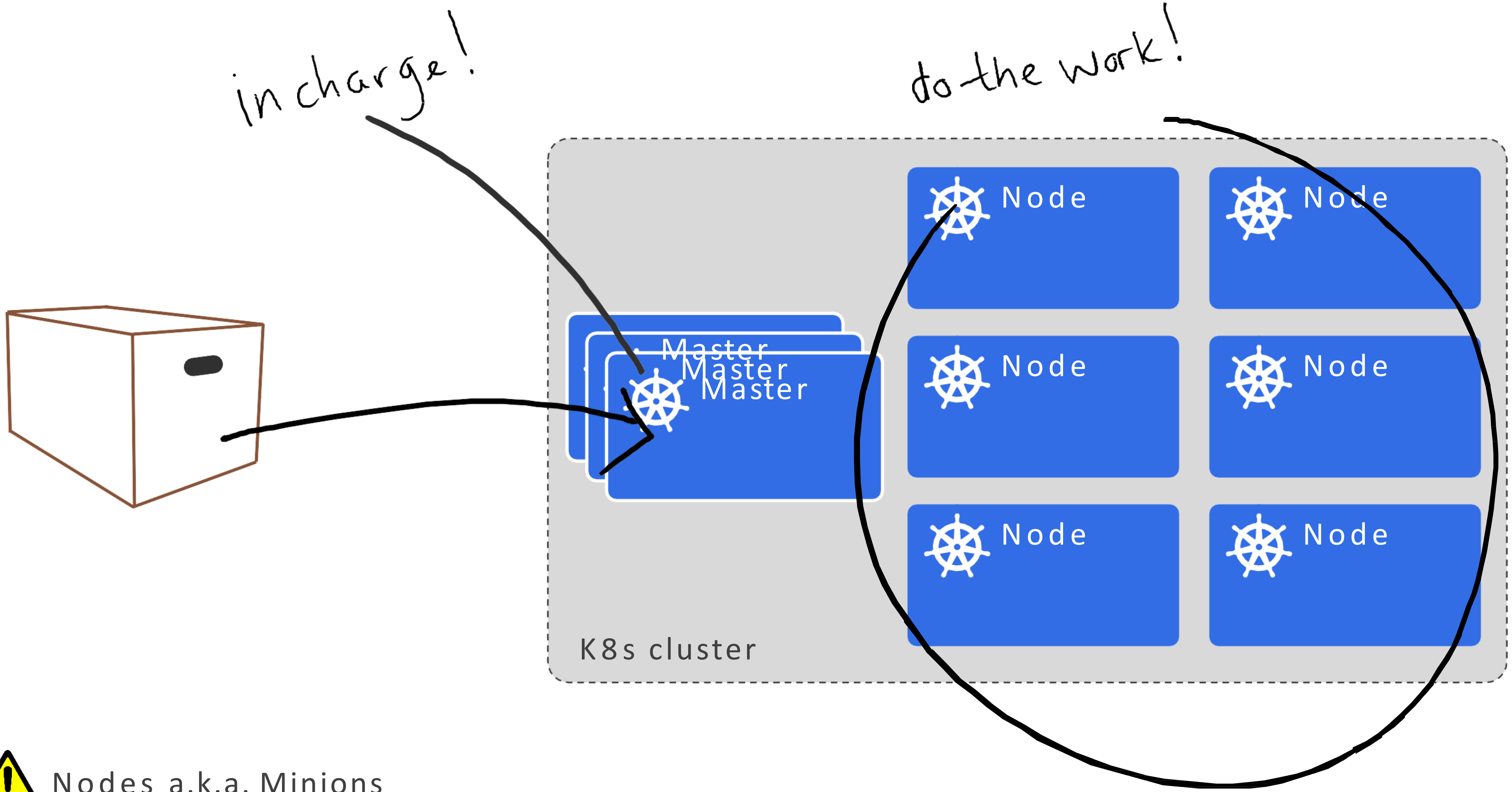

HTTPS
HTTPS
Search
Auth
K/V store
MySQL
Log
Load balancer



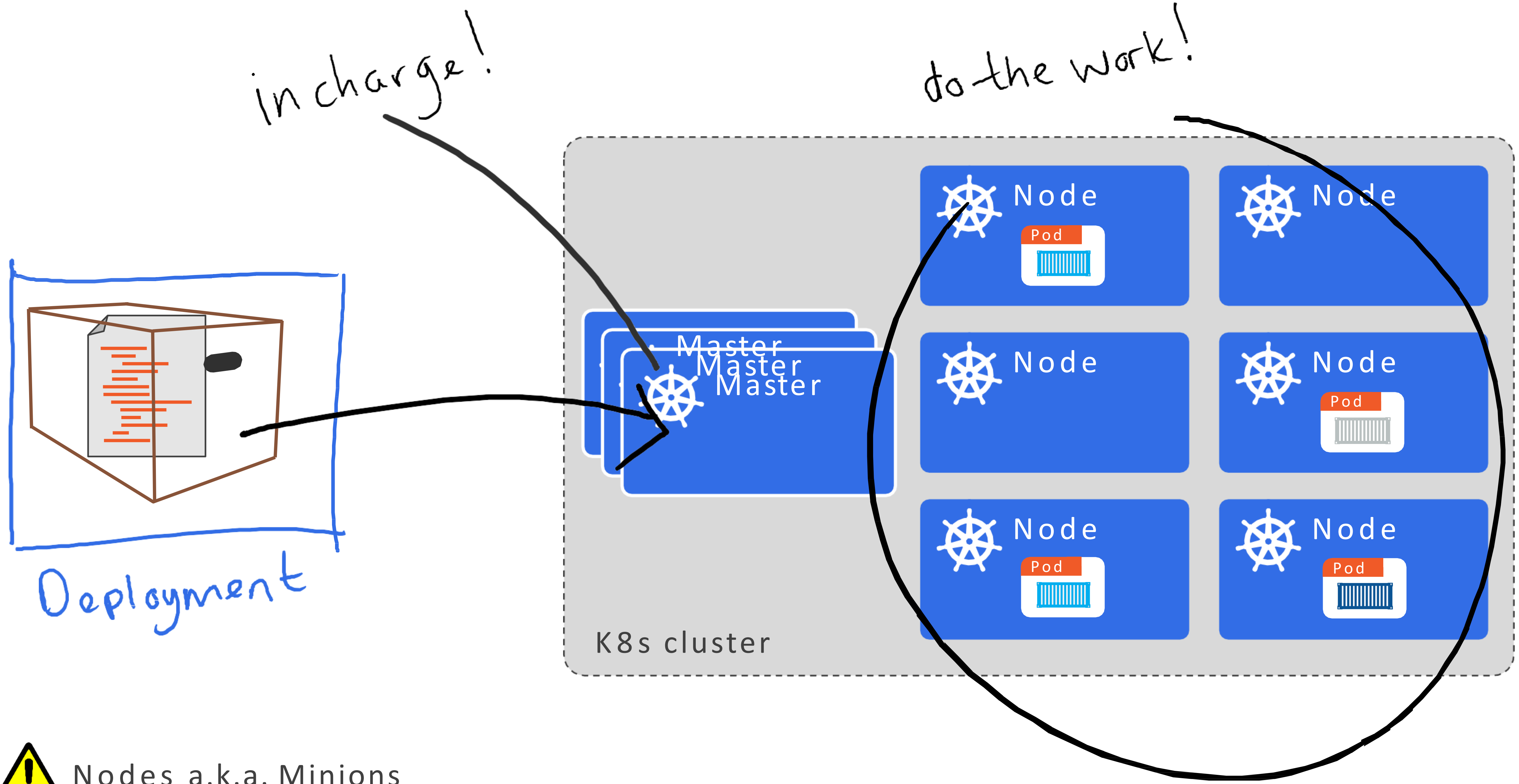
🟪	Node 1
🟩	Node 2
🟧	Node 3



⚠ Nodes a.k.a. Minions



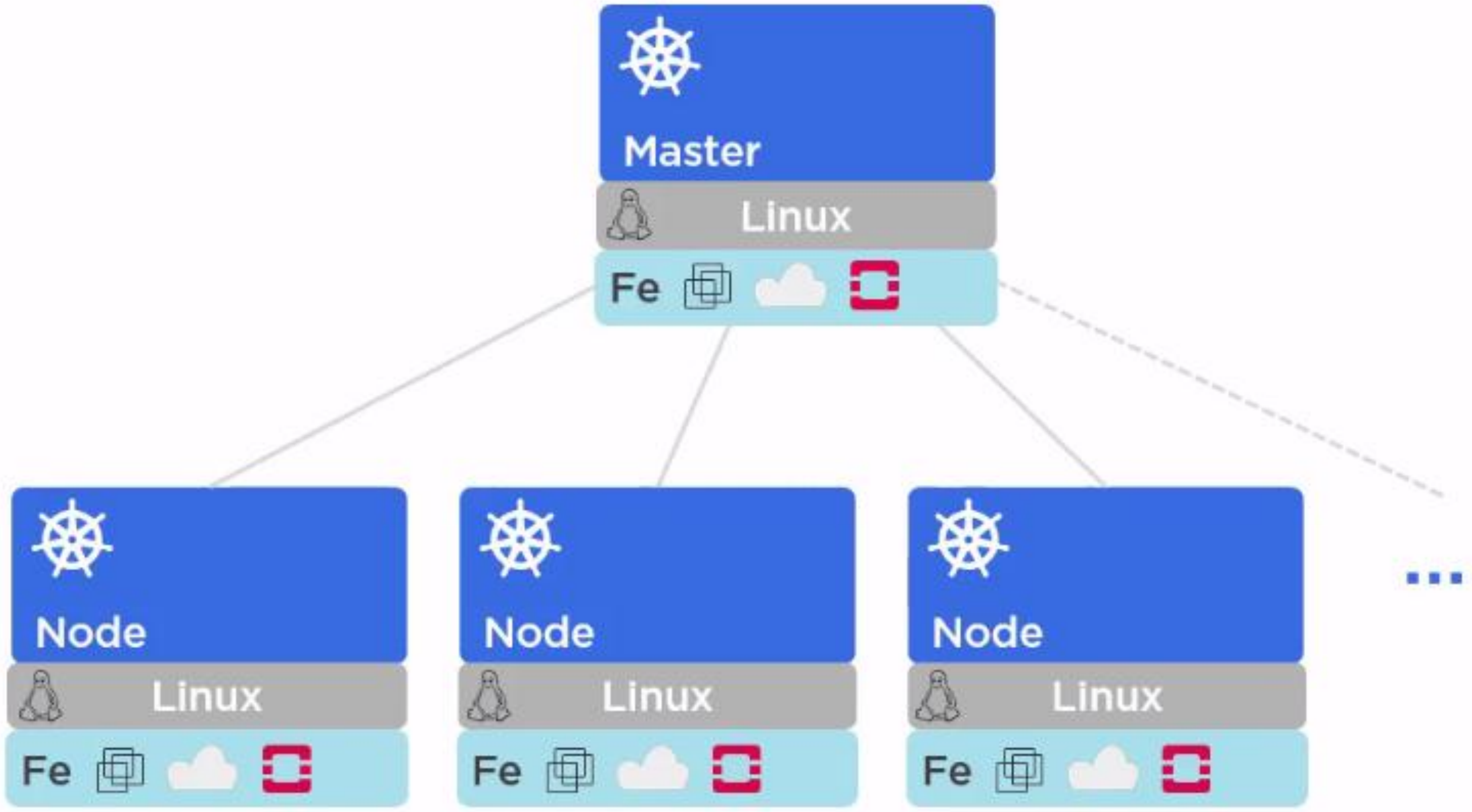
⚠ Nodes a.k.a. Minions

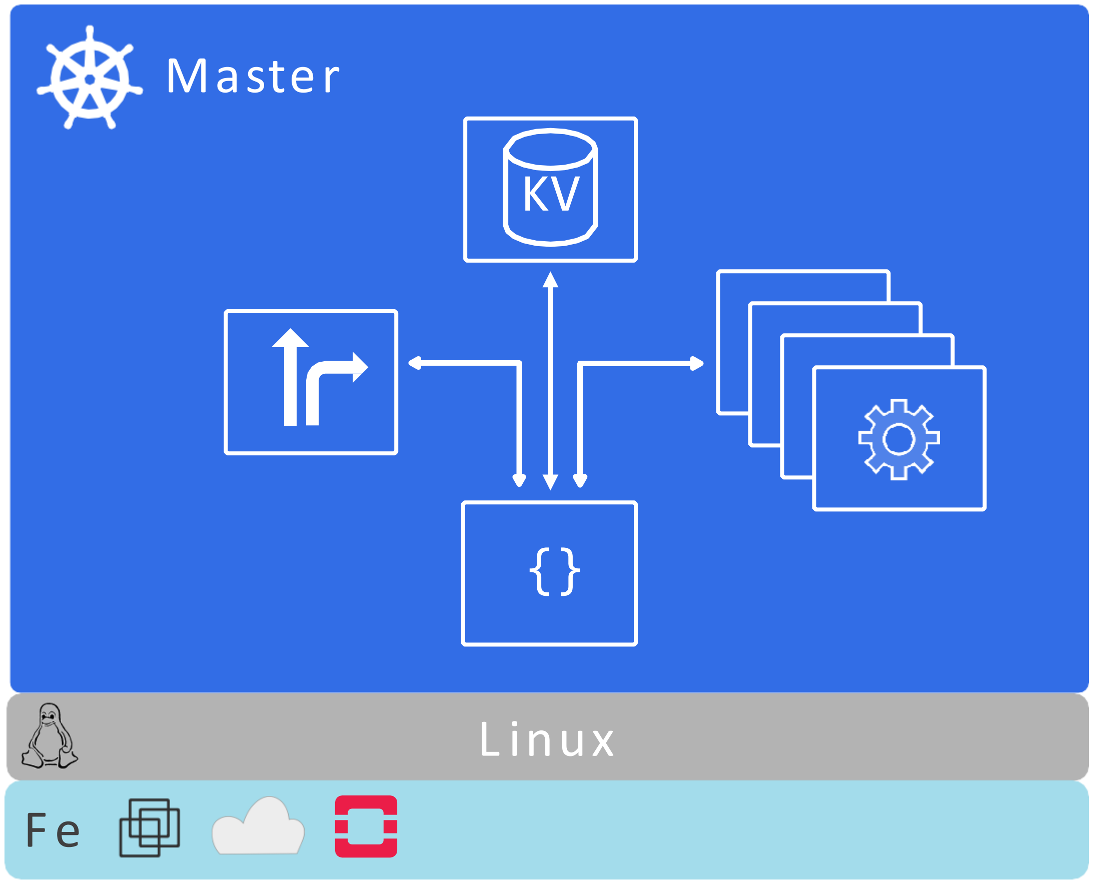


Nodes a.k.a. Minions

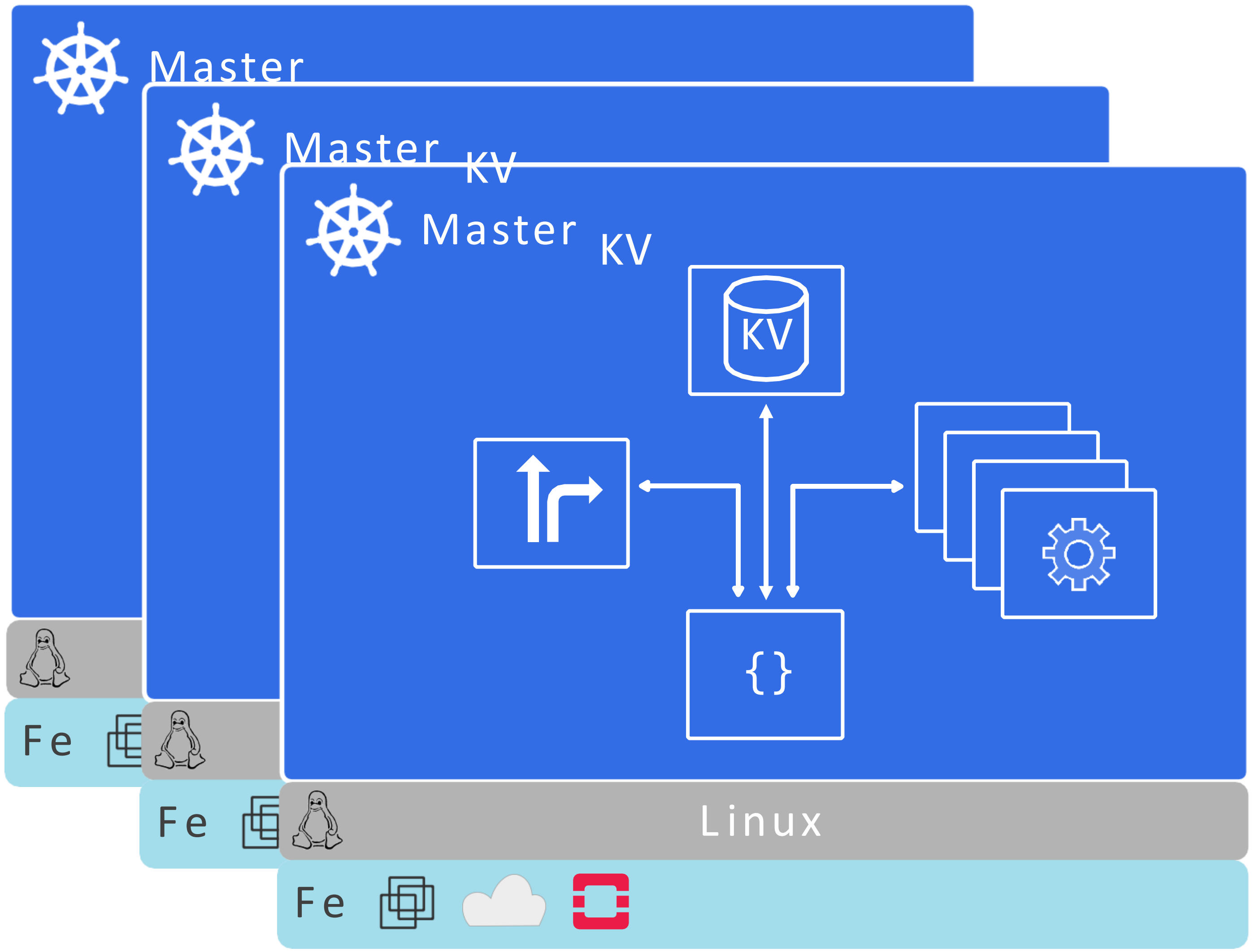
Masters

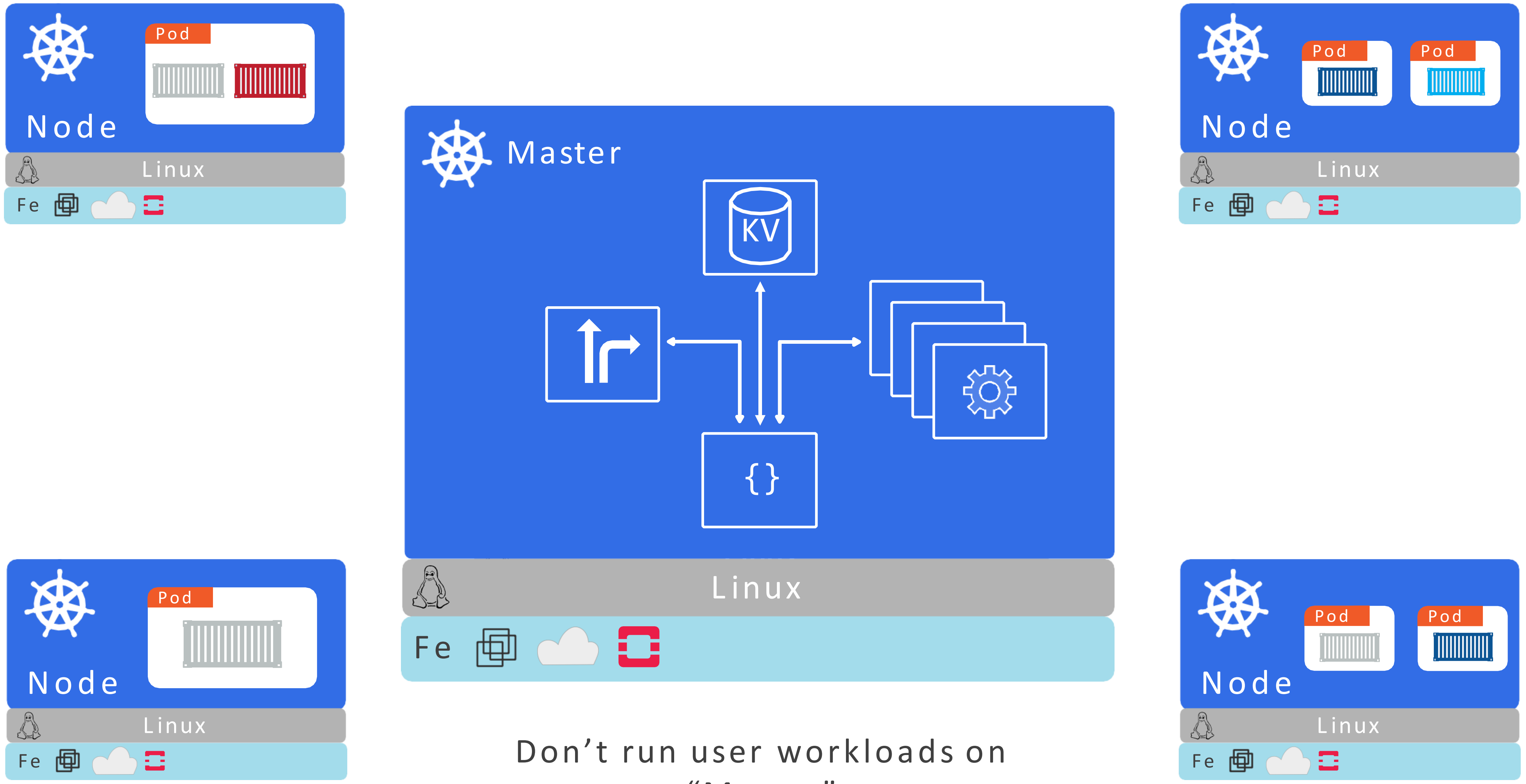
The Kubernetes Control Plane


Master
Linux
Fe
Node
Linux
Fe
Node
Linux
Fe
Node
Linux
Fe
...

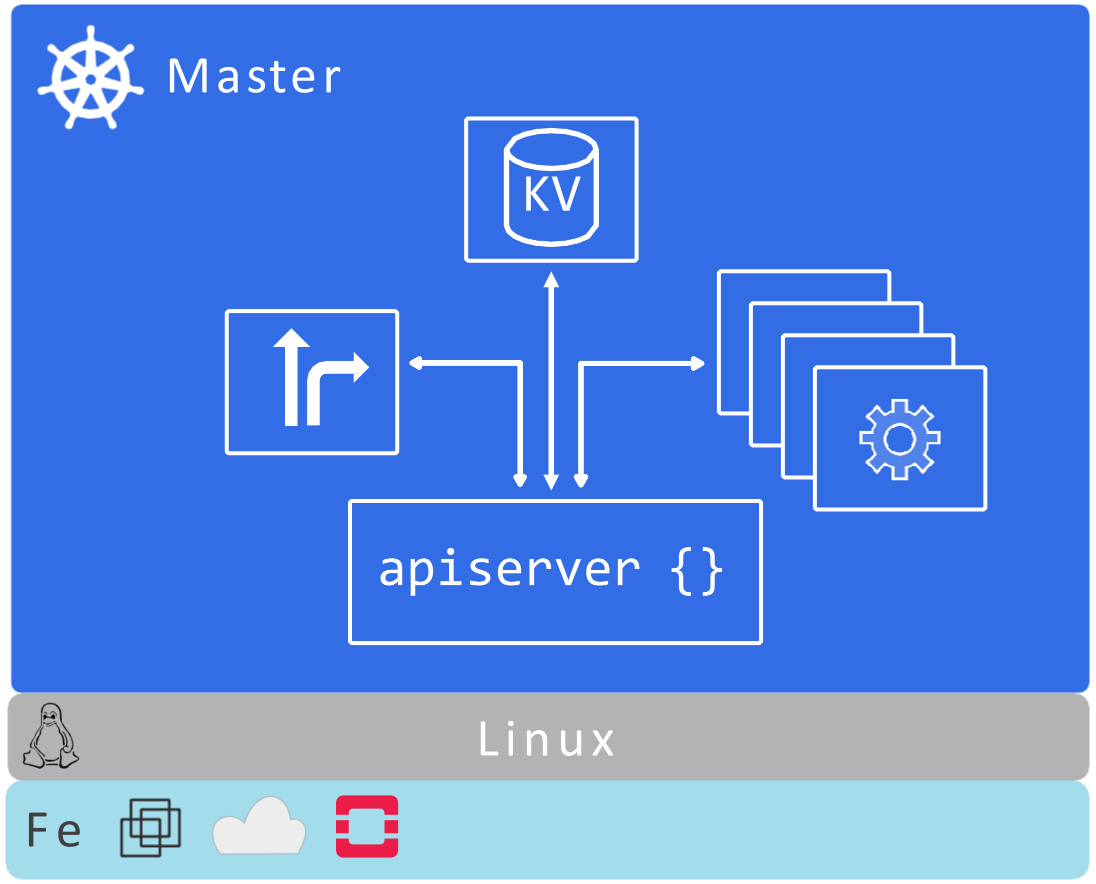

Master
KV
{ }
Linux
Fe

Multi-master HA





Don’t run user workloads on “Master”


Master
KV
apiserver {}
Linux
Fe

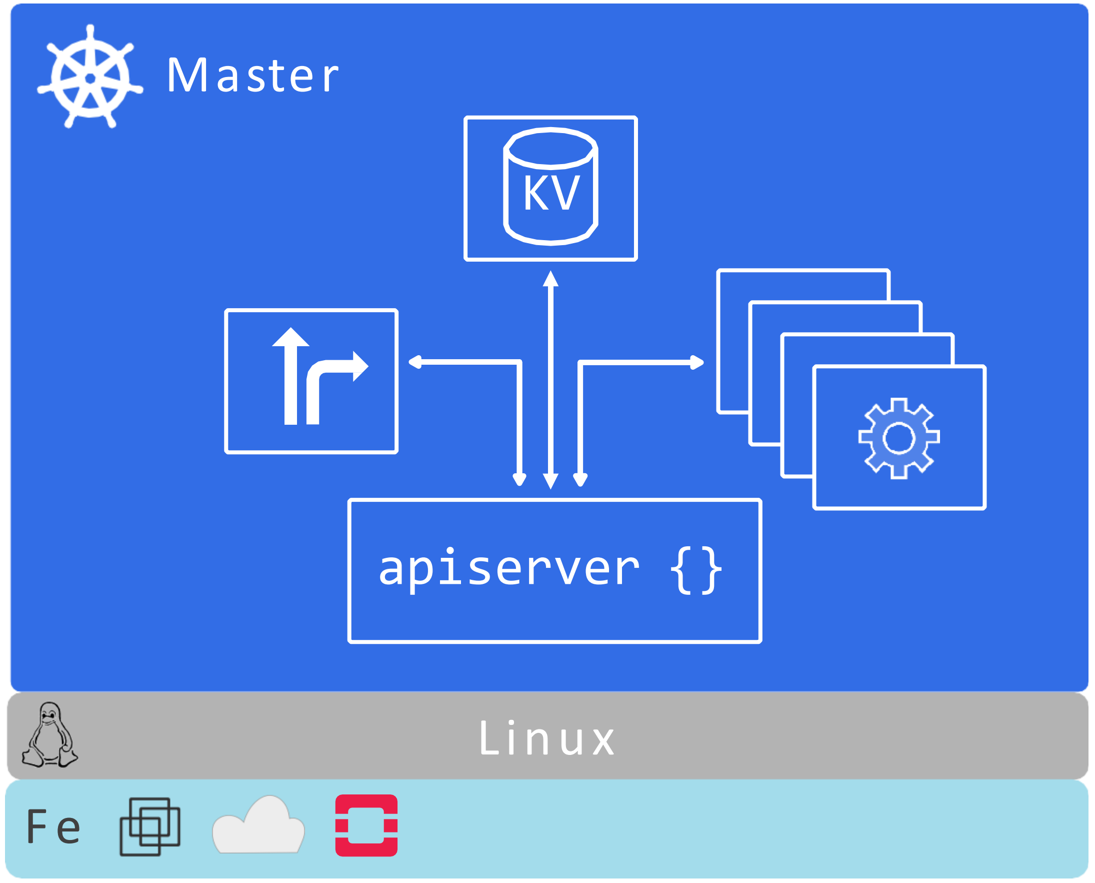
kube-apiserver

Front-end to the control plane

Exposes the API (REST)

Consumes JSON

(via manifest files)



Cluster store

Persistent storage

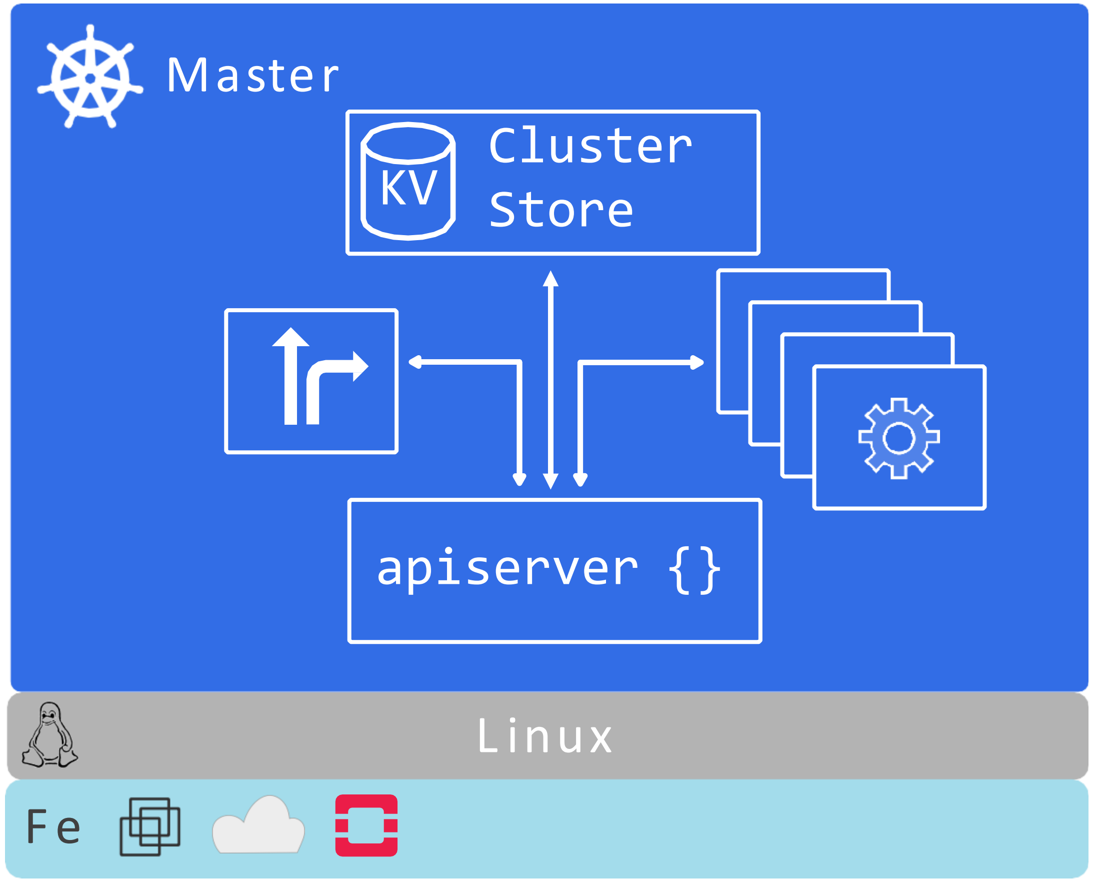
Cluster state and config

Uses etcd

Distributed, consistent, watchable...

The *"source of truth"* for the cluster

Have a backup plan for it!



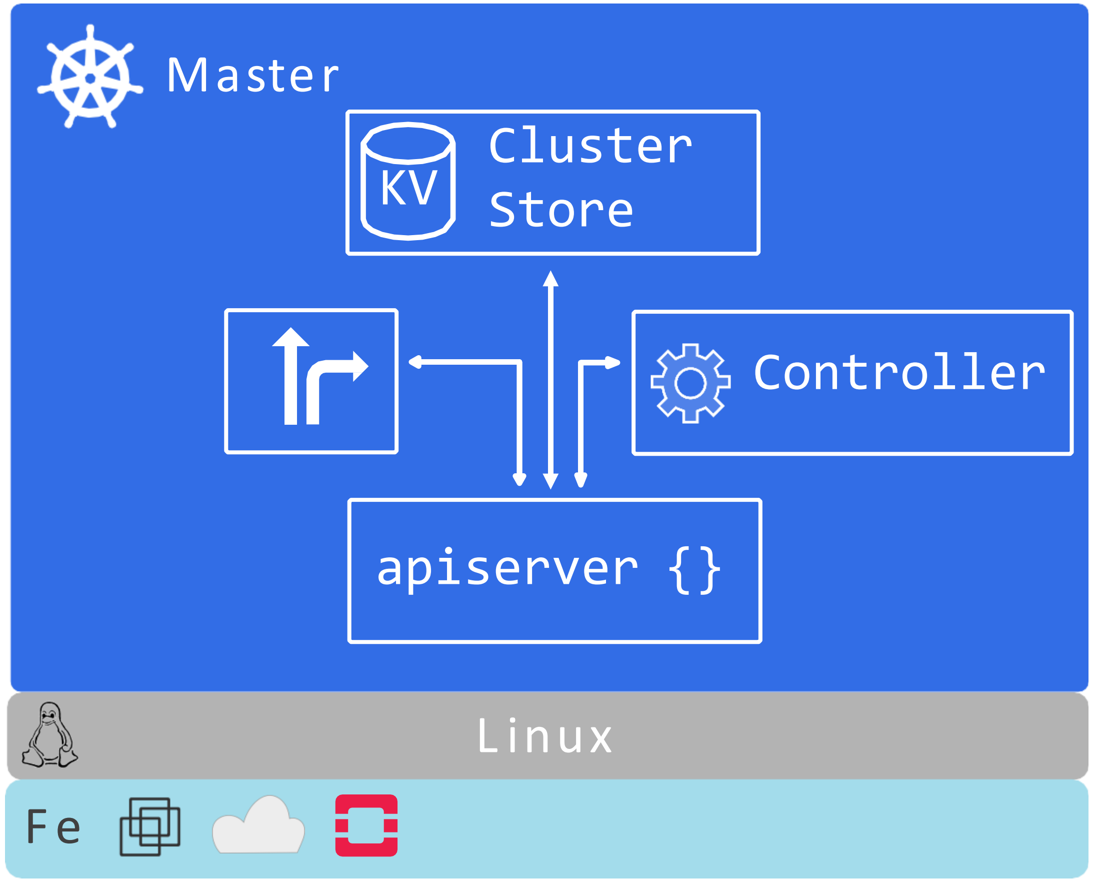
kube-controller-manager

Controller of controllers

- Node controller
- Endpoints controller
- Namespace controller
- …

Watches for changes

Helps maintain *desired state*

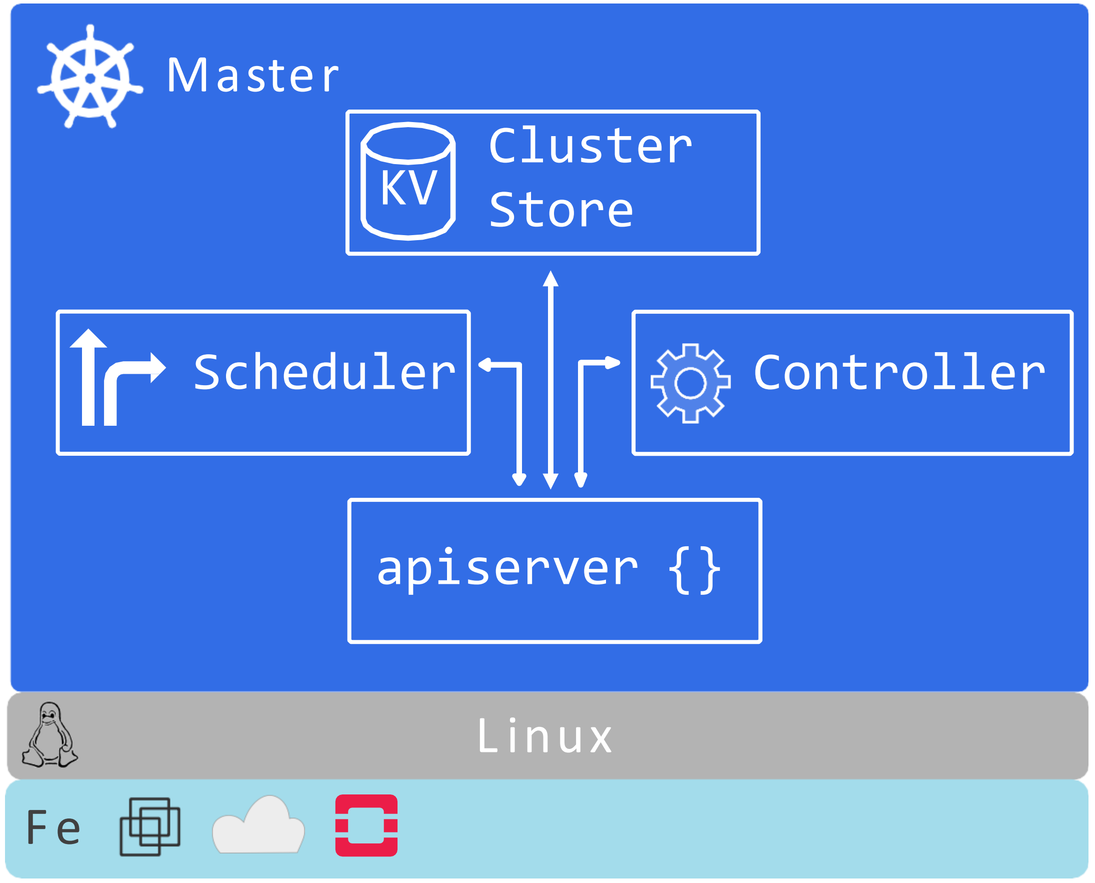


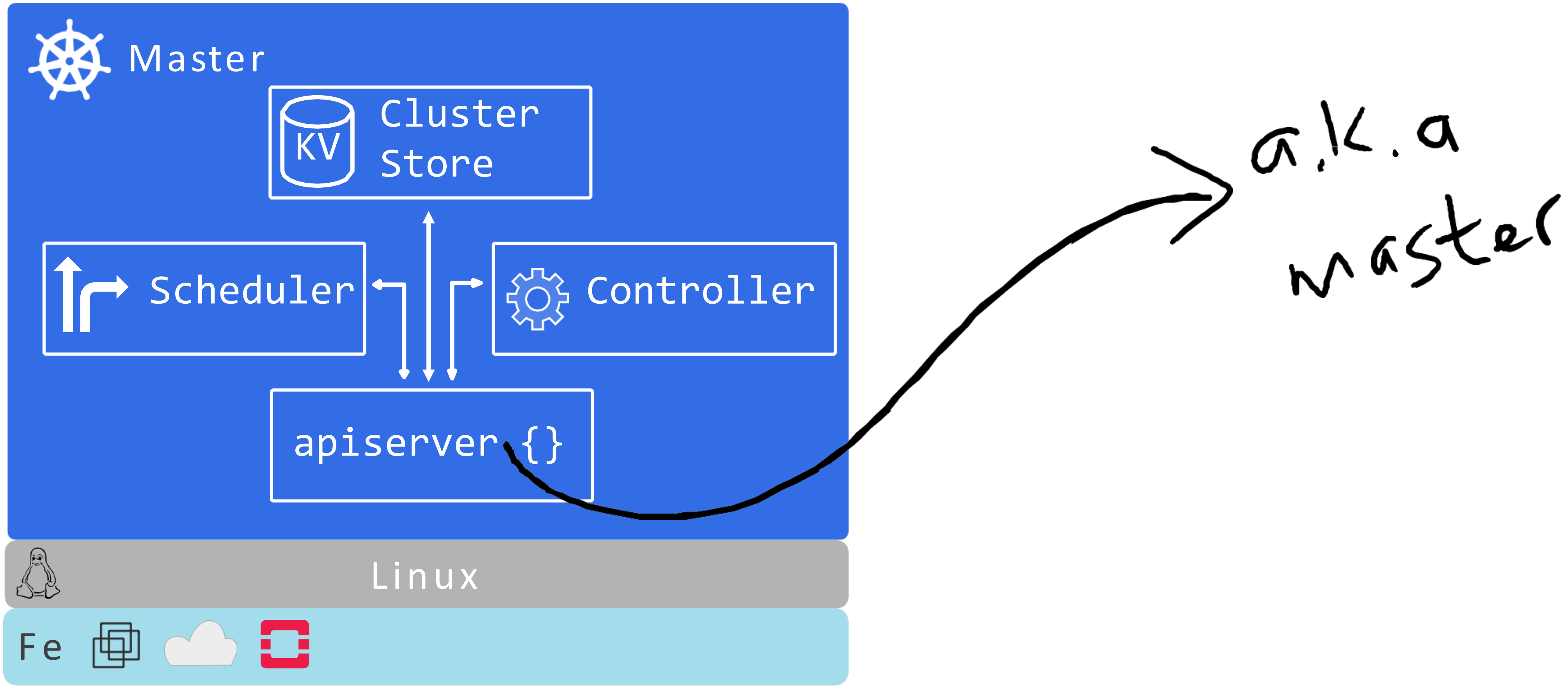
kube-scheduler

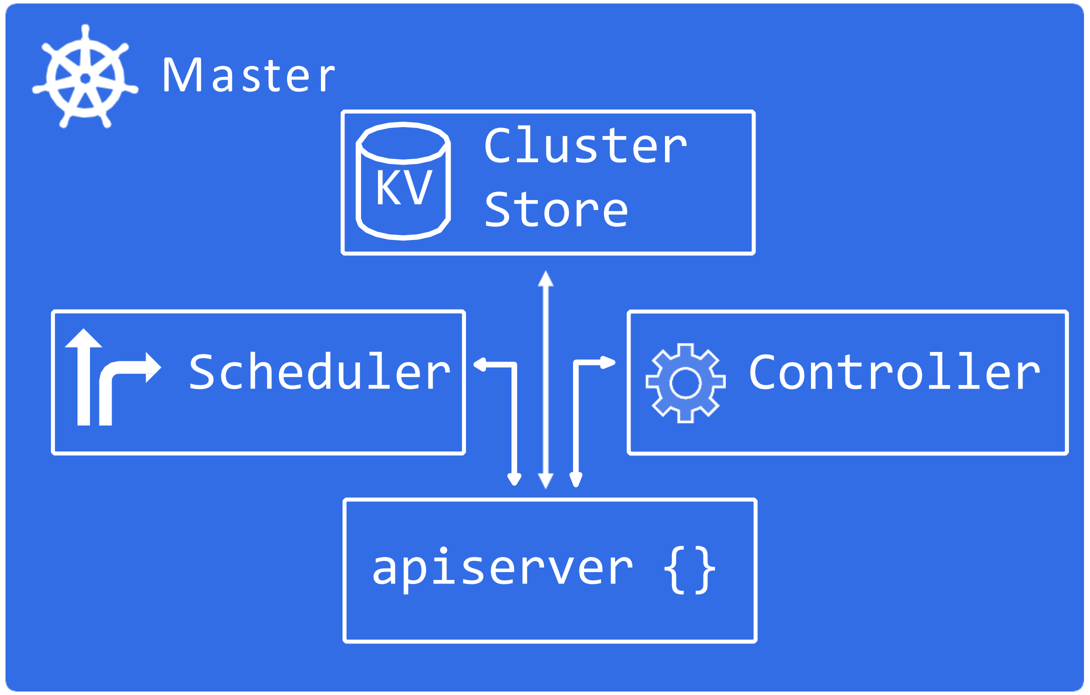
Watches apiserver for new pods

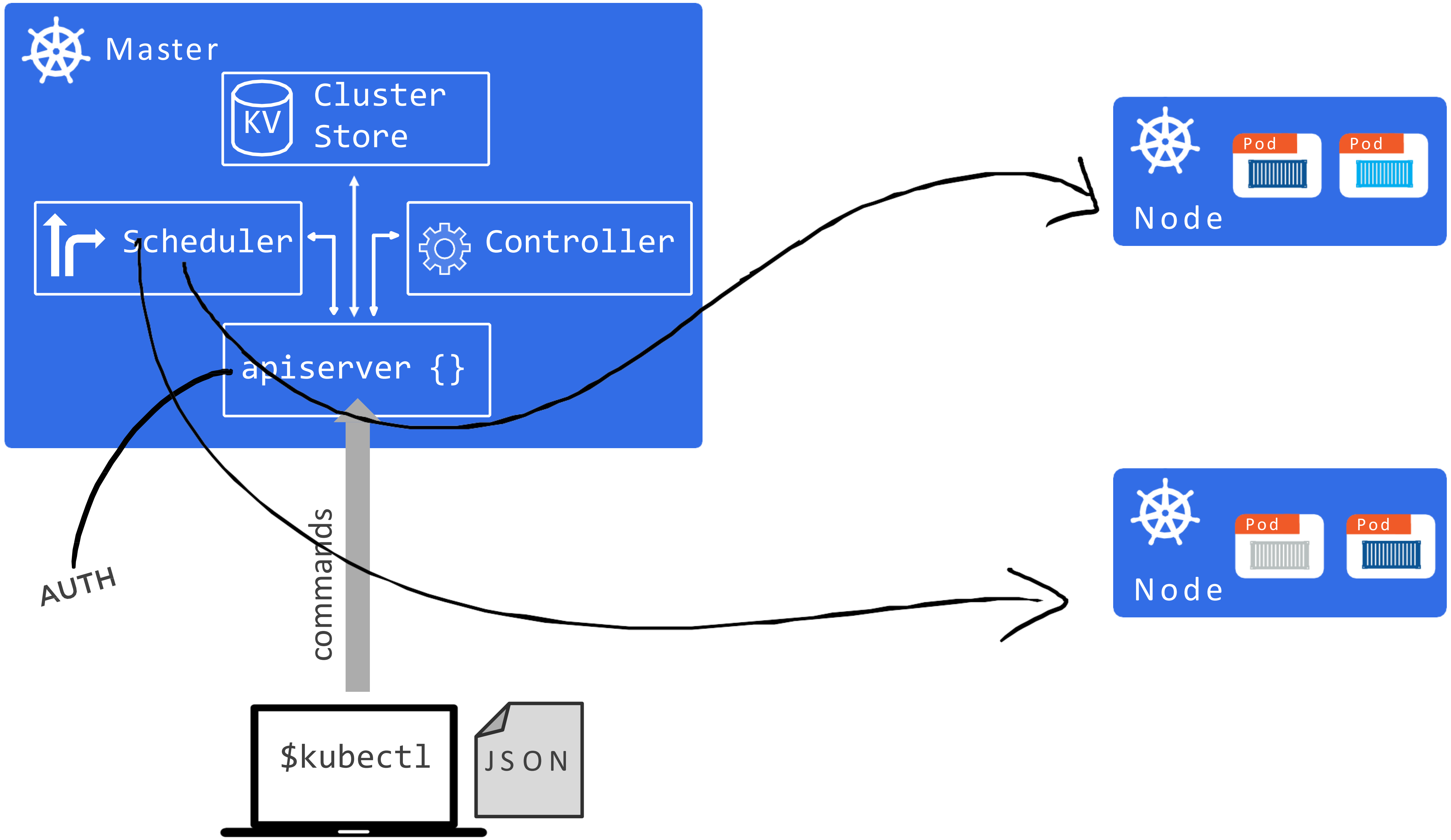
Assigns work to nodes

- affinity/anti-affinity
- constraints
- resources
- …



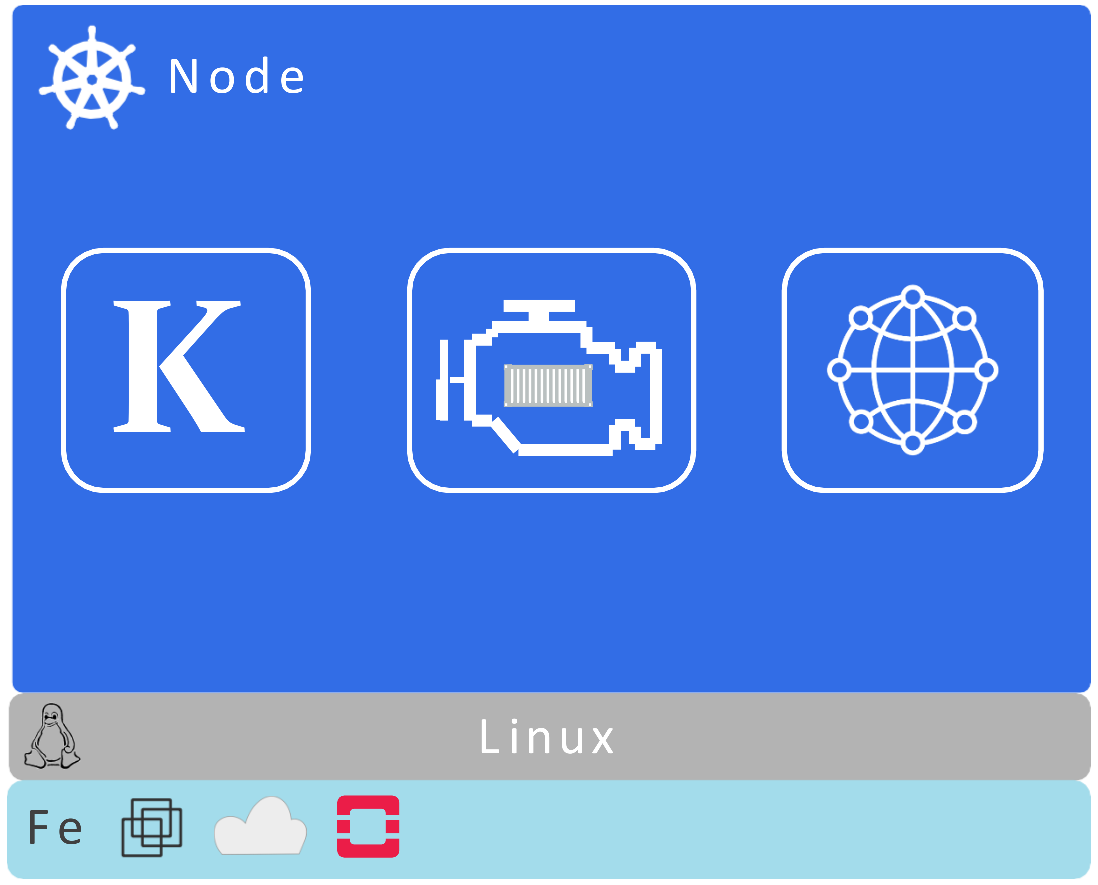

Master
KV
Cluster Store
Scheduler
Controller
apiserver {}
Linux
Fe
a.k.a master


Master
KV
Cluster
Store
Scheduler
Controller
apiserver {}


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

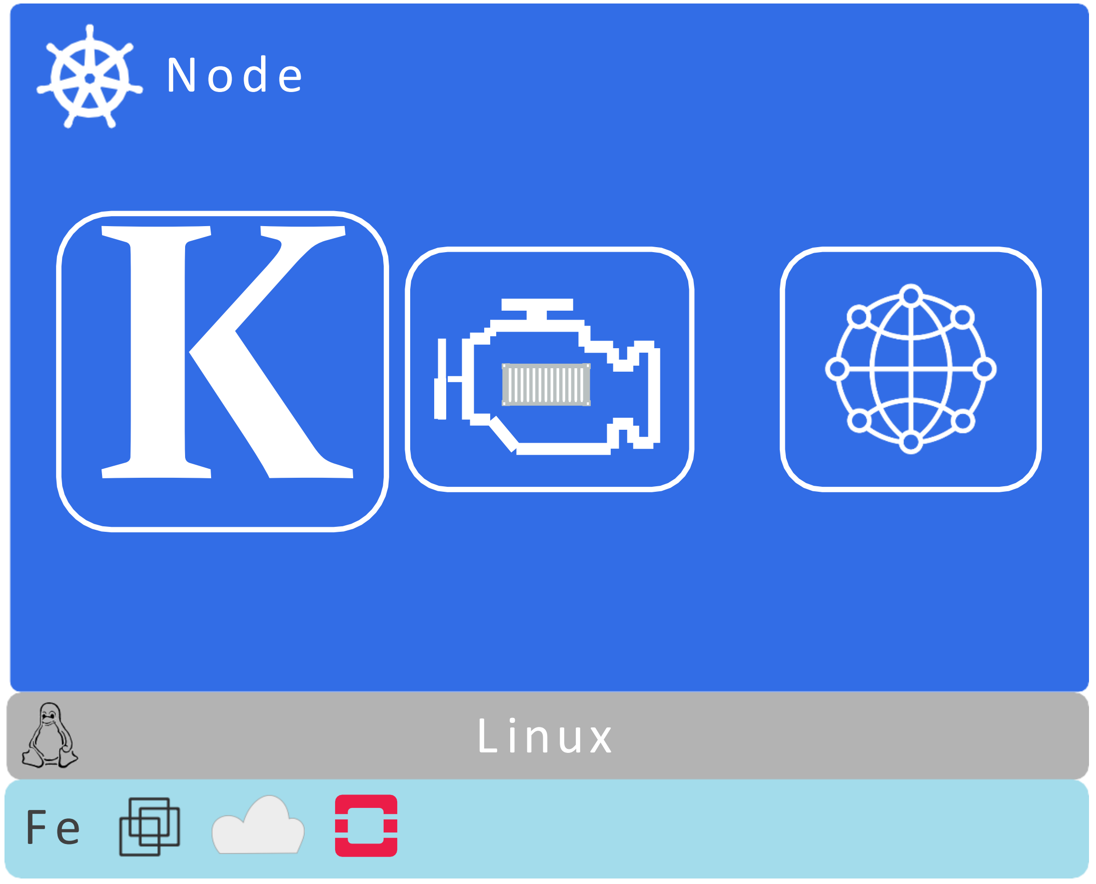
Nodes a.k.a “Minions”

The Kubernetes Worker


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



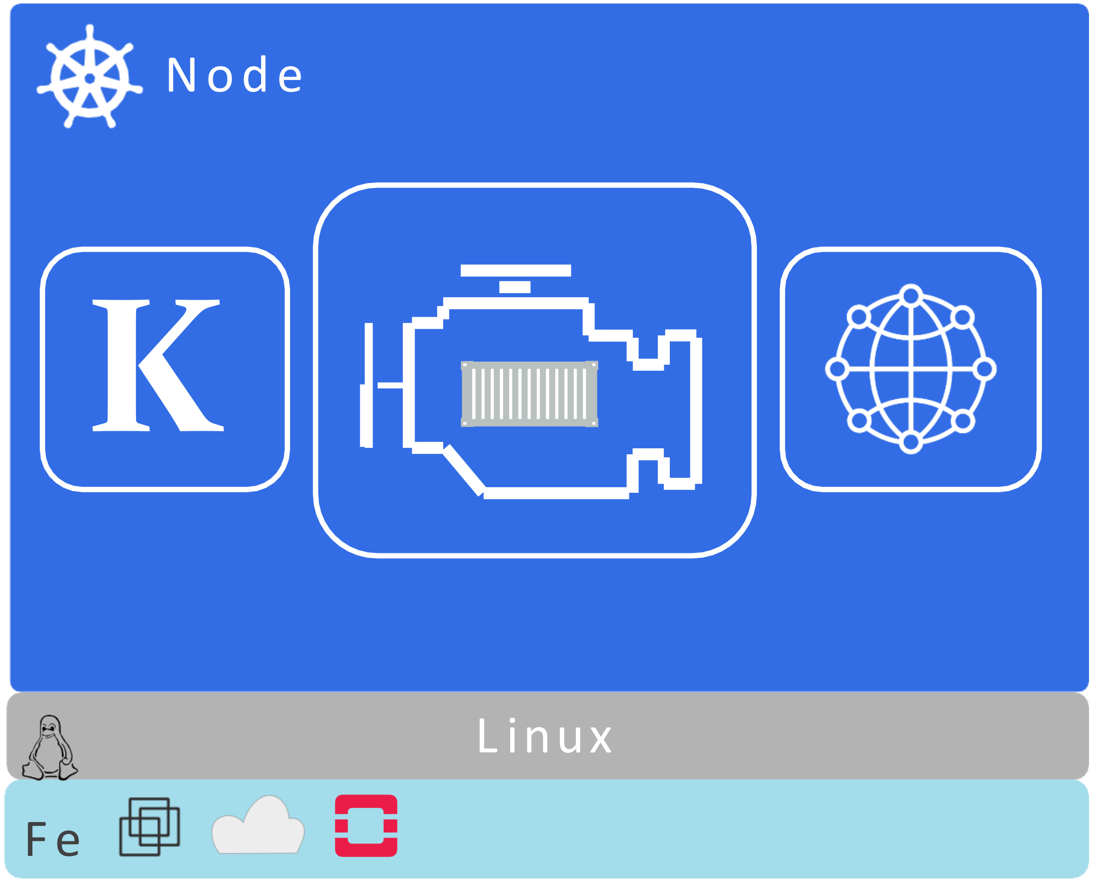
Container Engine

Does container management:

- Pulling images
- Starting/stopping containers
- …

Pluggable:

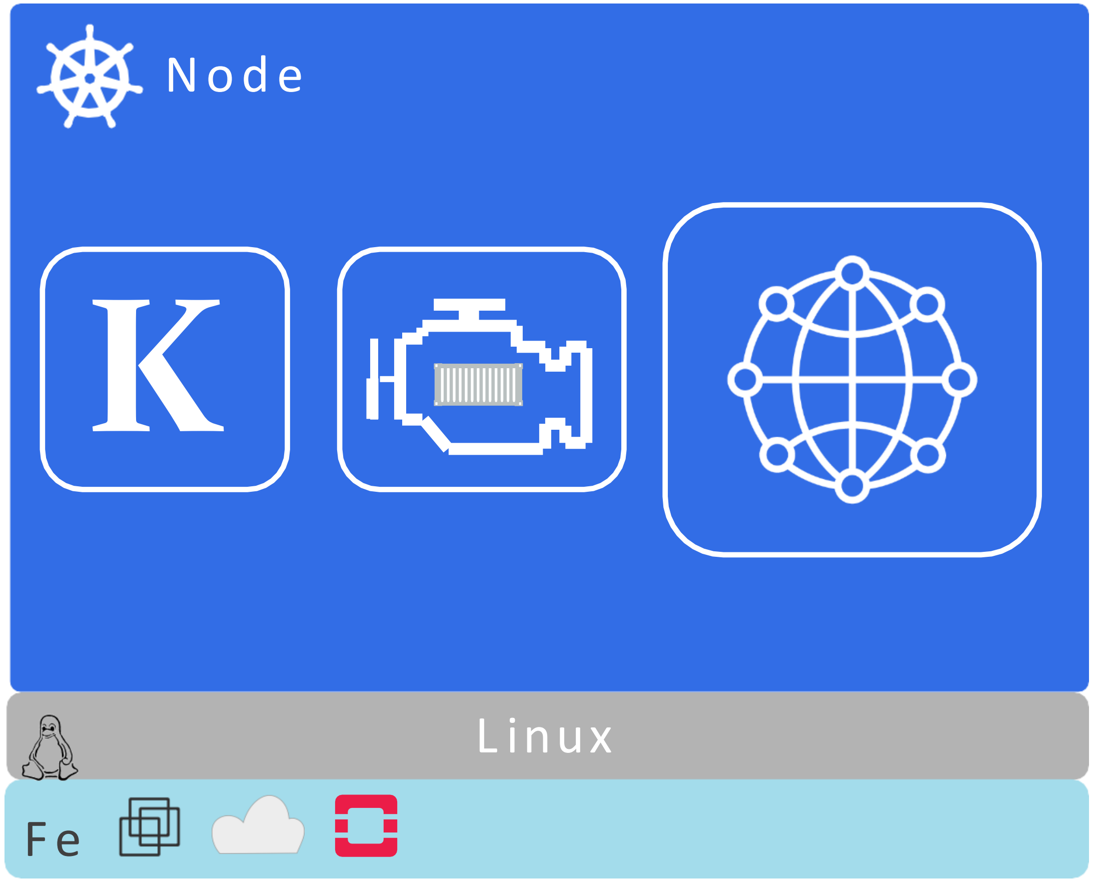
- Usually Docker
- Can be rkt

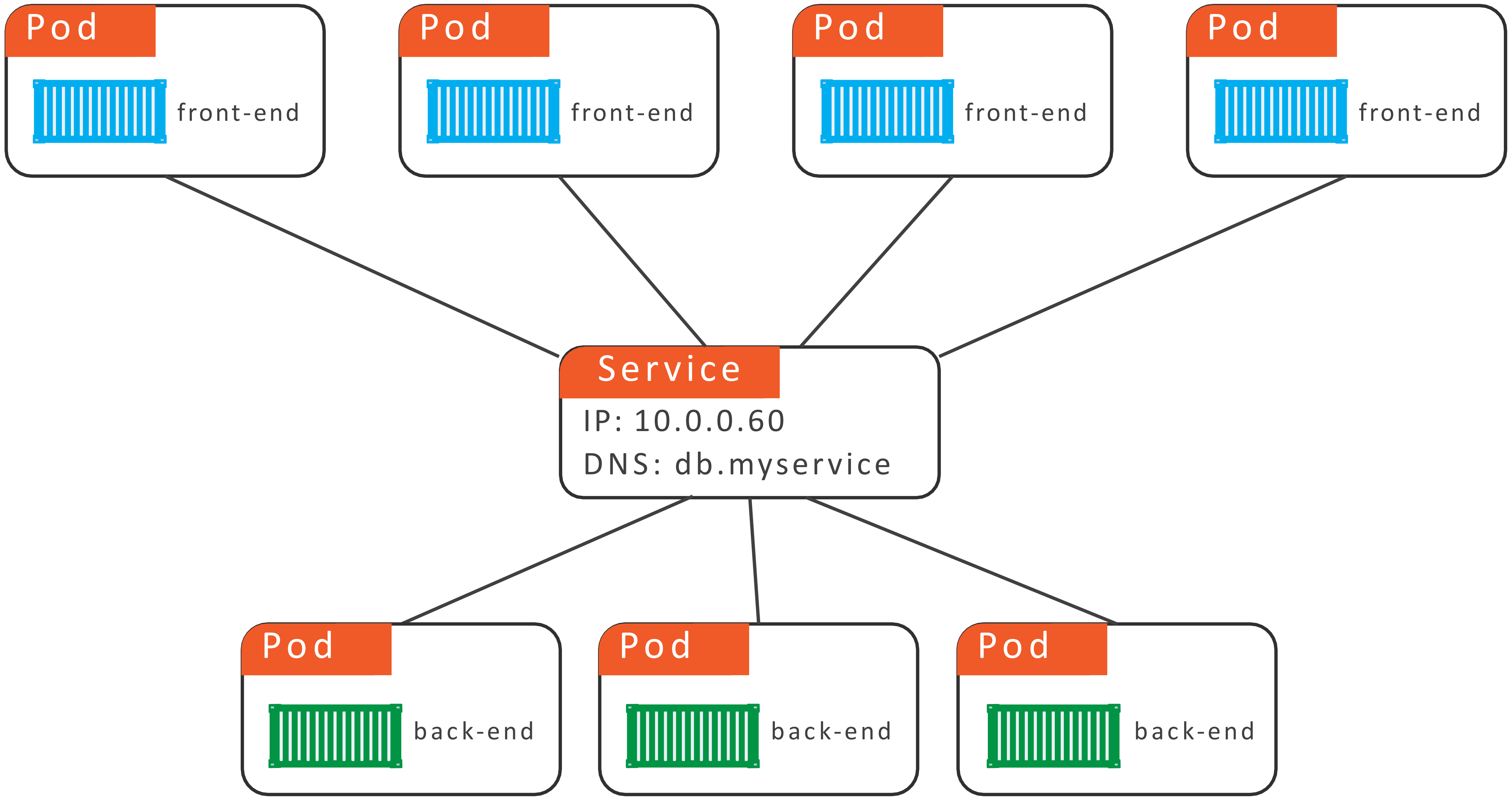


kube-proxy

Kubernetes networking:

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




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

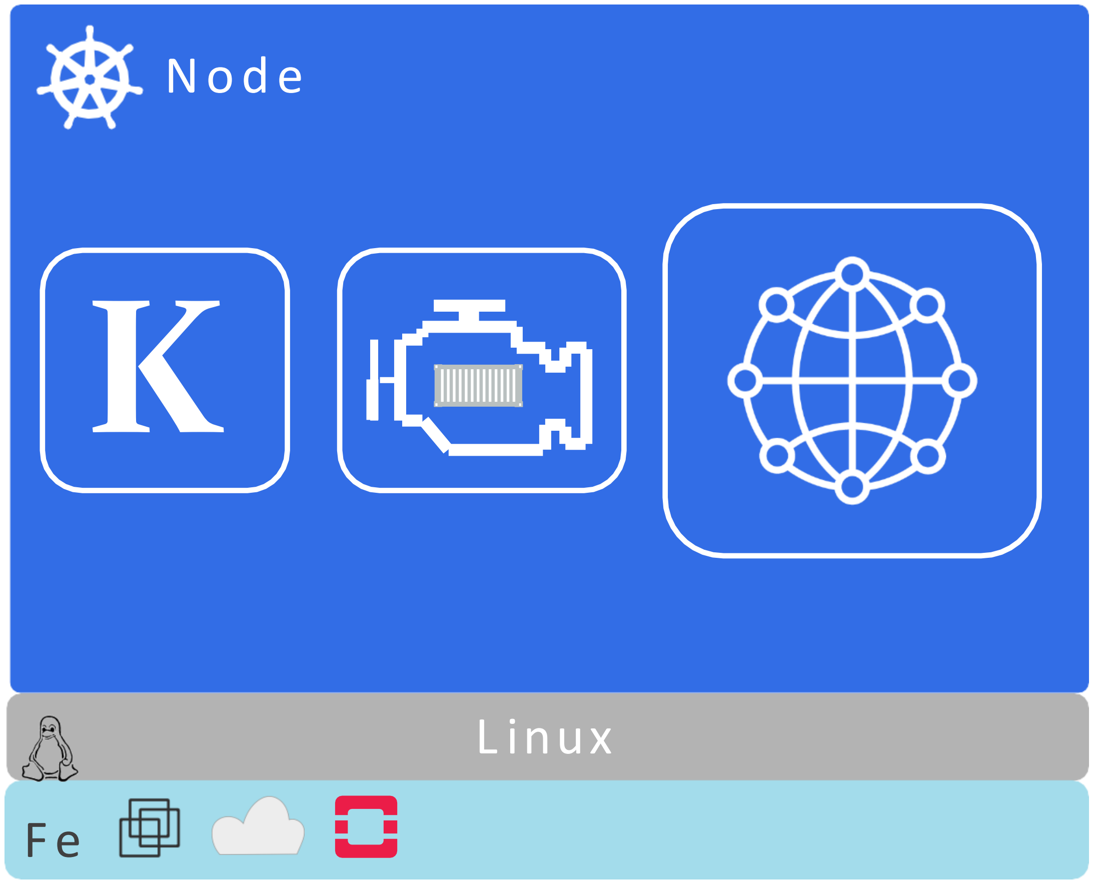
Main Kubernetes agent

Container engine

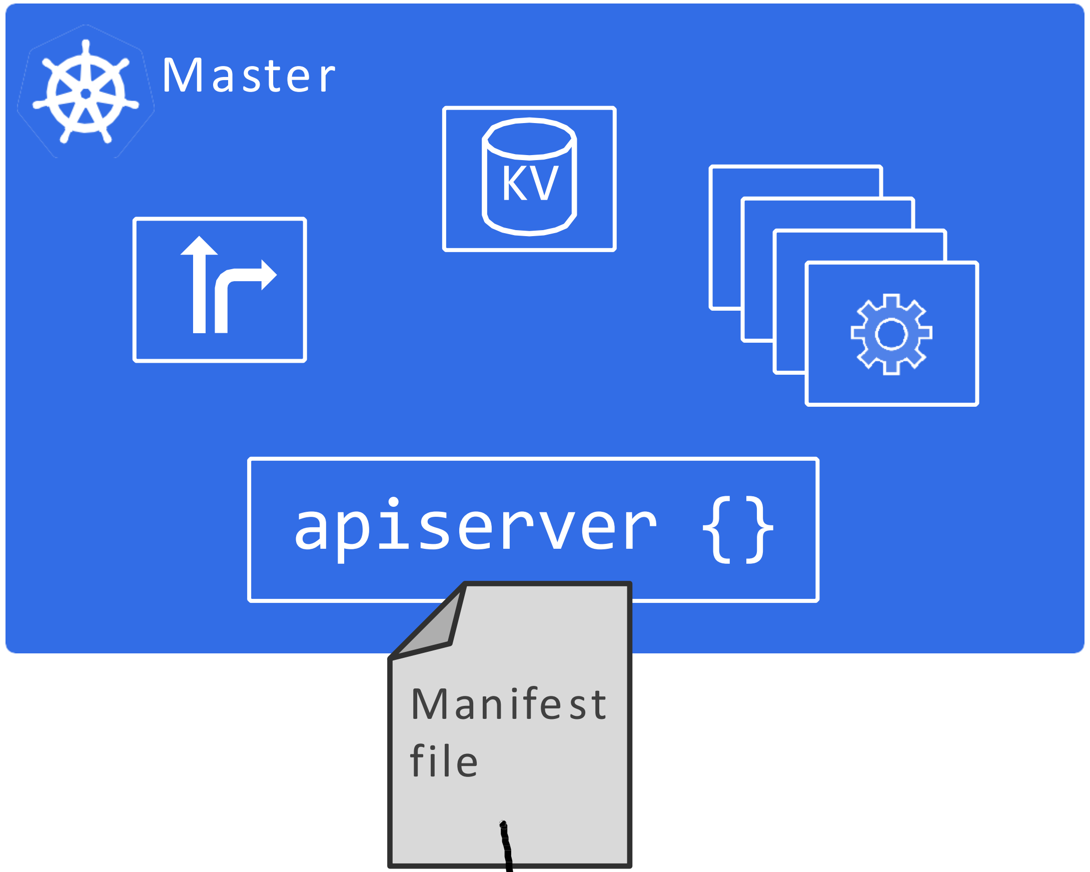
Docker or rkt

kube-proxy

Kubernetes networking

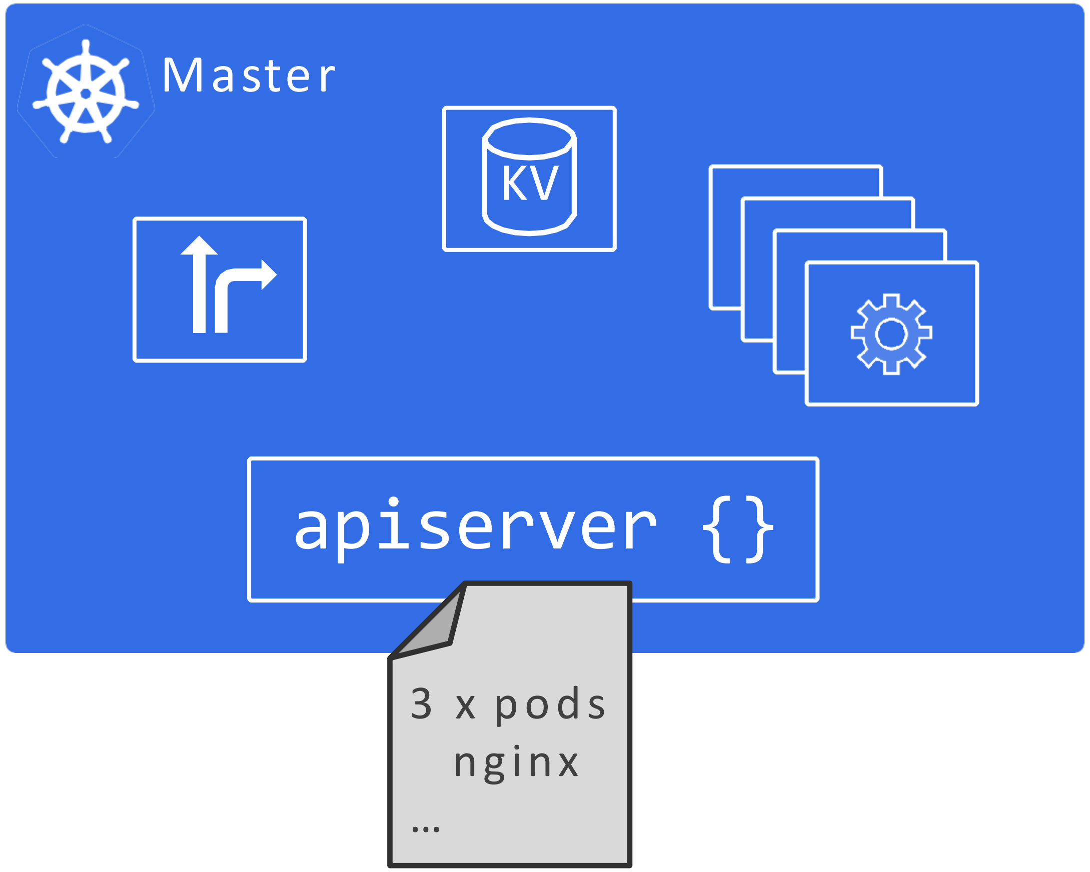


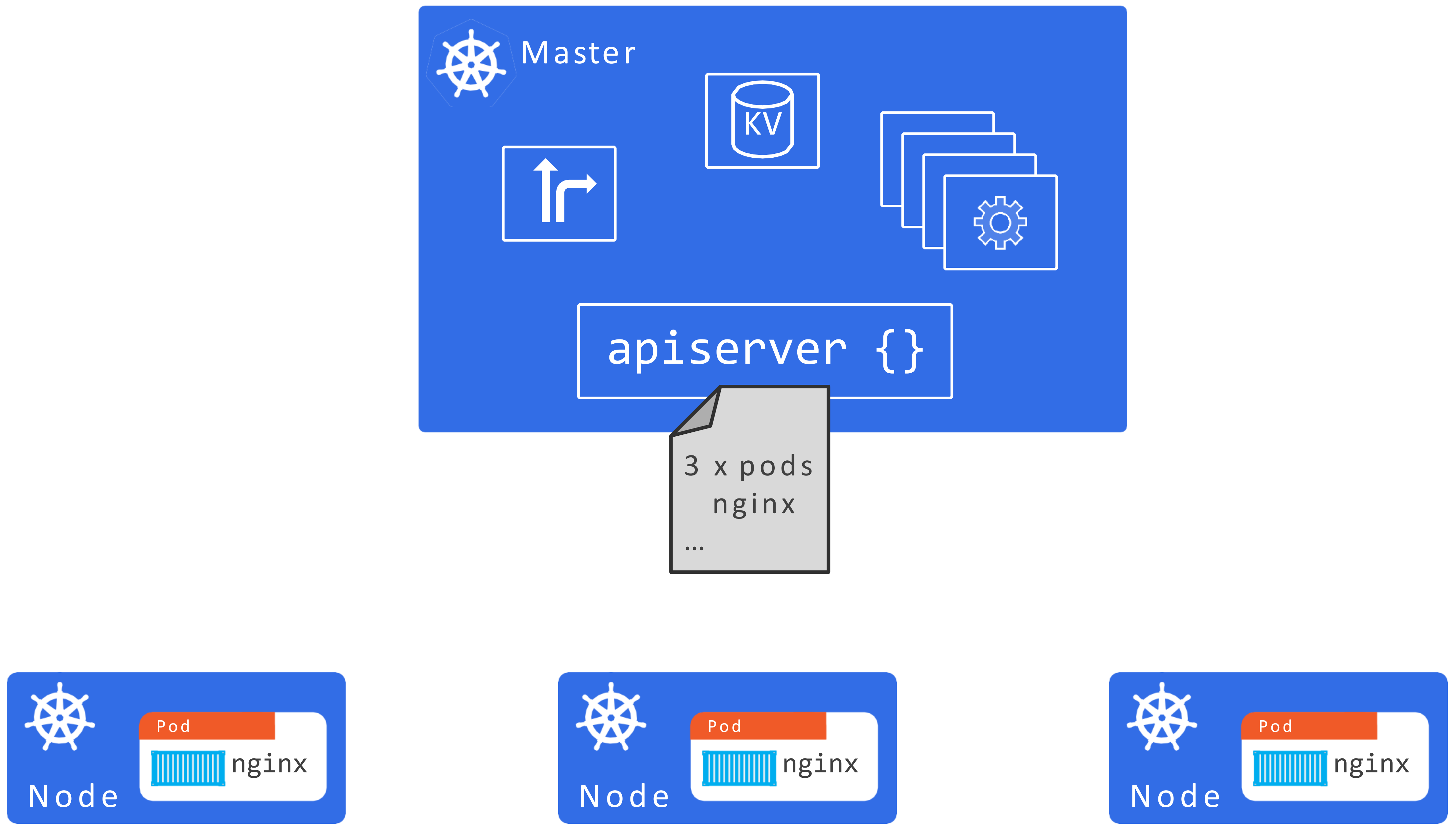
Declarative Model
&
Desired State

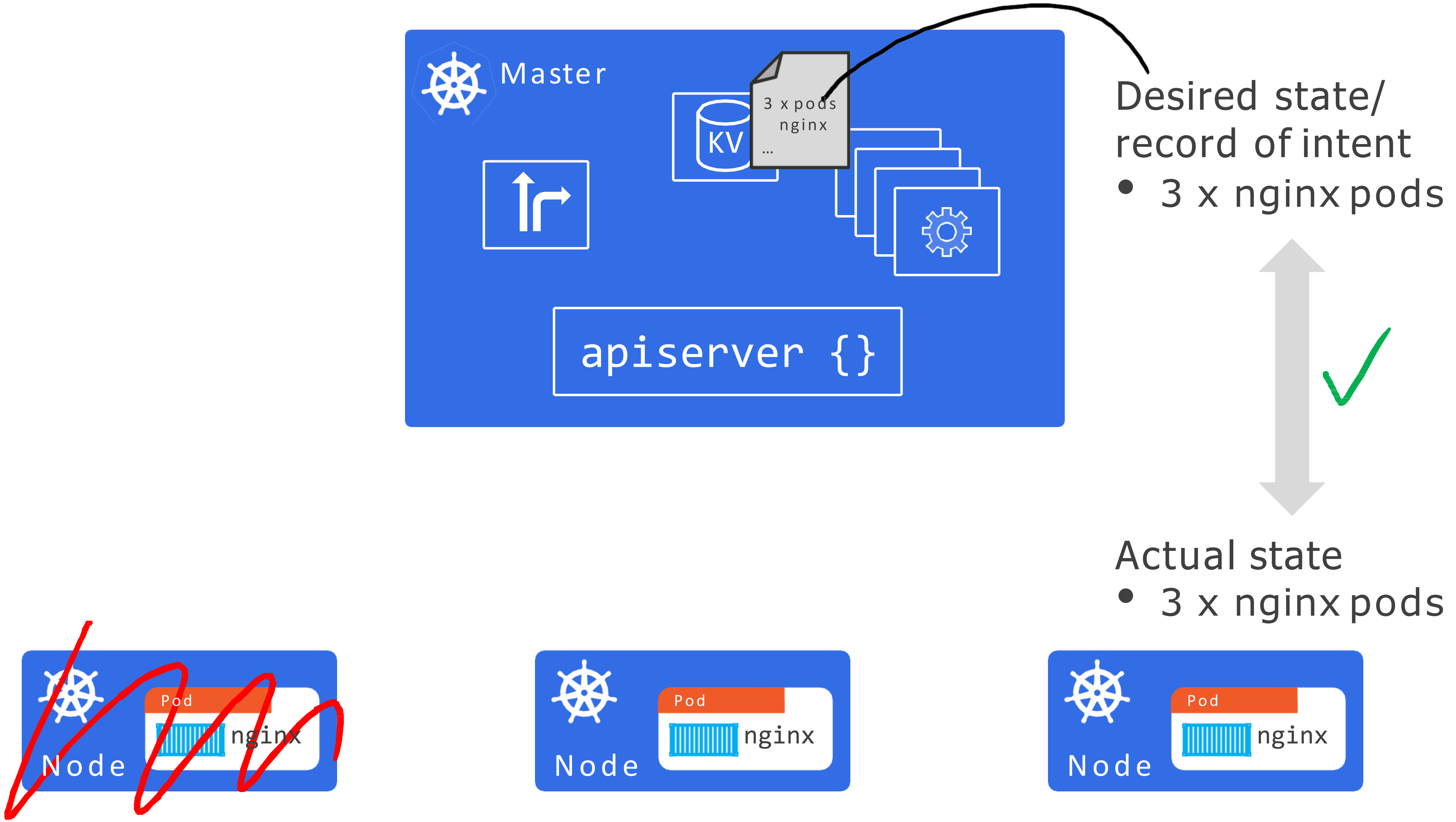


YAML or JSON

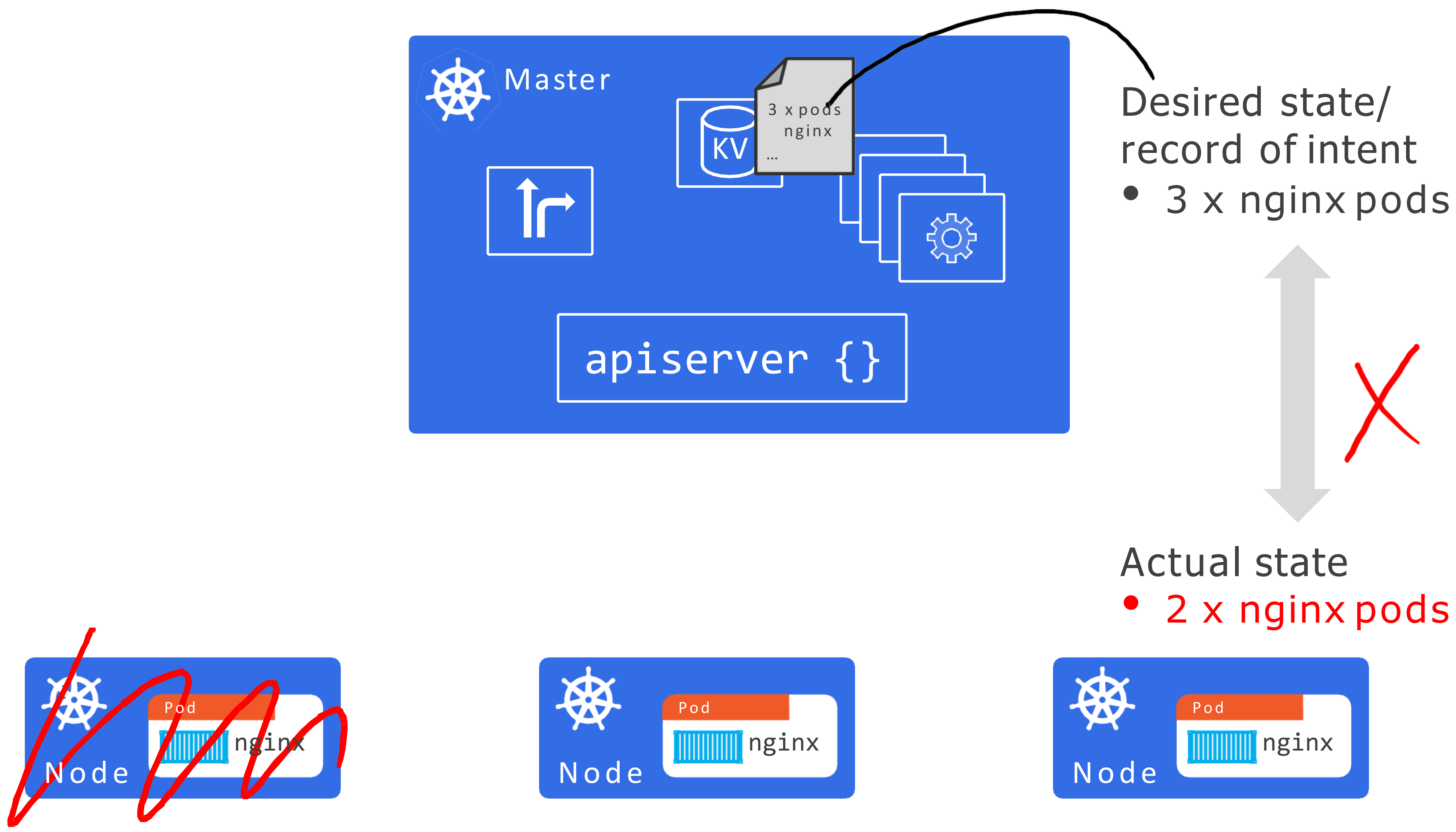
Describe desired state


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


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



Master
3 x pods
nginx
...
KV
apiserver {}
Desired state/
record of intent
• 3 x nginx pods
Actual state
• 3 x nginx pods
Pod
nginx
Node
Pod
nginx
Node
Pod
nginx
Node

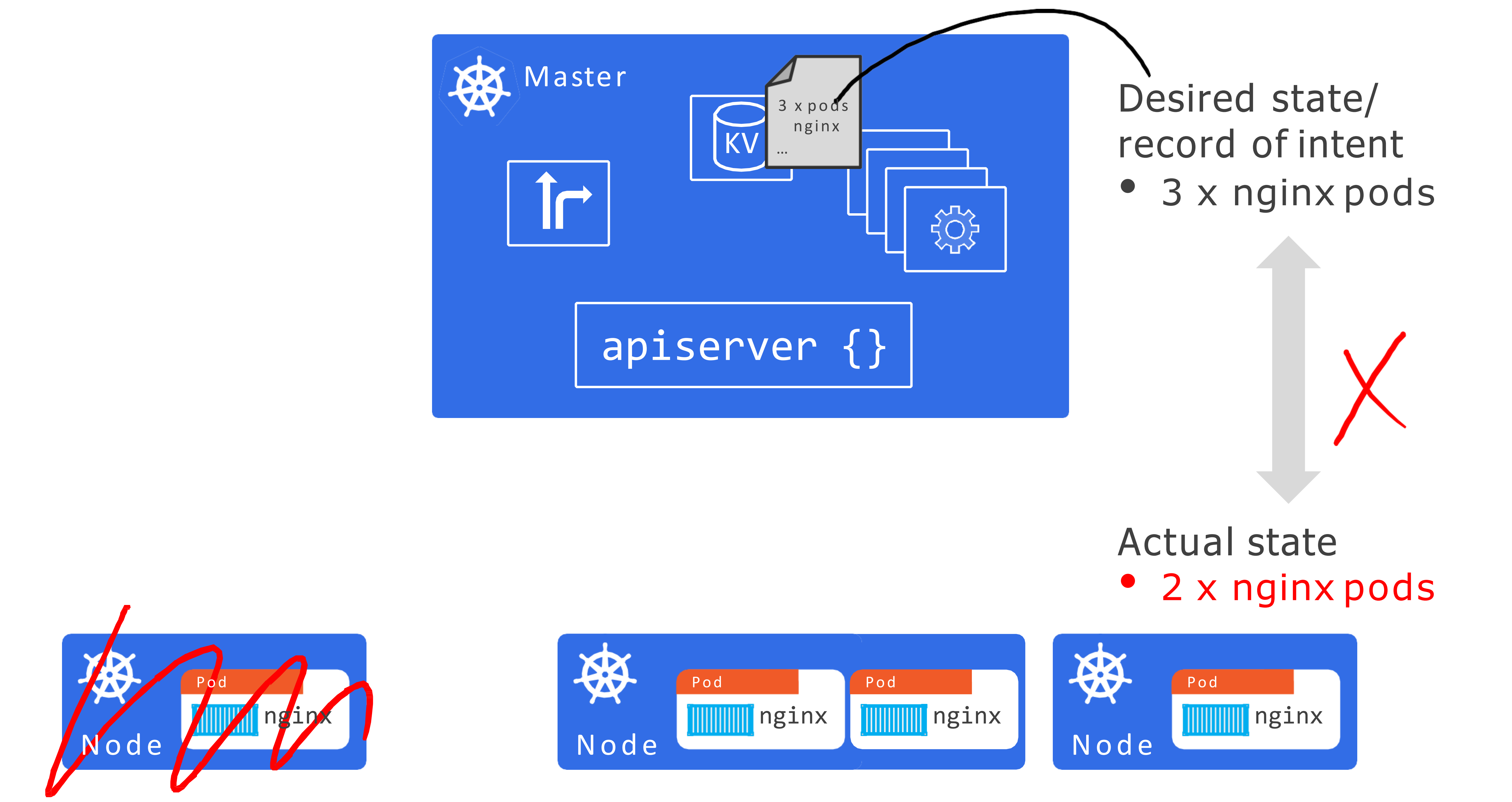


Desired state/
record of intent

- 3 x nginx pods

Actual state

- 2 x nginx pods

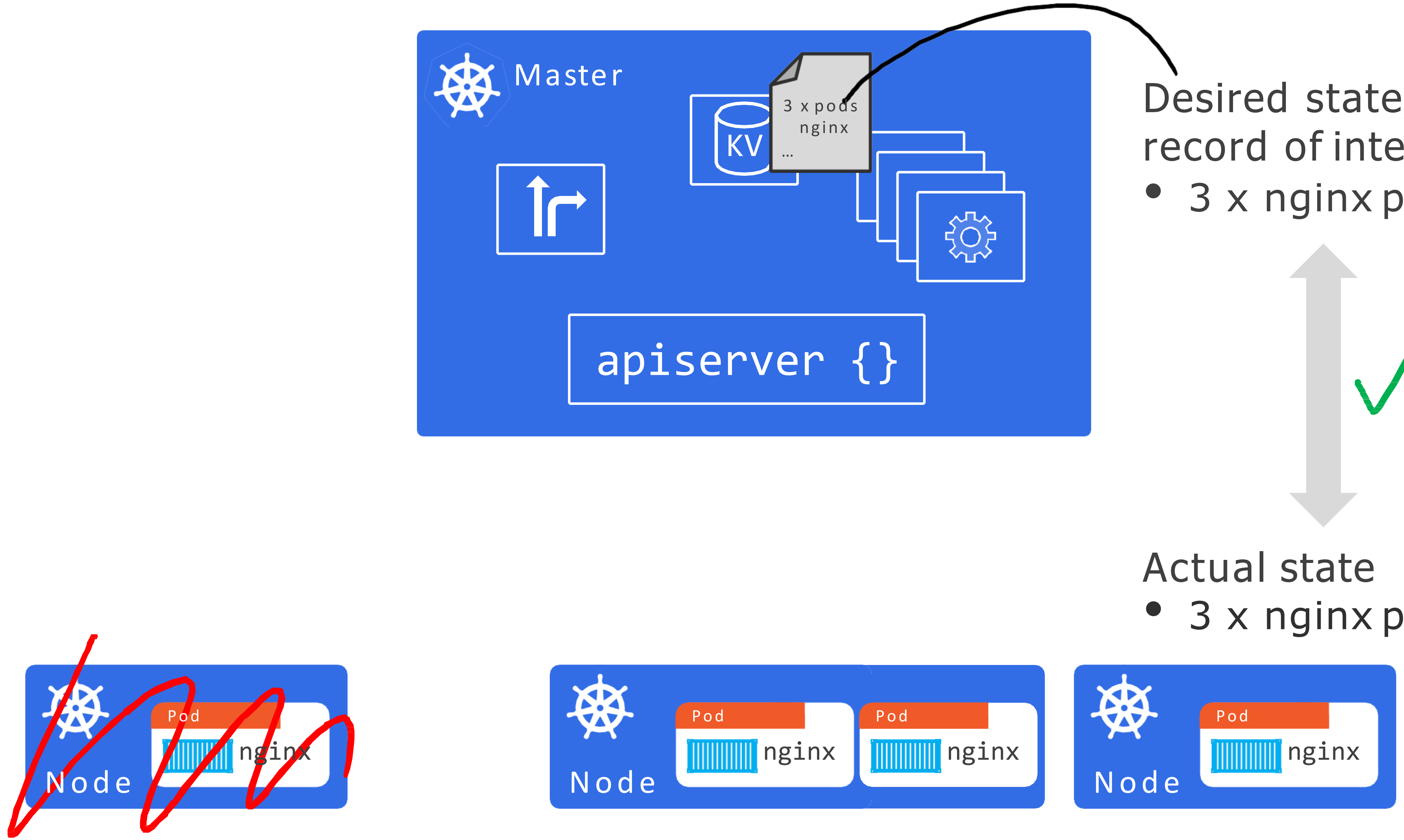


Desired state/
record of intent

- 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

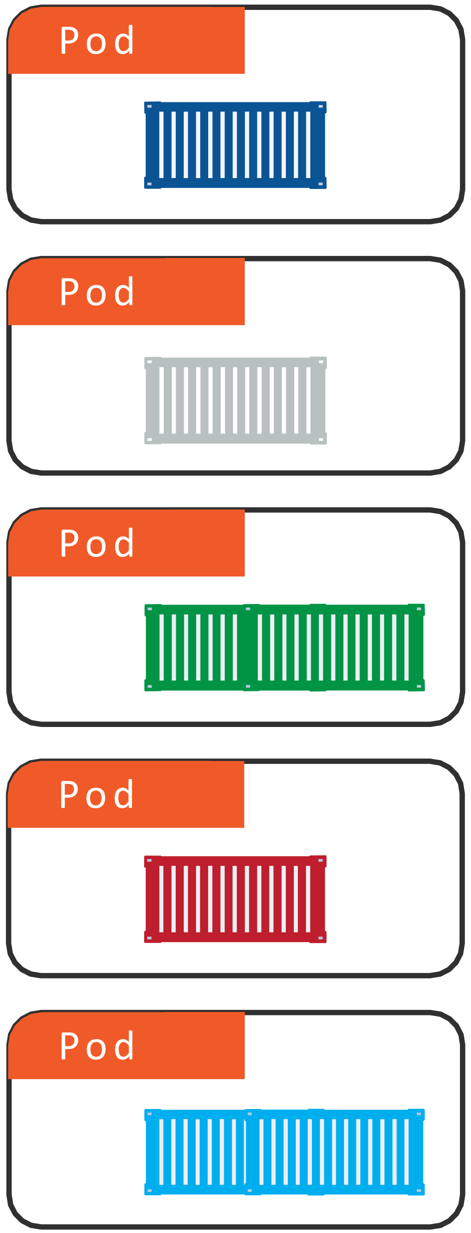
Container

Pod

Atomic units of scheduling

Containers always run inside of pods

Pods can have multiple containers
(advanced use-case)

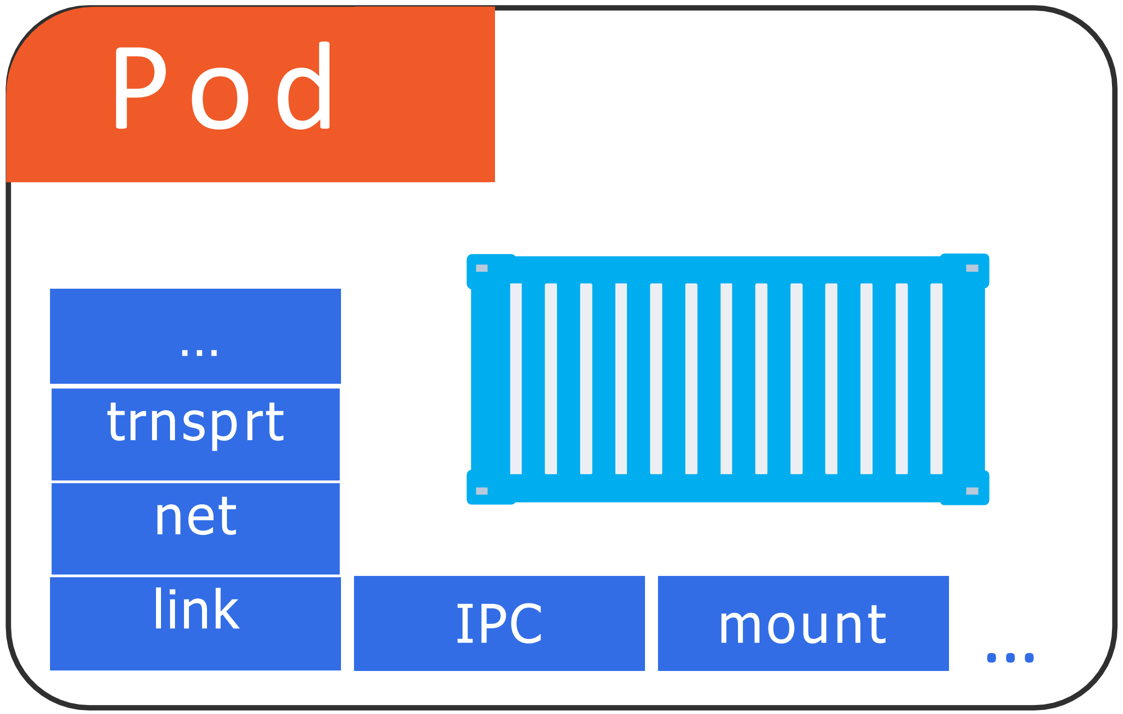


Ring-fenced environment

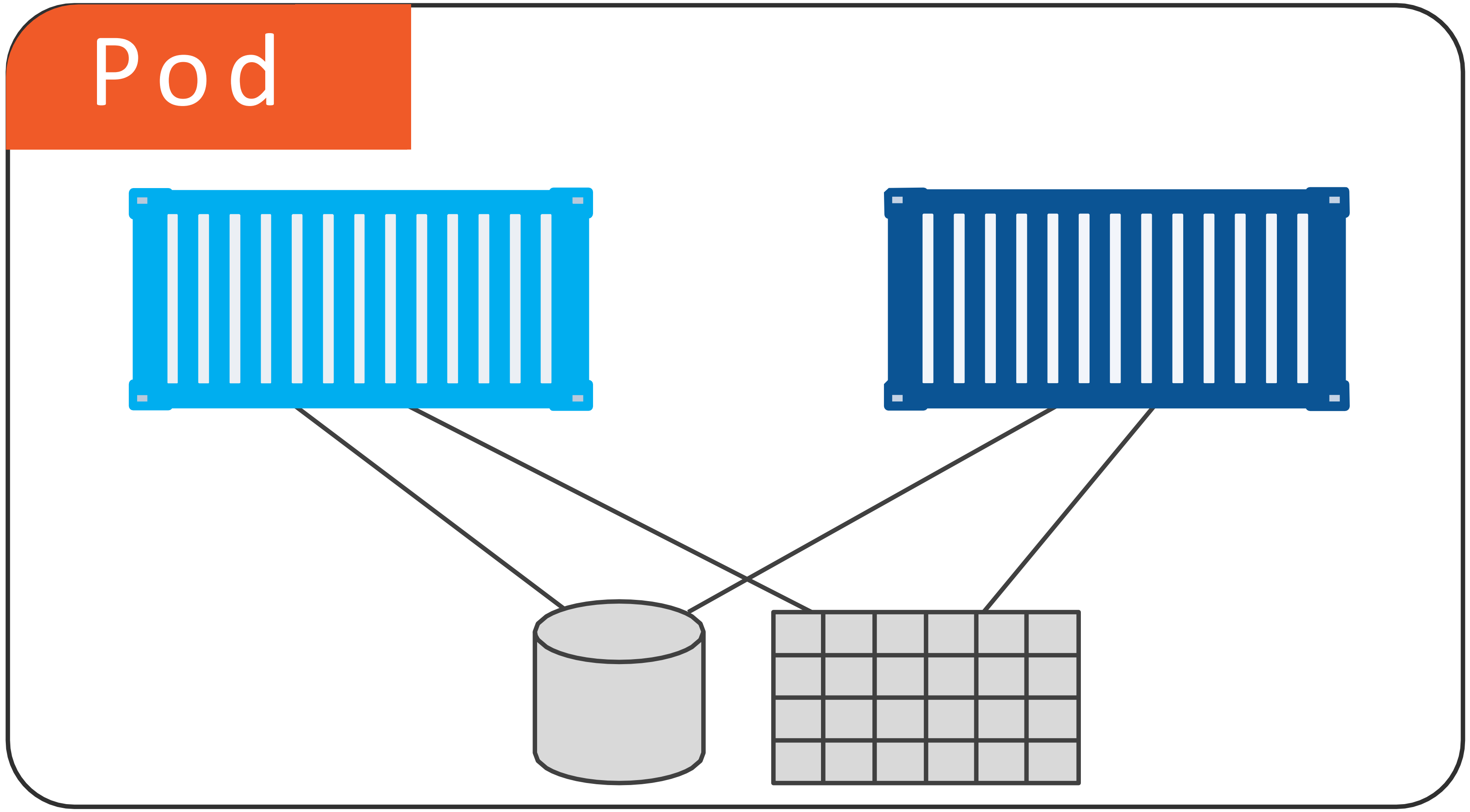
- Network stack
- Kernel namespaces
- ...

n containers

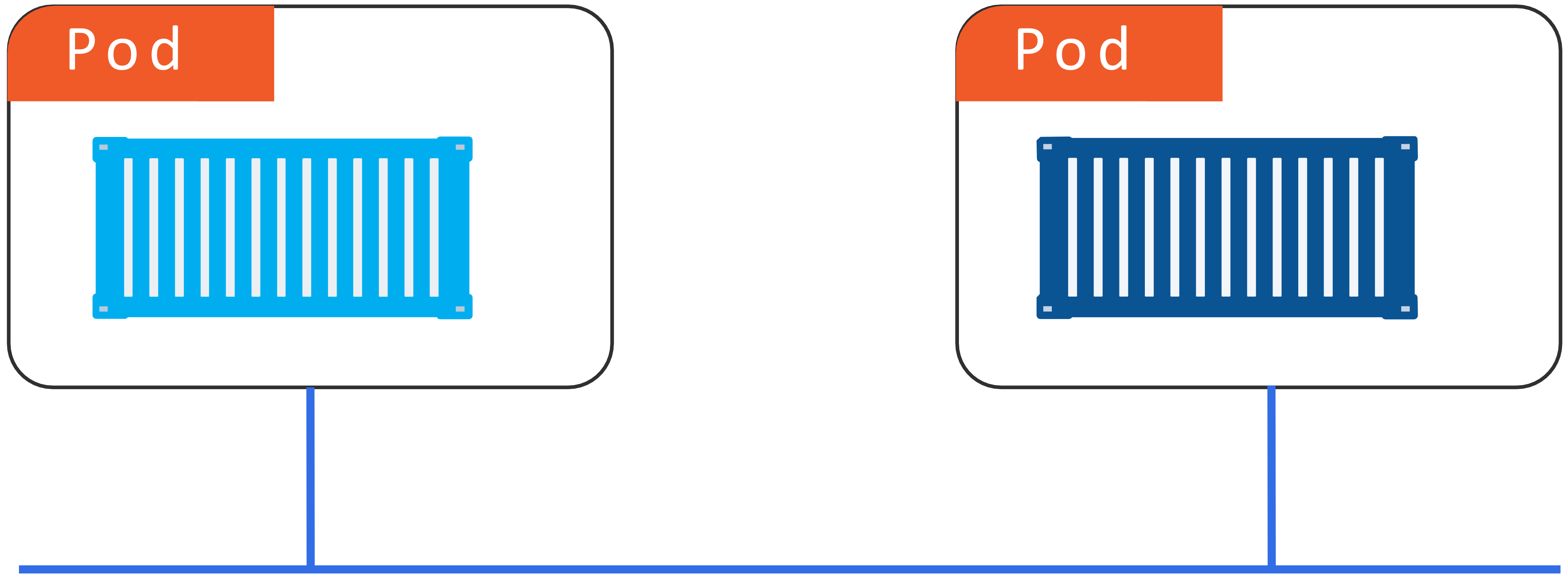
<u>All containers in pod share the pod environment</u>



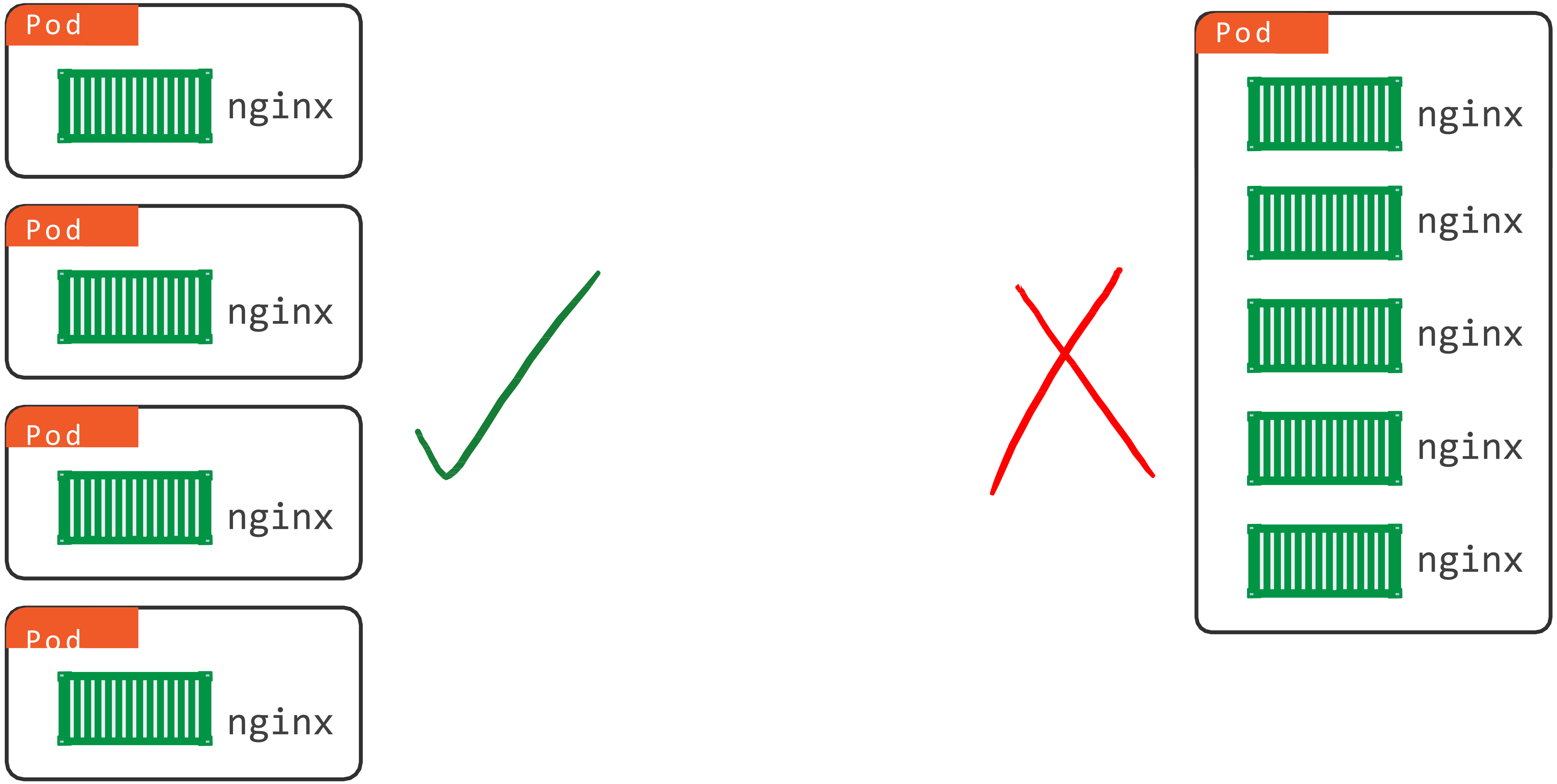
Tight Coupling



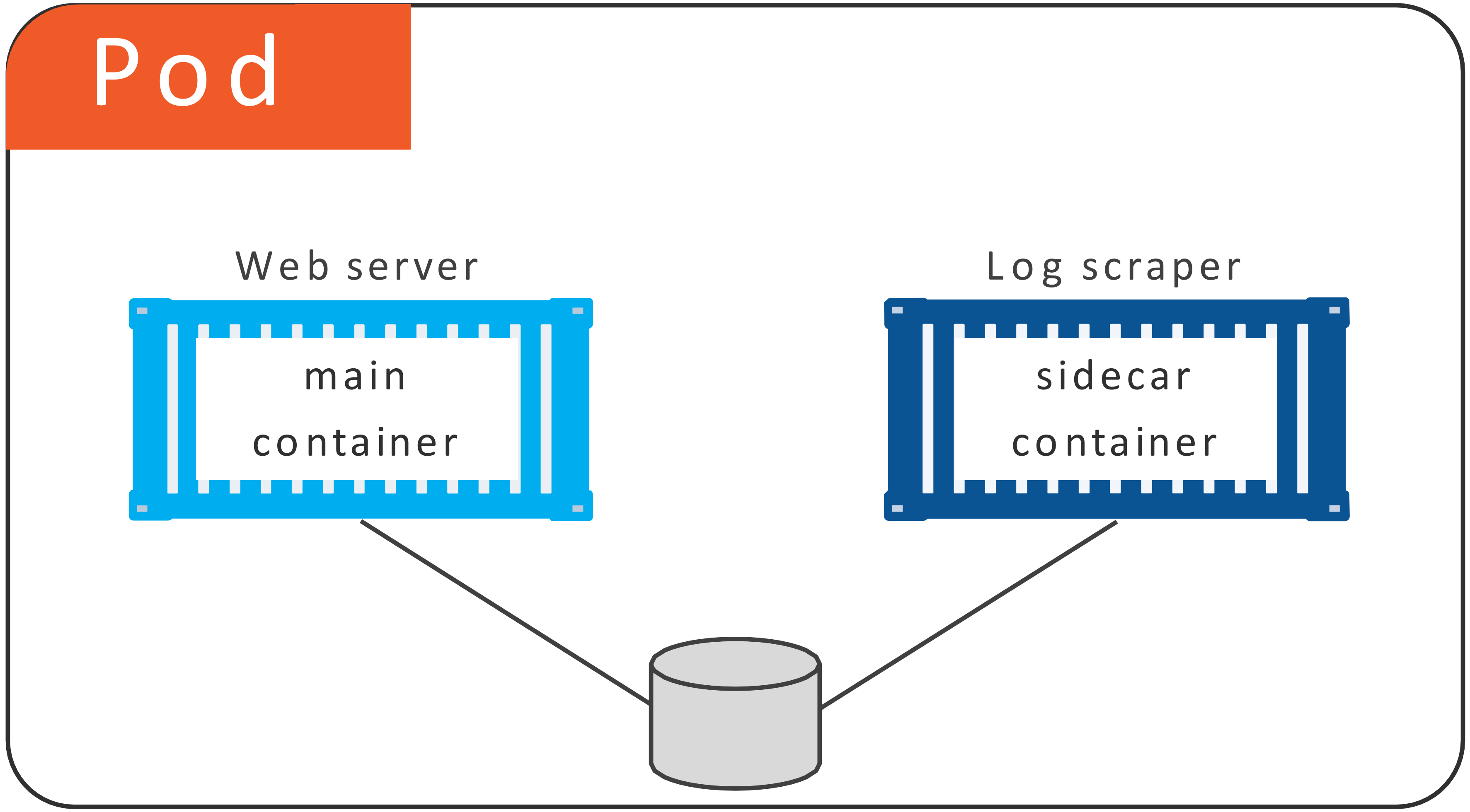
Loose Coupling



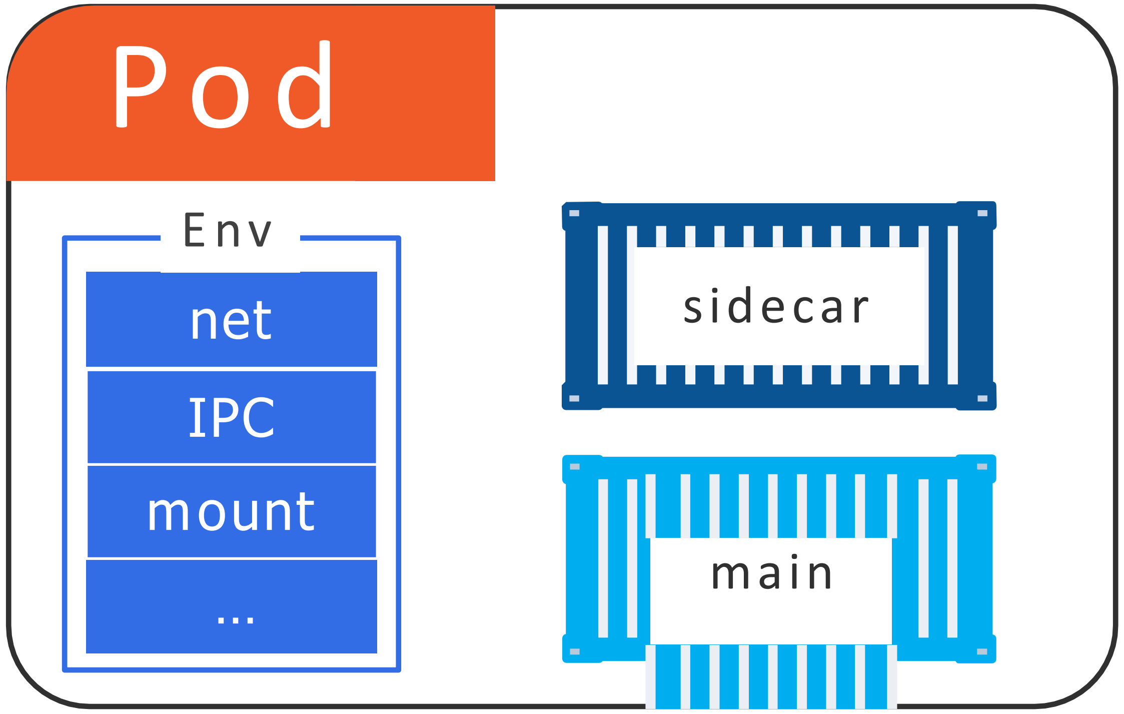
Pods and Scaling



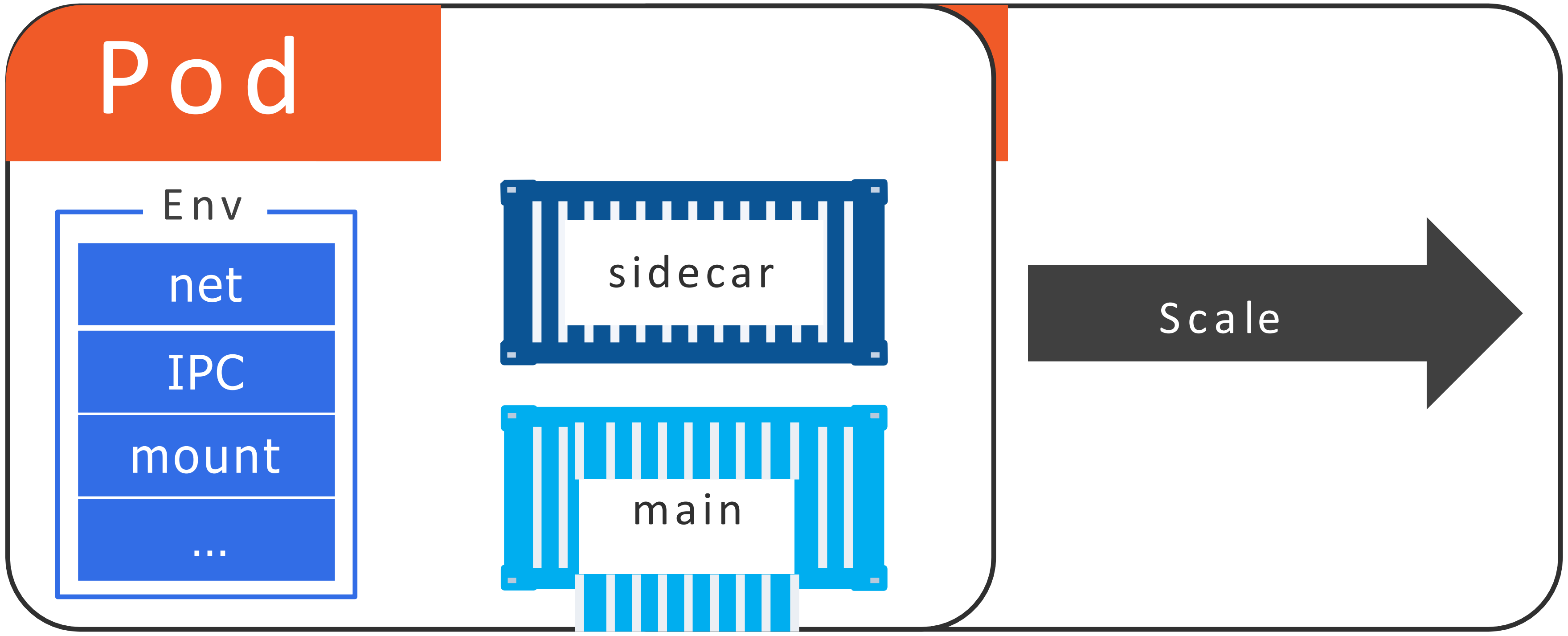
Multi-container Pods



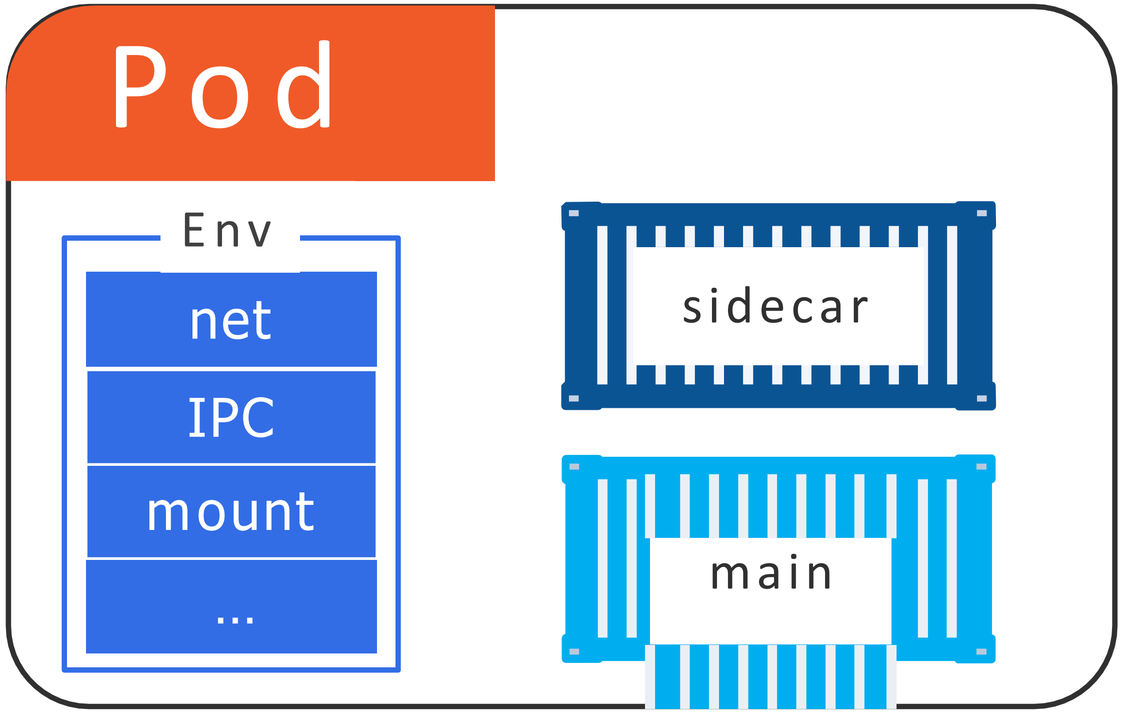
Pods are Atomic



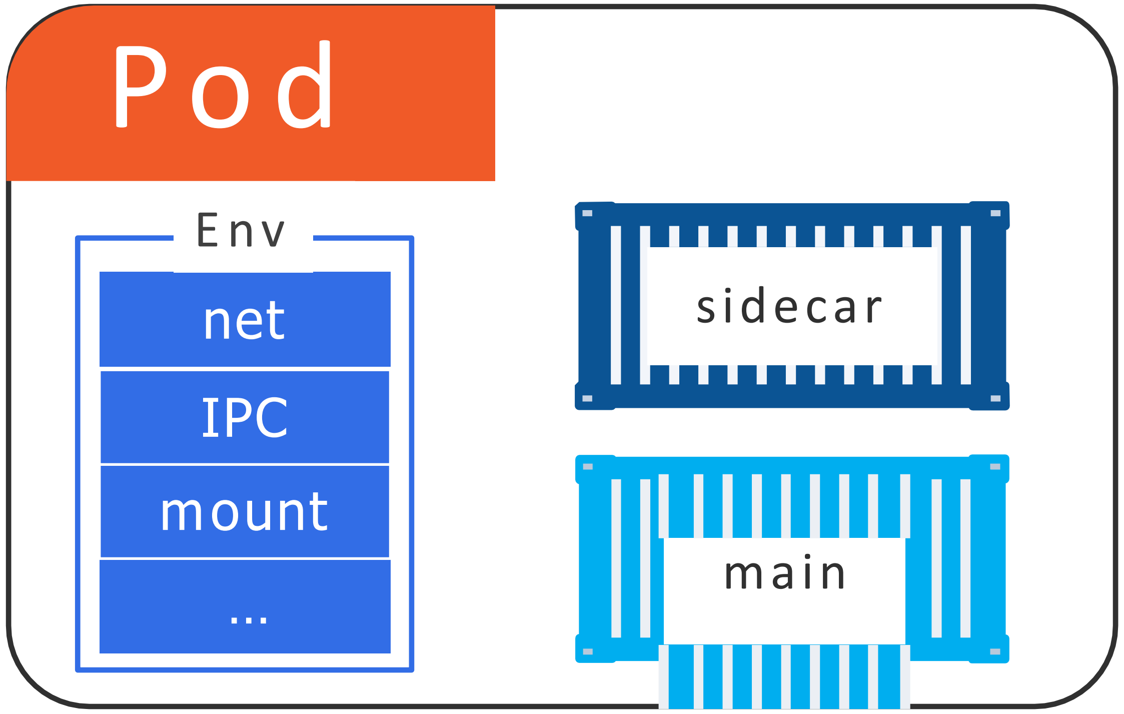
Pods are Atomic



Pods are Atomic

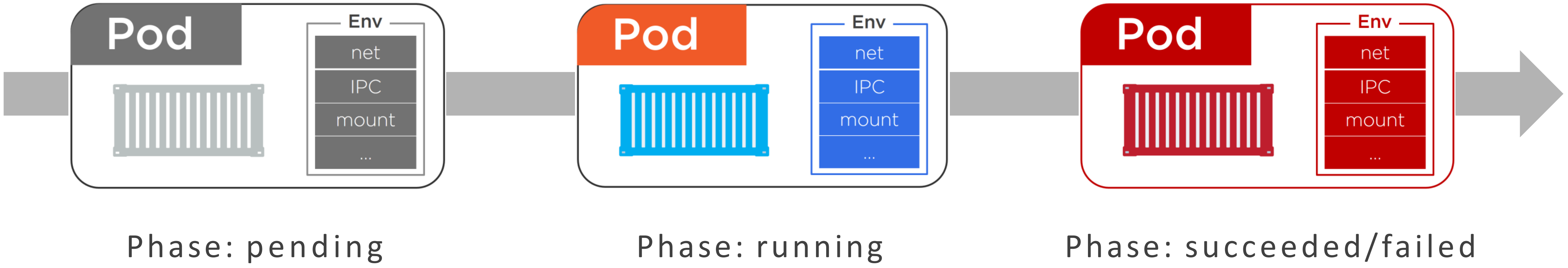


#1 Status:ready



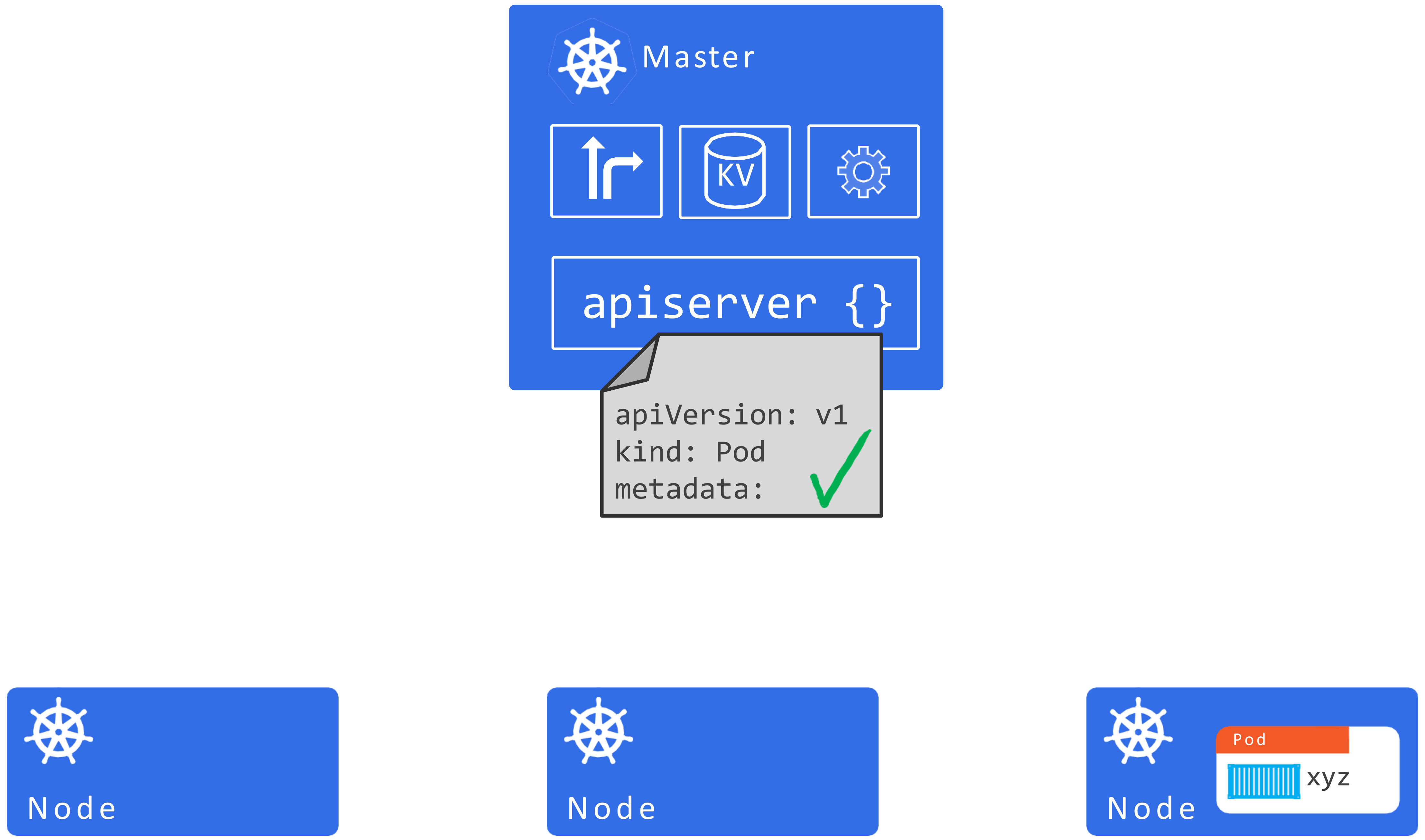
#2 Status:pending

Pod Lifecycle



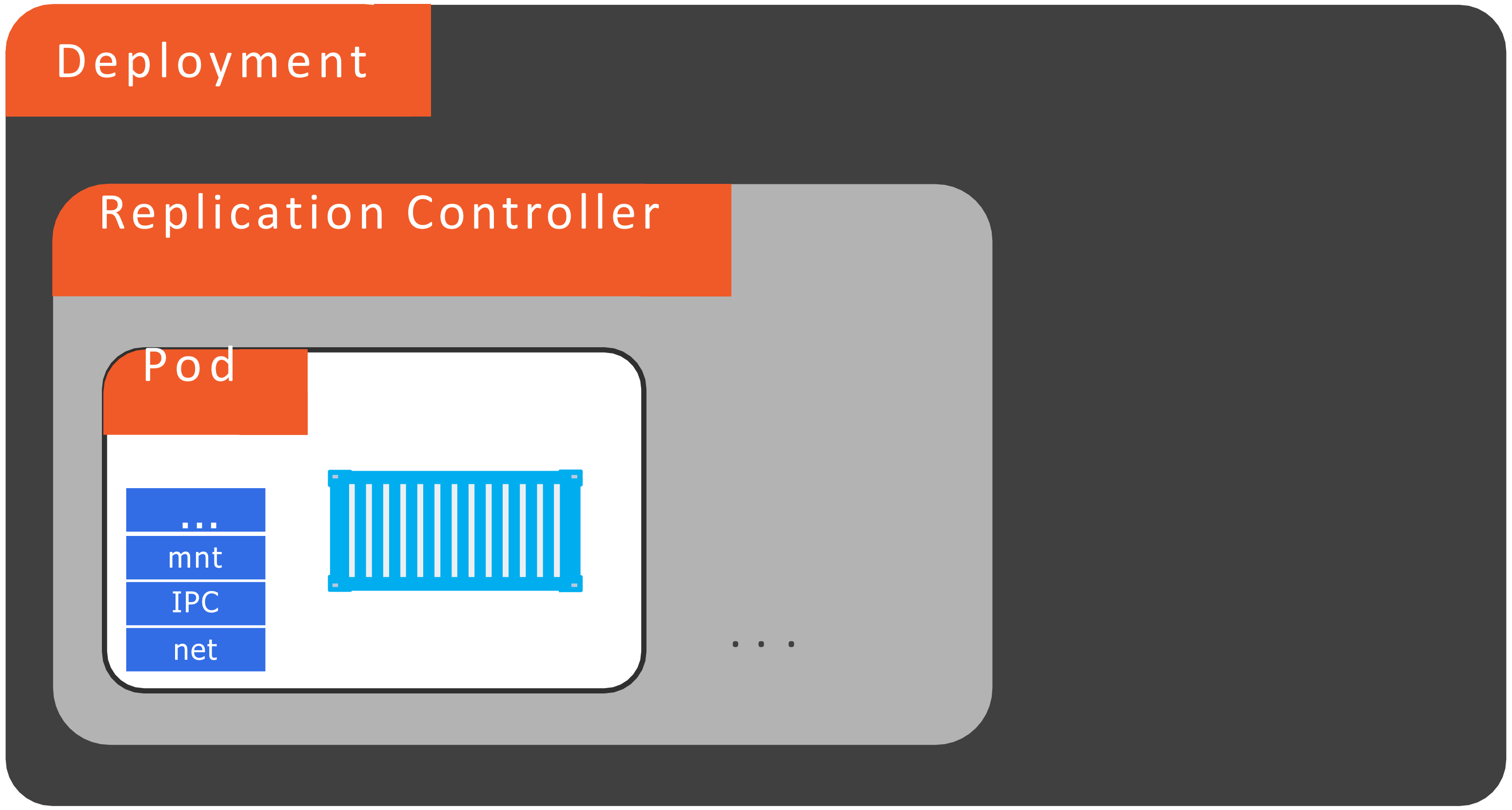
Deploying Pods

Usually via higher level objects

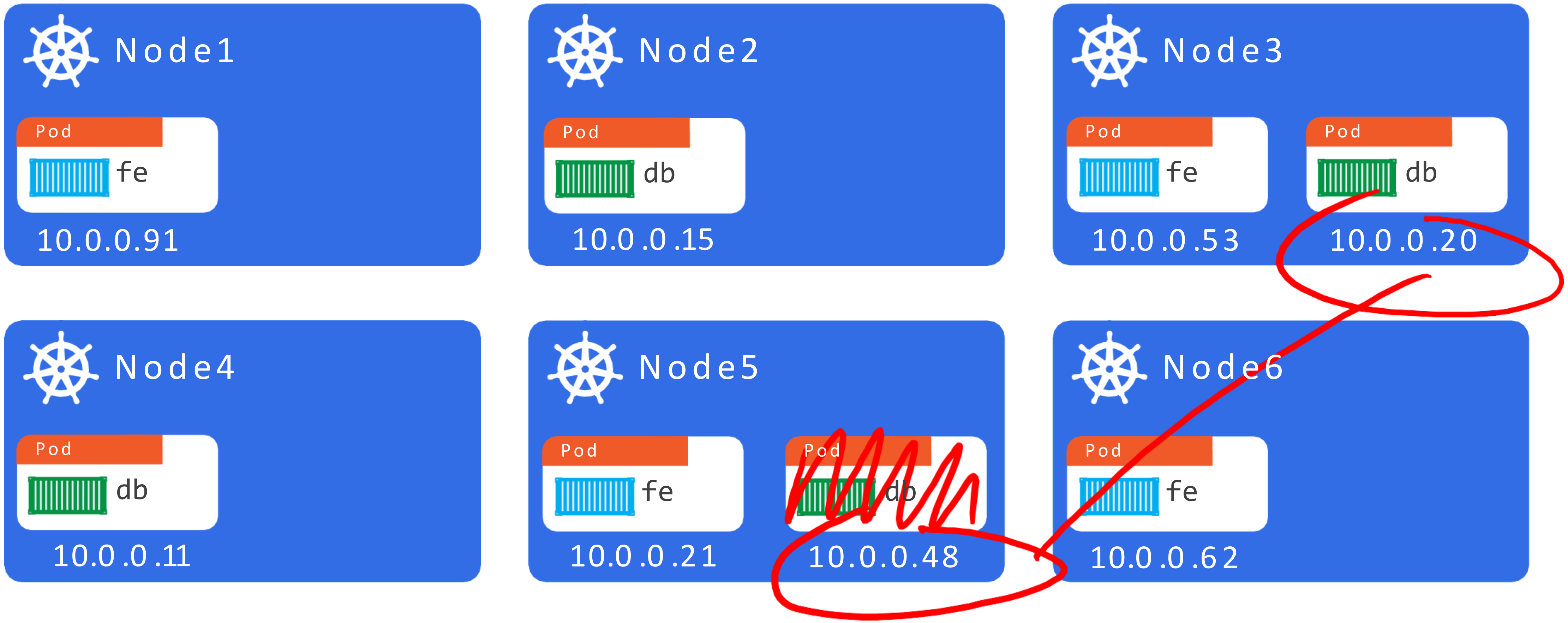

Master
KV
apiserver {}
apiVersion: v1
kind: Pod
metadata:
Node
Node
Node
Pod
xyz

Deploying Pods

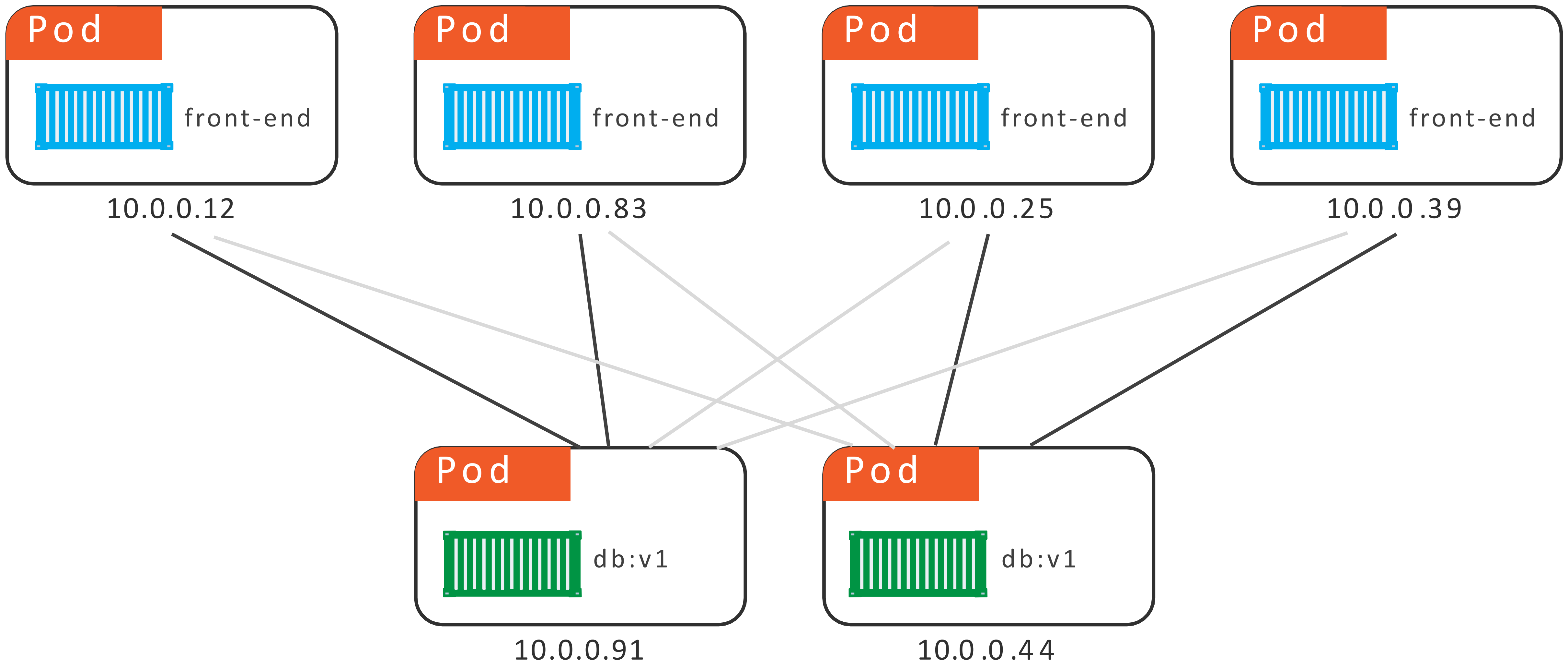
Usually via higher level objects

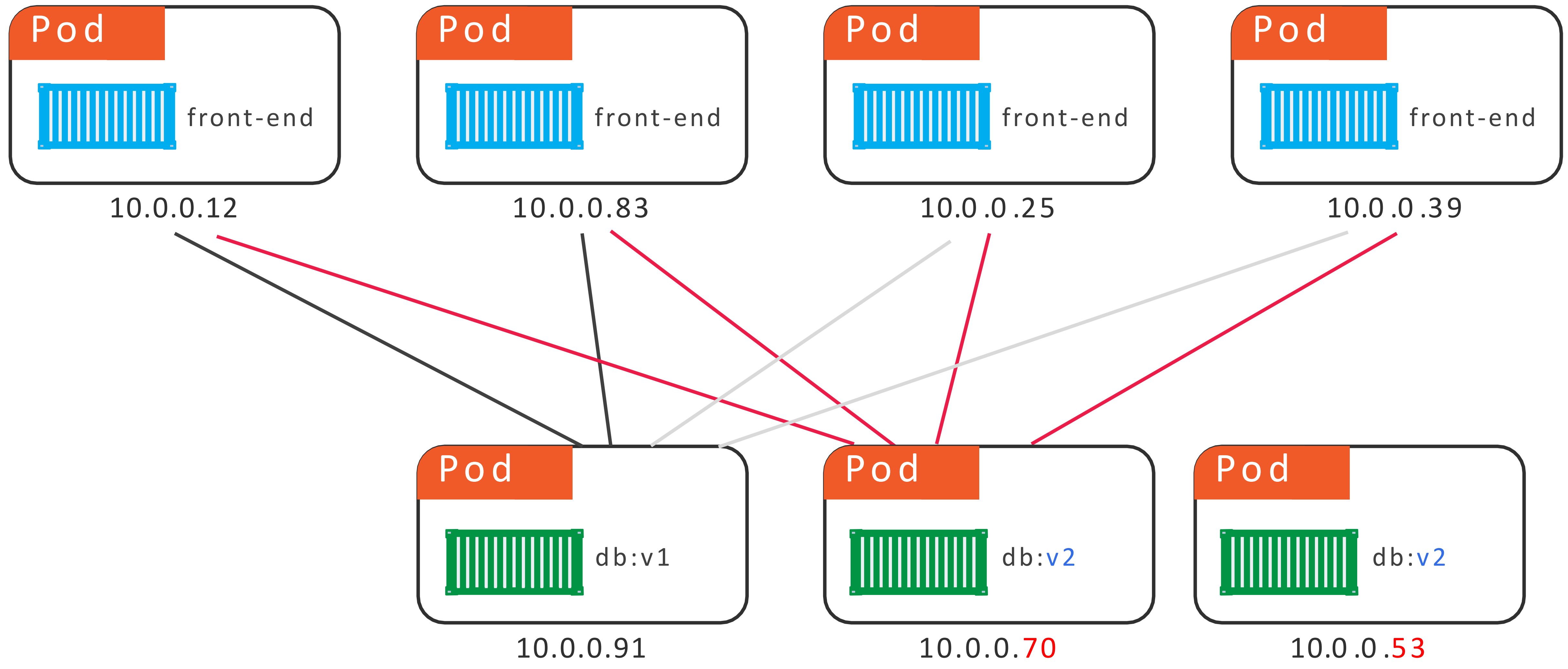


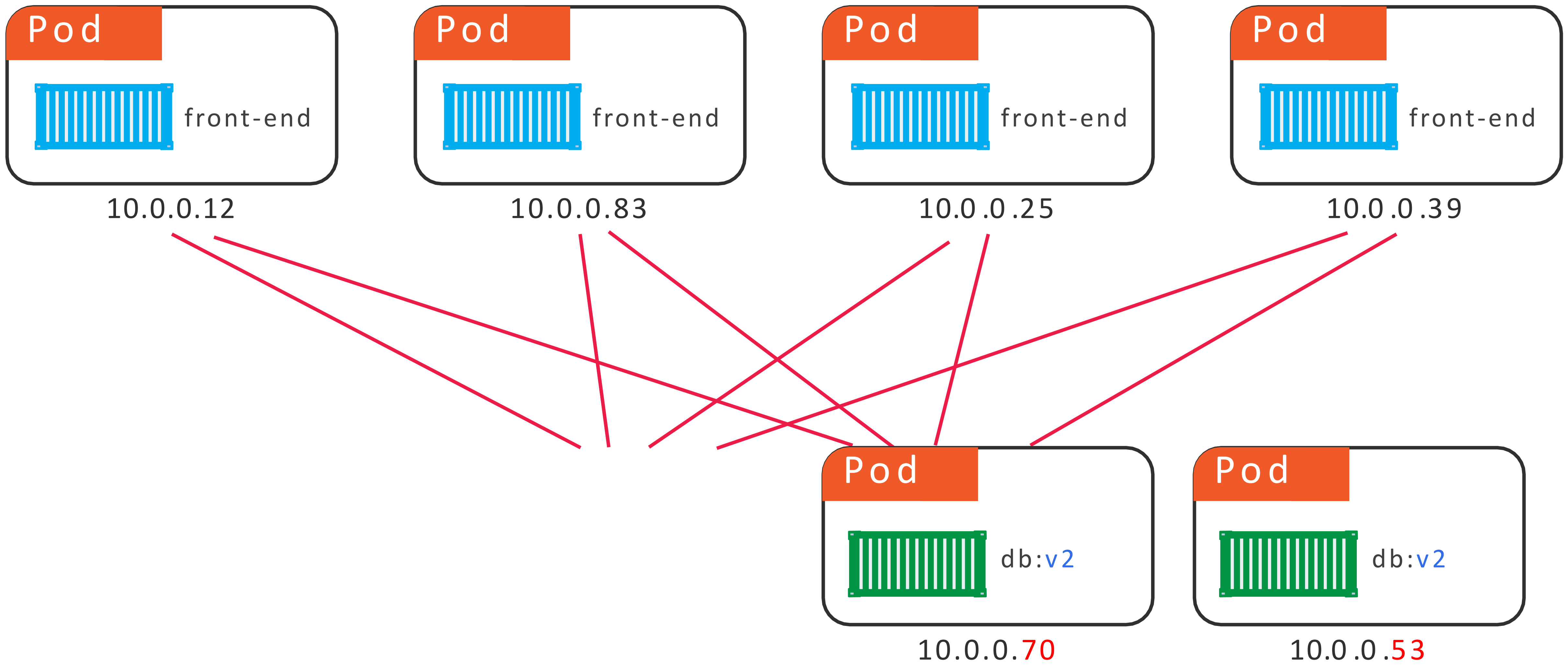
Services

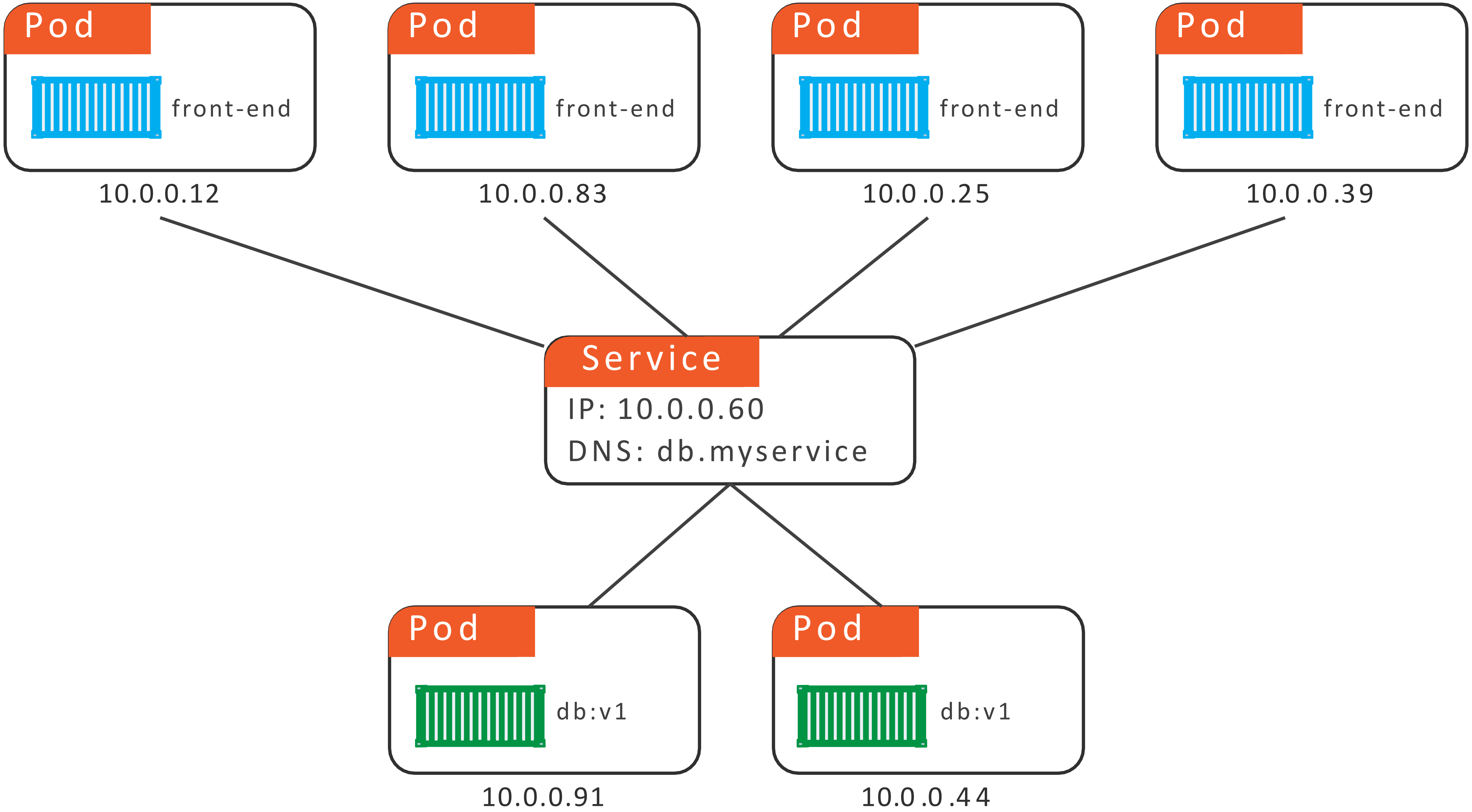


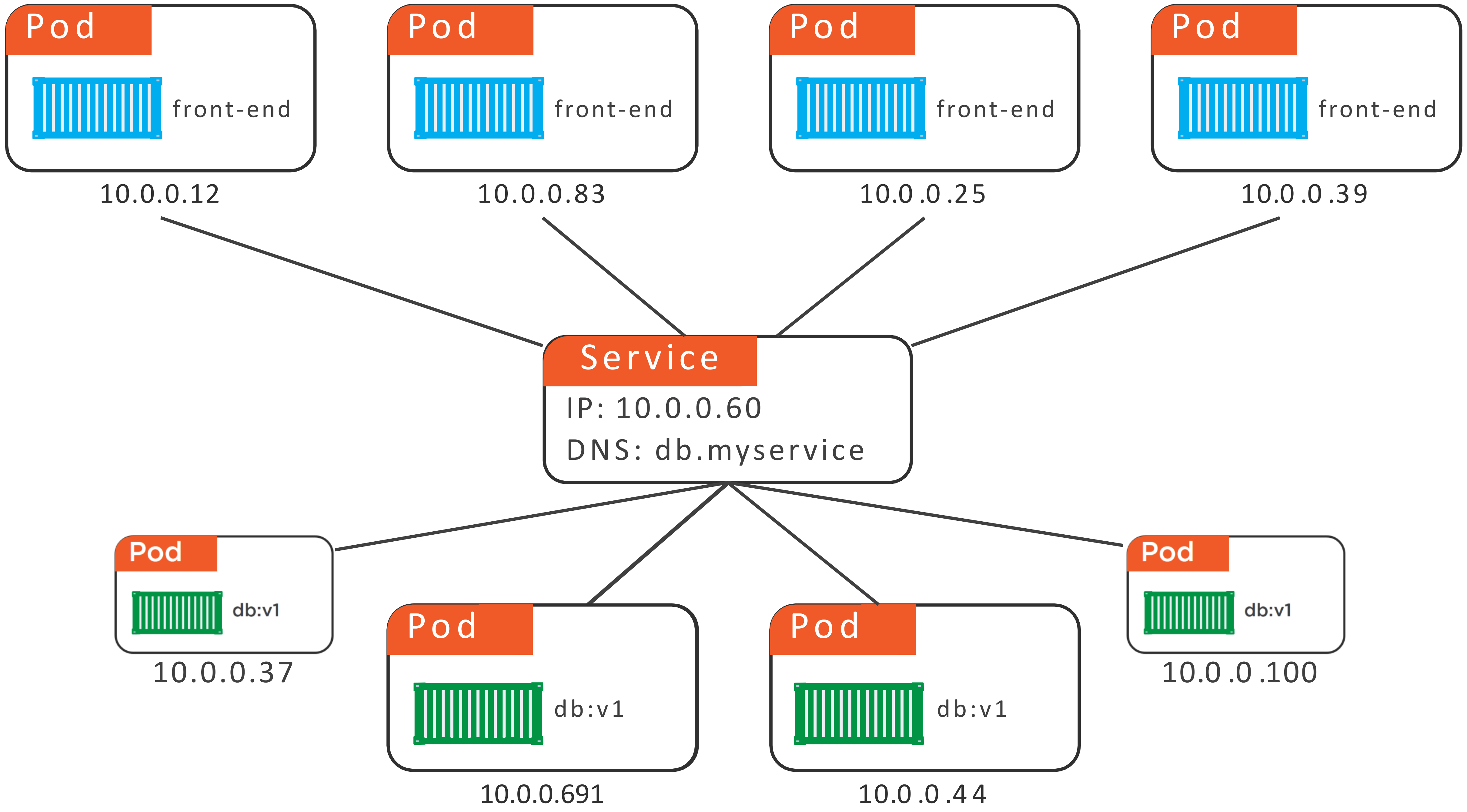
Every new pod gets a new IP = IP churn!

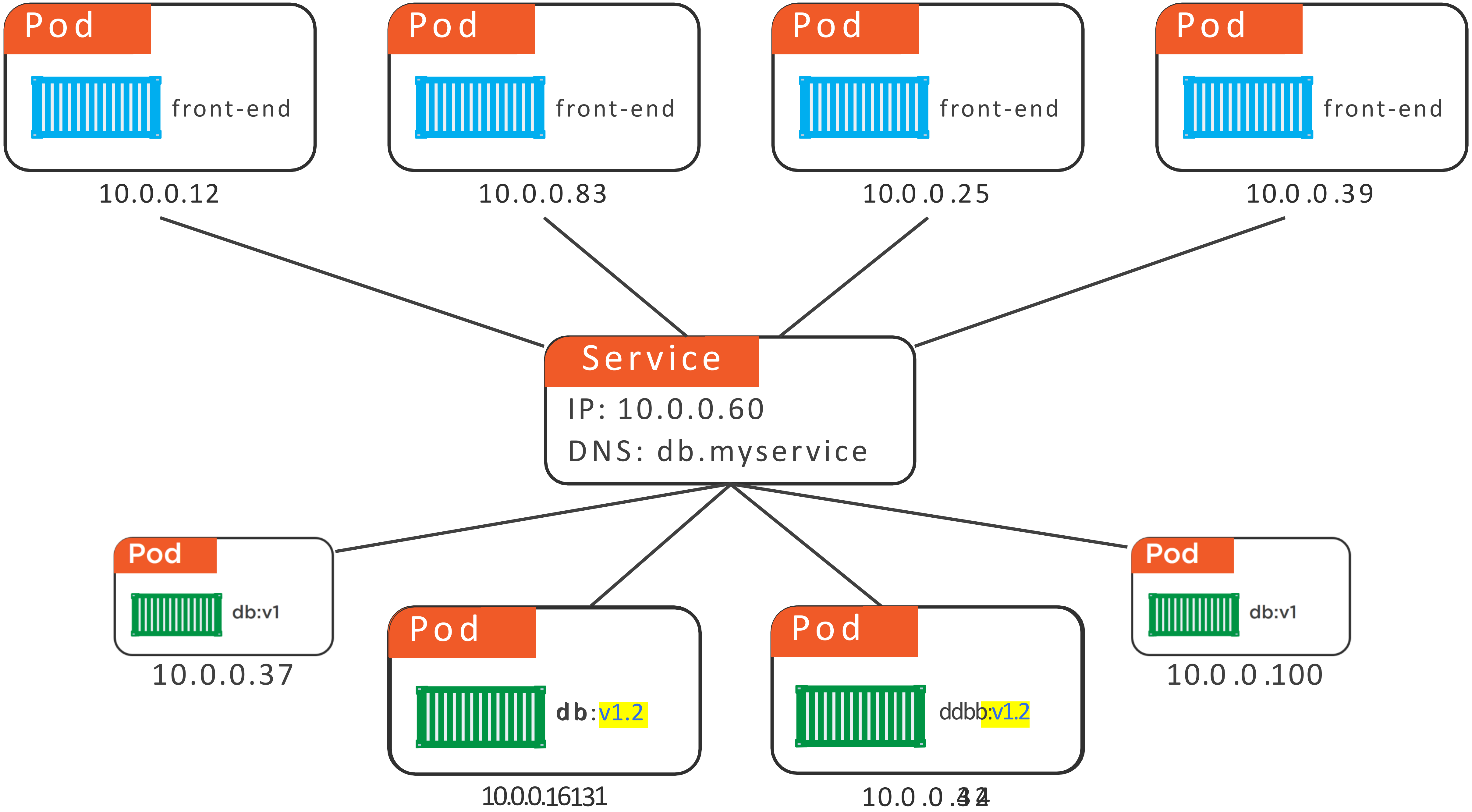

Pod
front-end
10.0.0.12
Pod
front-end
10.0.0.83
Pod
front-end
10.0.0.25
Pod
front-end
10.0.0.39
Pod
db:v1
10.0.0.91
Pod
db:v1
10.0.0.44

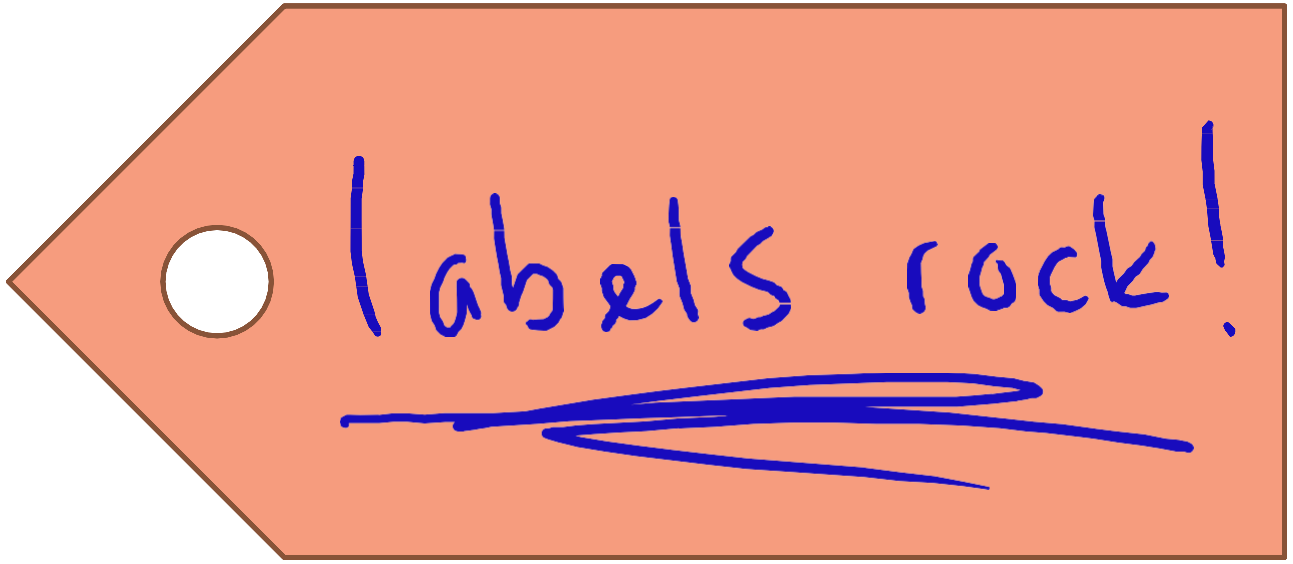

Pod
front-end
10.0.0.12
Pod
front-end
10.0.0.83
Pod
front-end
10.0.0.25
Pod
front-end
10.0.0.39
Pod
db:v1
10.0.0.91
Pod
db:v2
10.0.0.70
Pod
db:v2
10.0.0.53

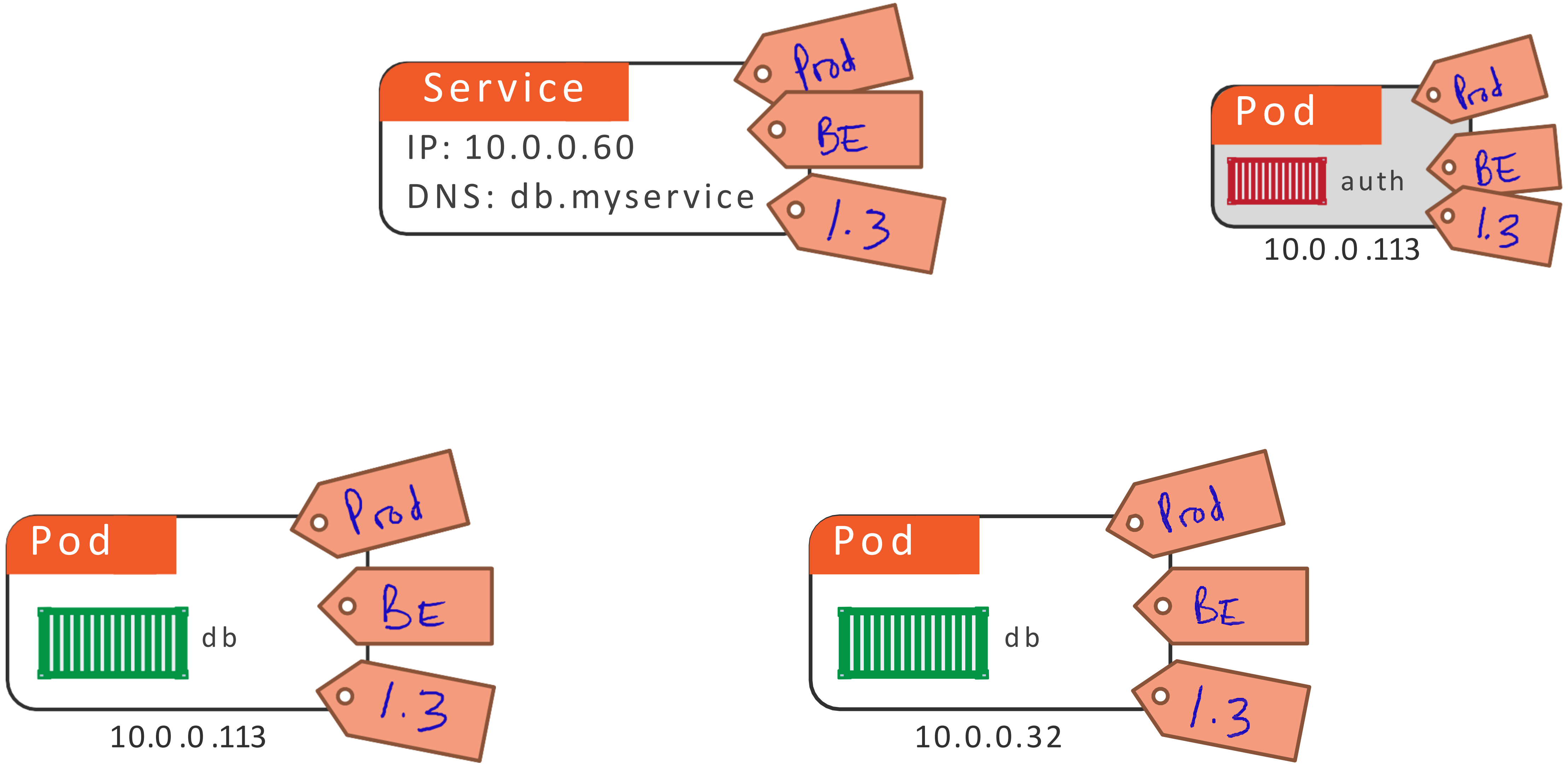

Pod
front-end
10.0.0.12
Pod
front-end
10.0.0.83
Pod
front-end
10.0.0.25
Pod
front-end
10.0.0.39
Pod
db:v2
10.0.0.70
Pod
db:v2
10.0.0.53

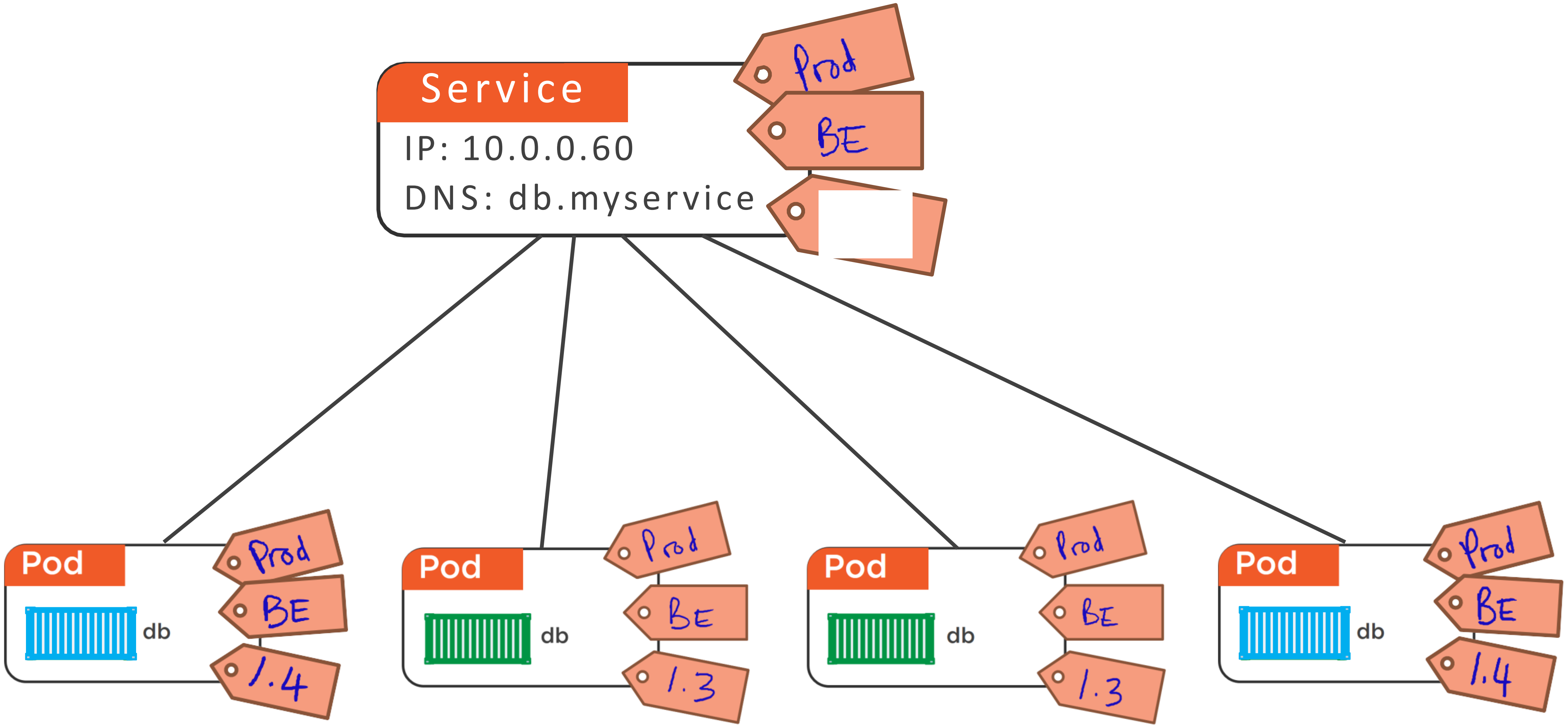

Pod
front-end
10.0.0.12
Pod
front-end
10.0.0.83
Pod
front-end
10.0.0.25
Pod
front-end
10.0.0.39
Service
IP: 10.0.0.60
DNS: db.myservice
Pod
db:v1
10.0.0.91
Pod
db:v1
10.0.0.44

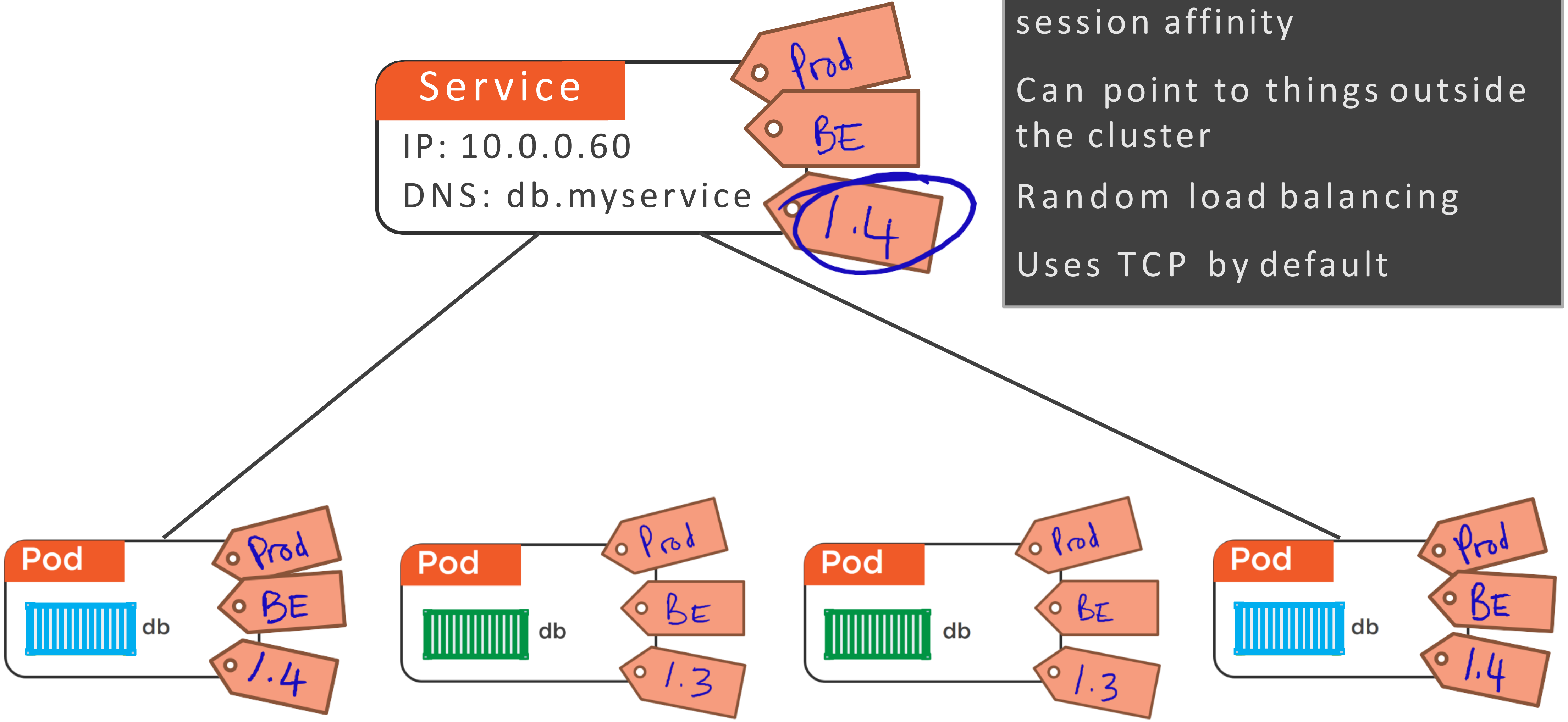

Pod
front-end
10.0.0.12
Pod
front-end
10.0.0.83
Pod
front-end
10.0 .0 .25
Pod
front-end
10.0 .0 .39
Service
IP: 10.0.0.60
DNS: db.myservice
Pod
db:v1
10.0.0.37
Pod
db:v1
10.0.0.691
Pod
db:v1
10.0 .0 .4 4
Pod
db:v1
10.0 .0 .100


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
10.0.0.16131
Pod
ddbb:v1.2
10.0.0.34
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
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

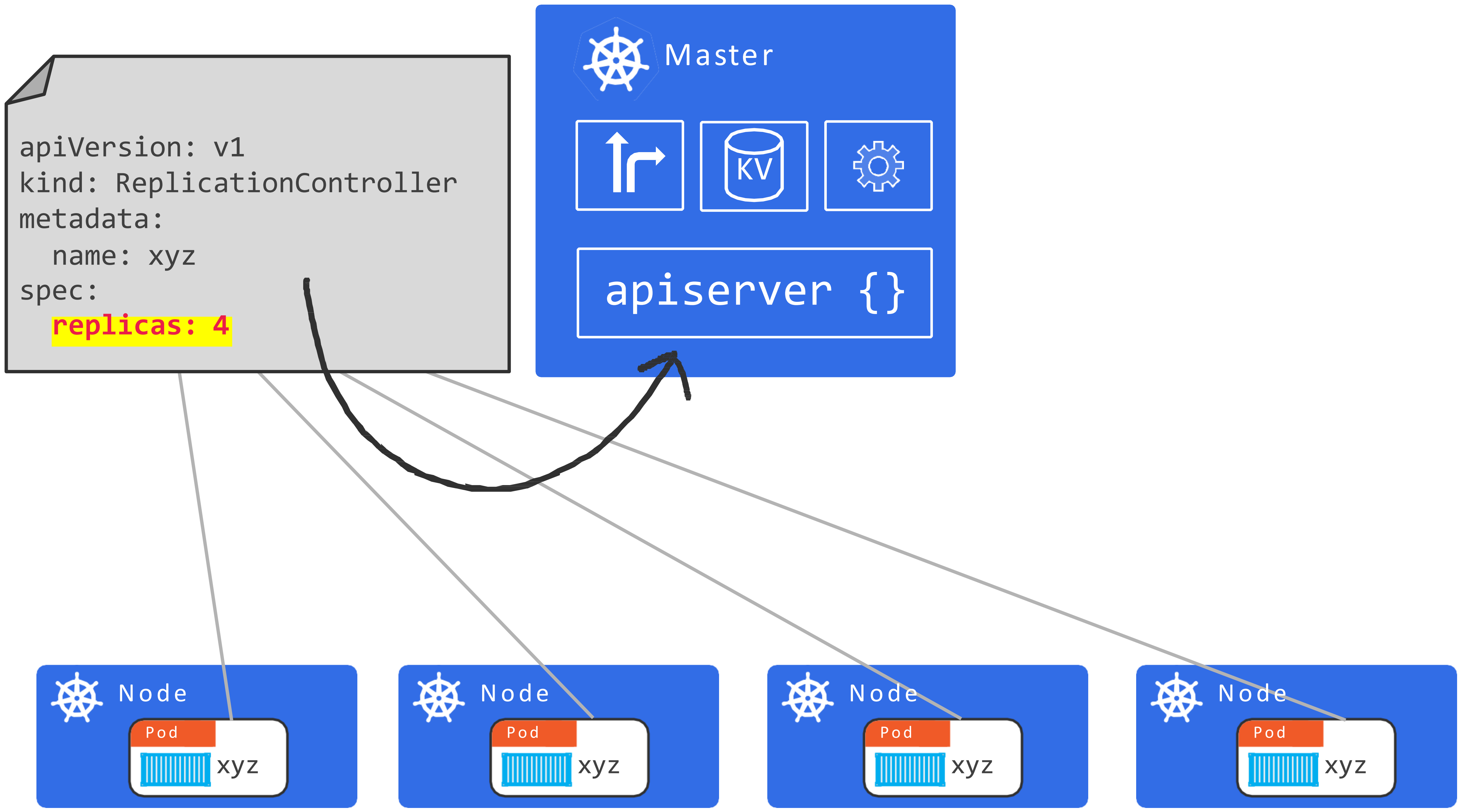
Can be configured for session affinity

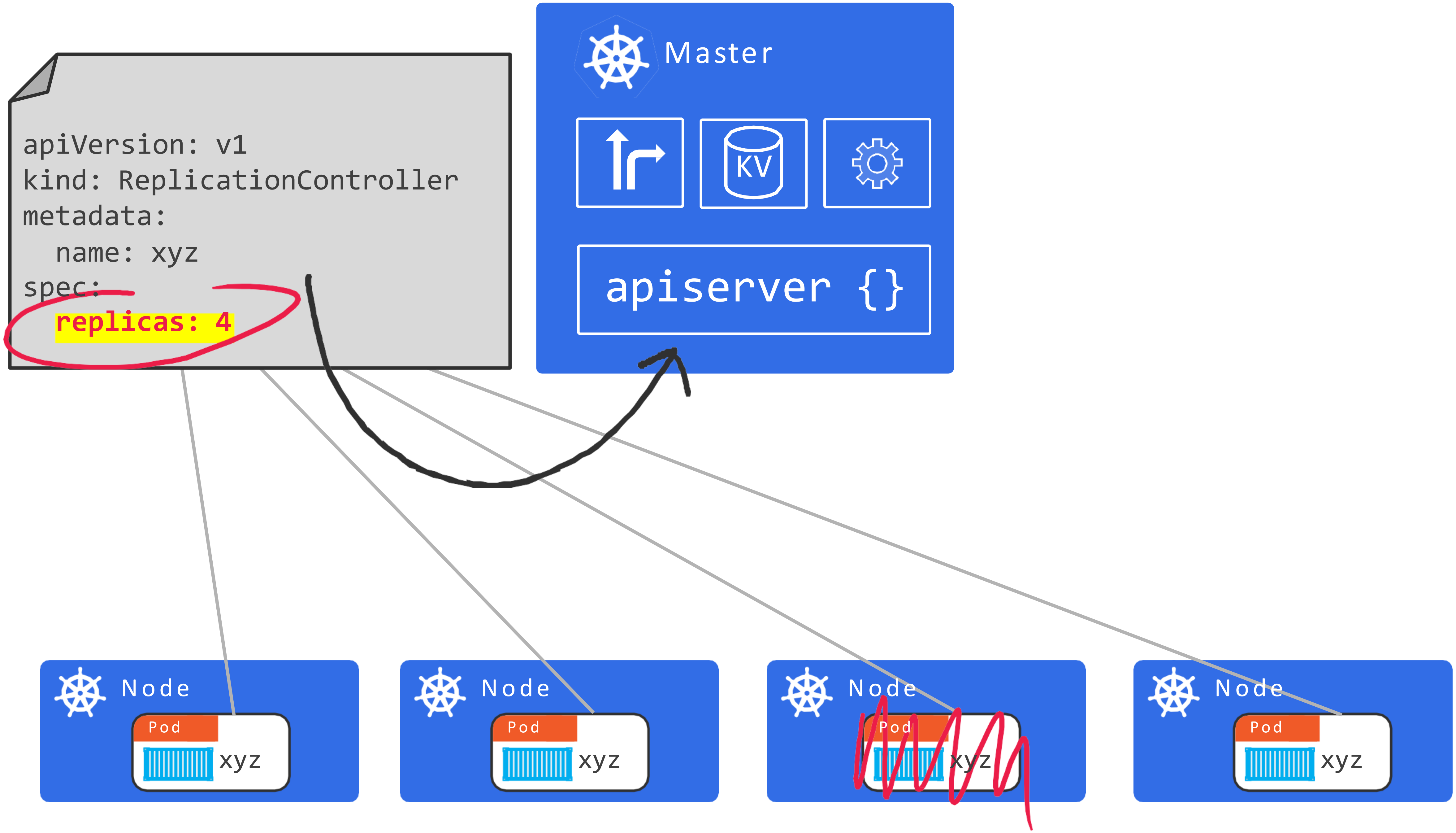
Can point to things outside the cluster

Random load balancing

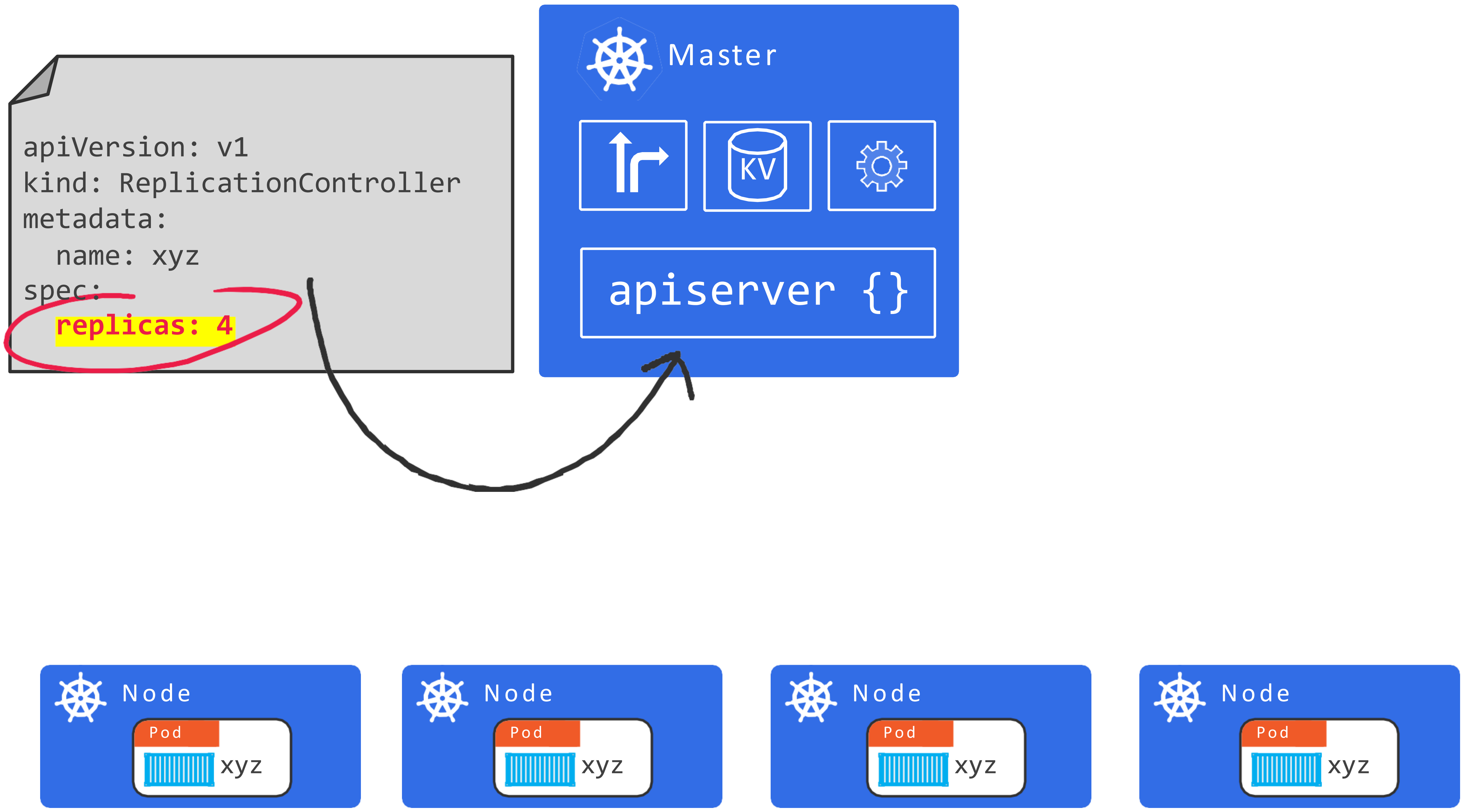
Uses TCP by default

Deployments


```yaml
apiVersion: v1
kind: ReplicationController
metadata:
  name: xyz
spec:
  replicas: 4
```



```
apiVersion: v1
kind: ReplicationController
metadata:
  name: xyz
spec:
  replicas: 4
```



```
apiVersion: v1
kind: ReplicationController
metadata:
  name: xyz
spec:
  replicas: 4
```

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) RCv2

Versioned

Deployed via the apiserver

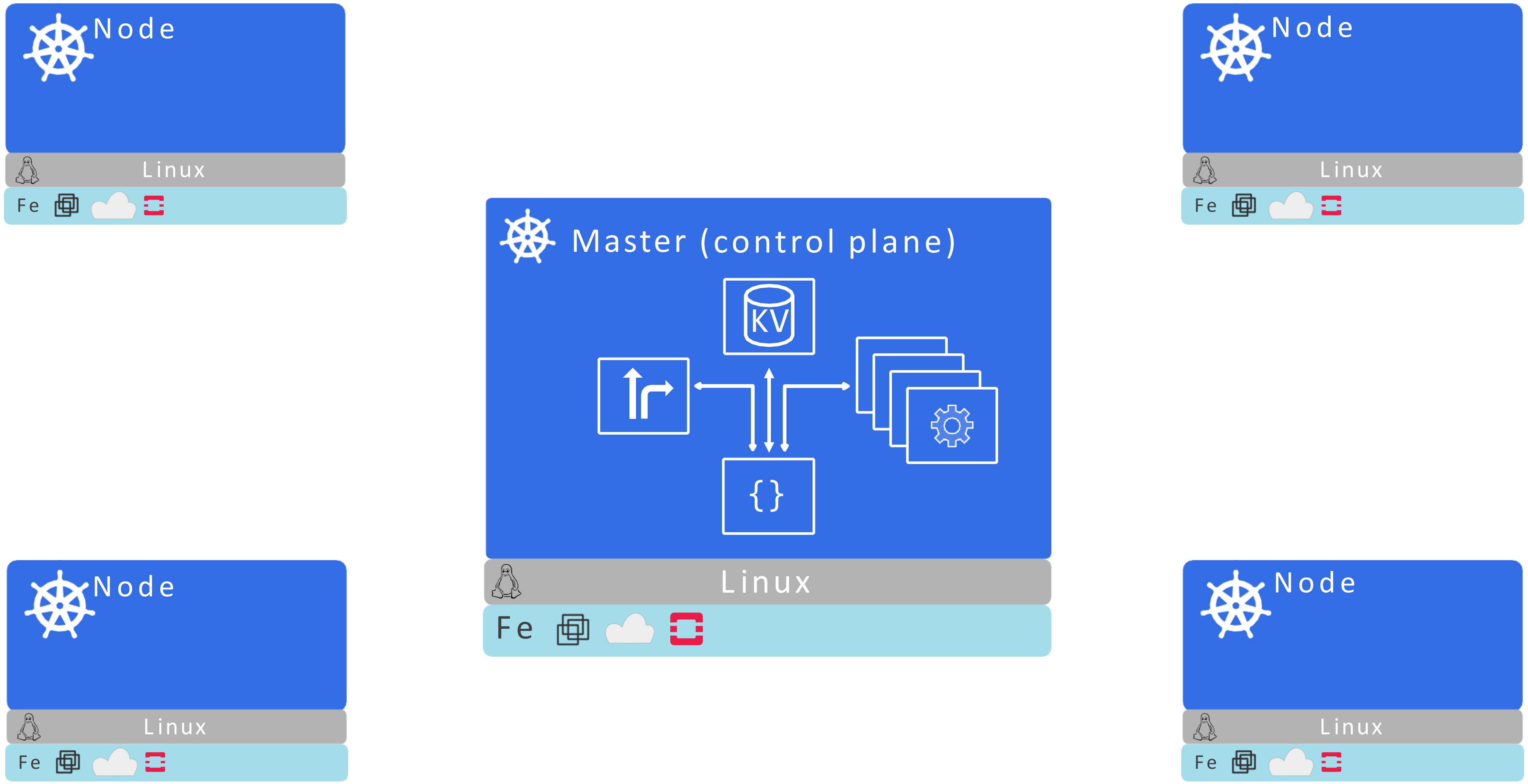
Multiple concurrent versions

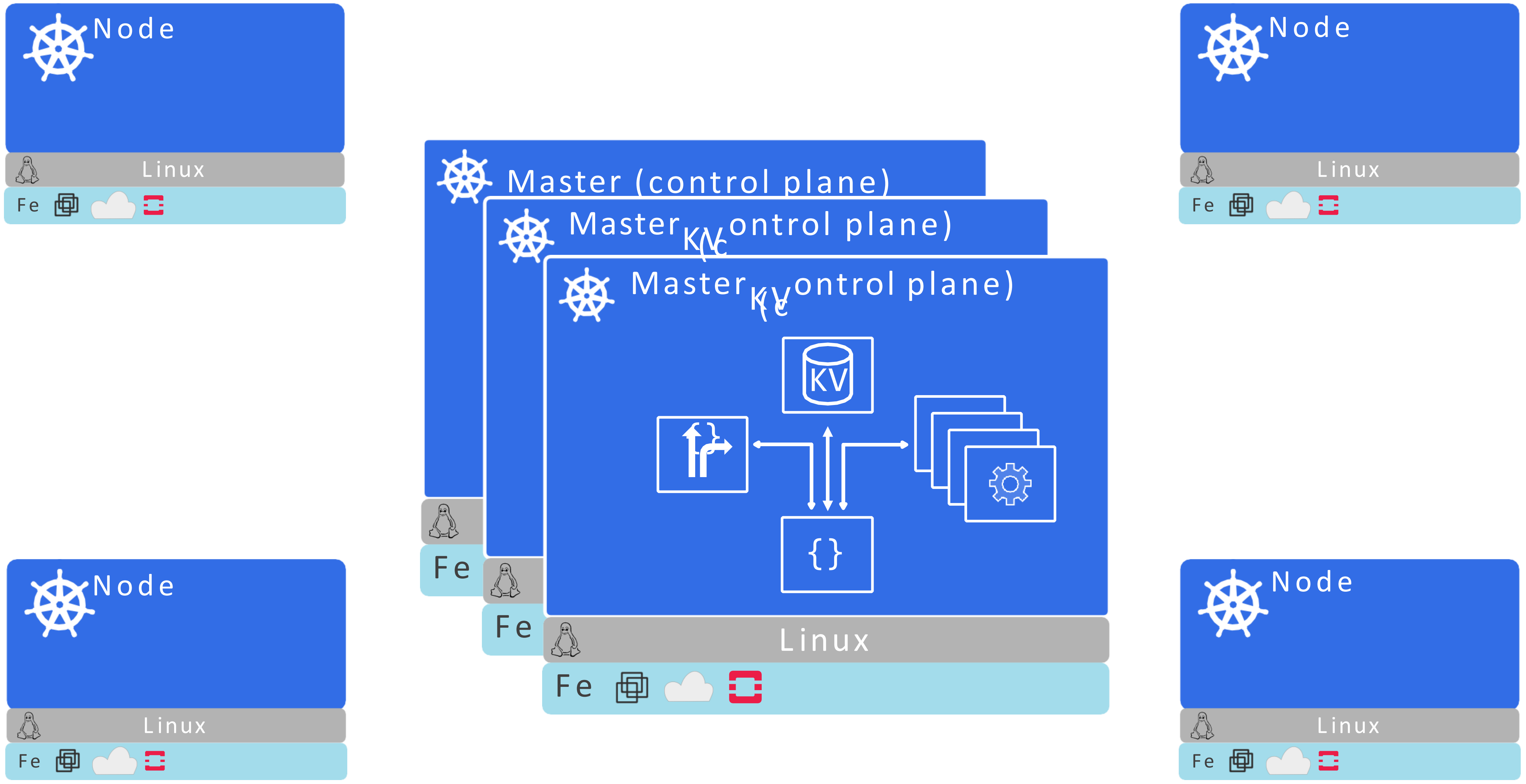
- Blue-green deployments
- Canary releases

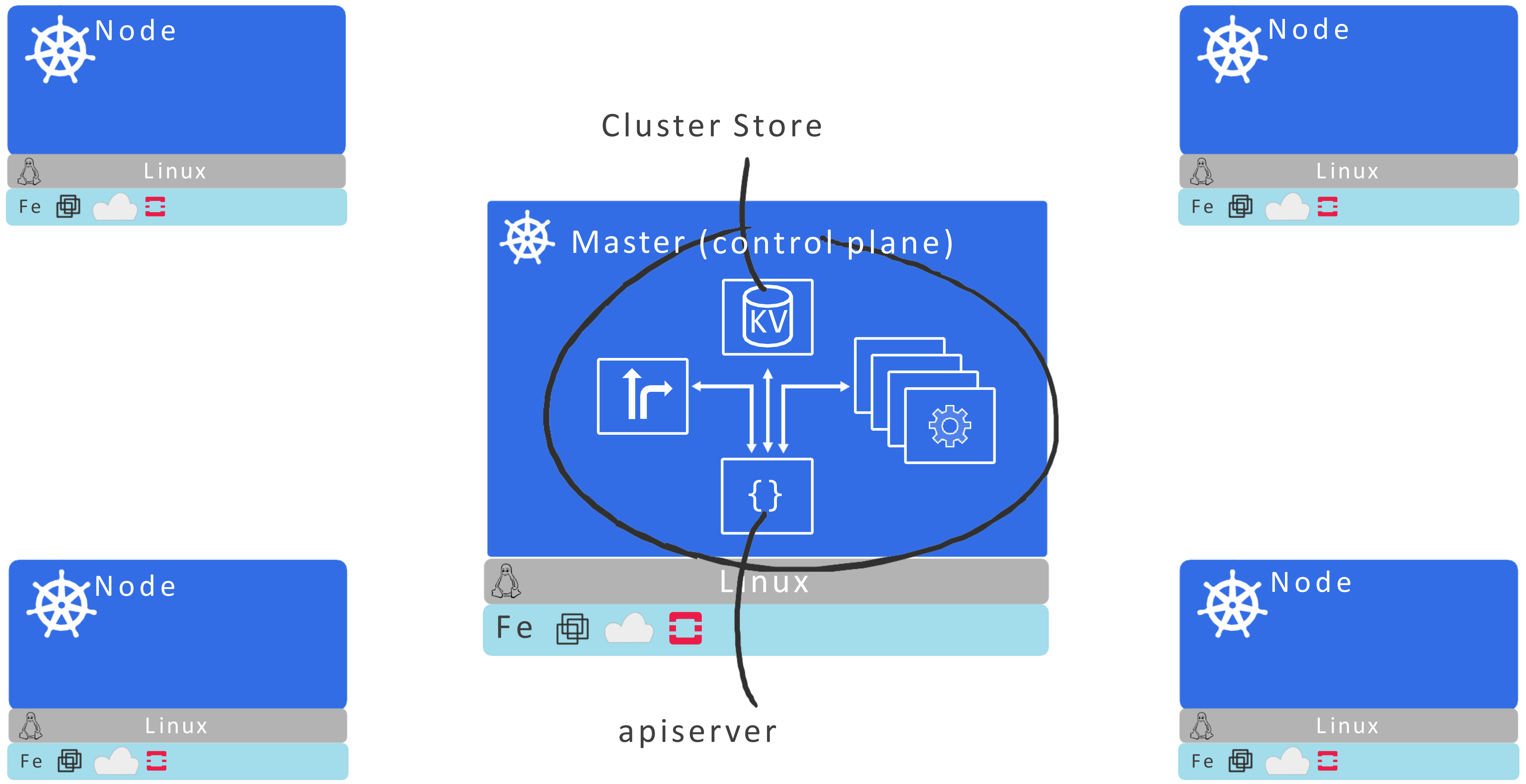
```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: xyz
spec:
  replicas: 4
```

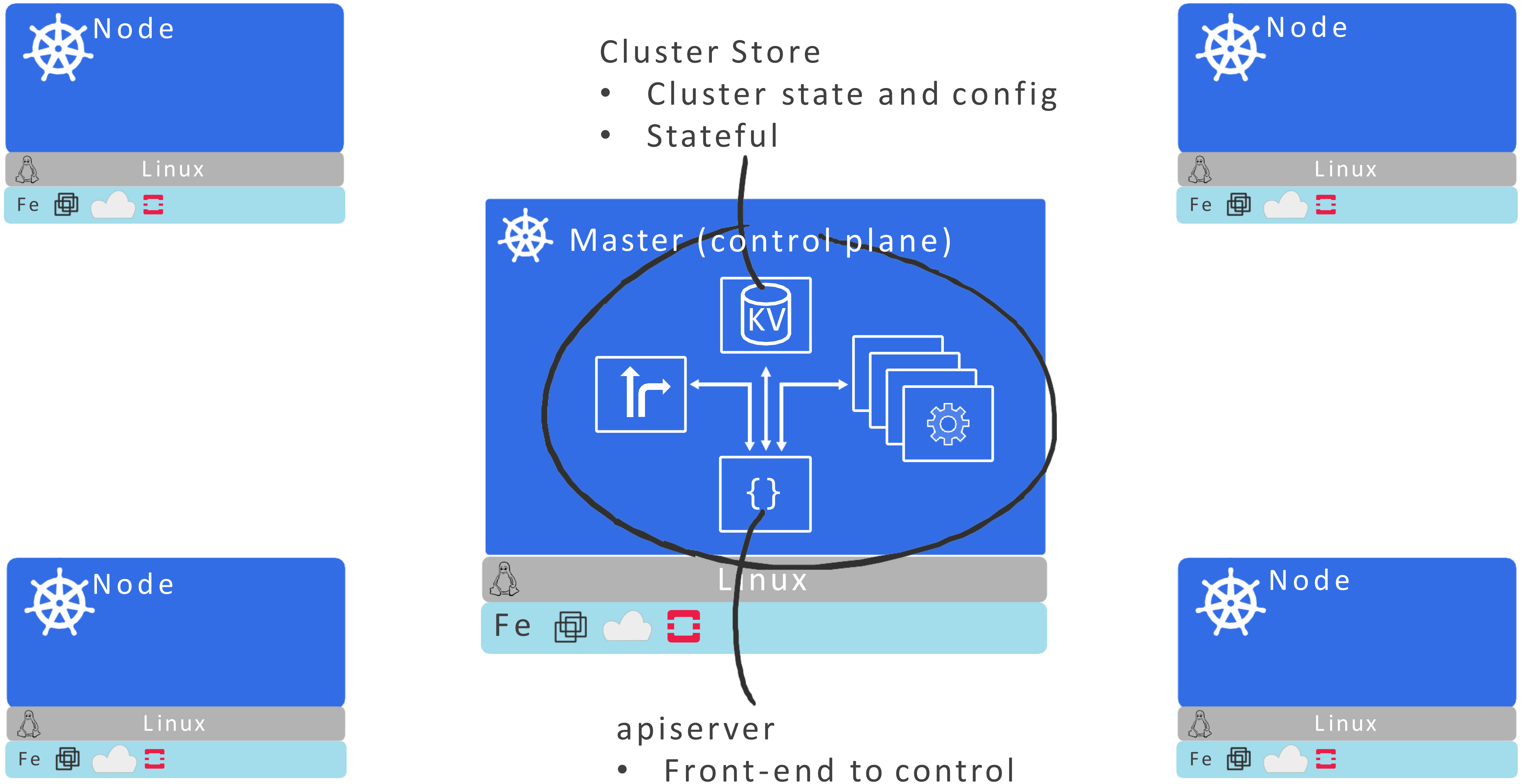
Simple rolling updates and rollbacks

Simple versioned rollbacks


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
{}
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

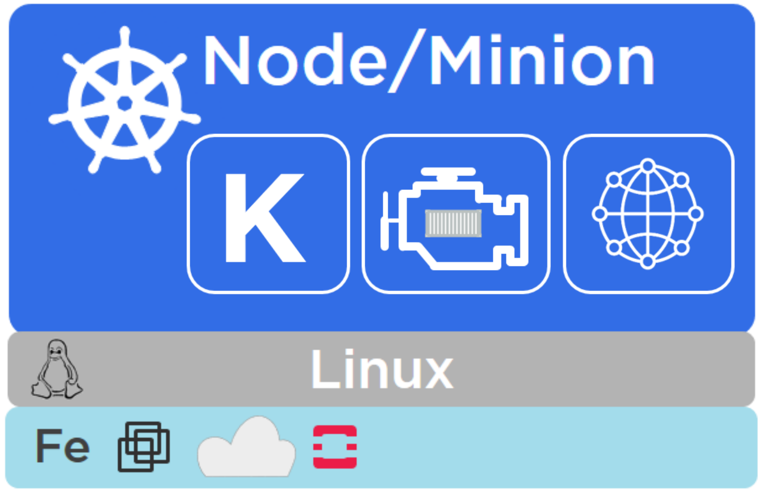


Cluster Store

- Cluster state and config
- Stateful

apiserver

- Front-end to control plane



Kubelet

Main Kubernetes agent

Container engine

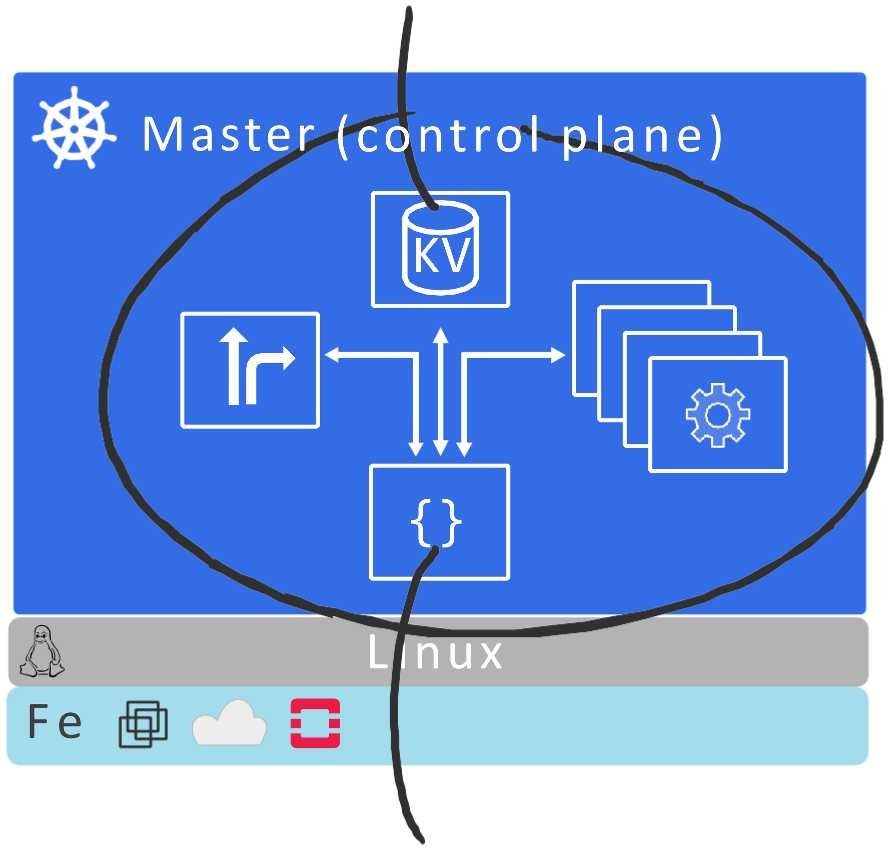
Docker or rkt

kube-proxy

Kubernetes networking

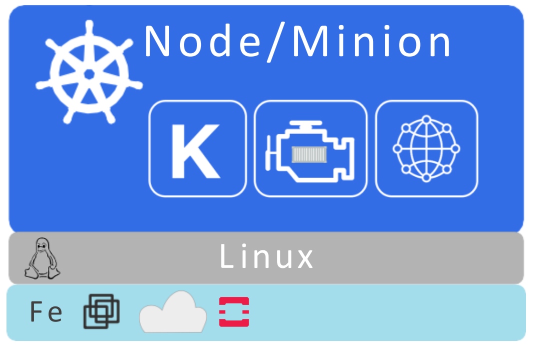
Cluster Store

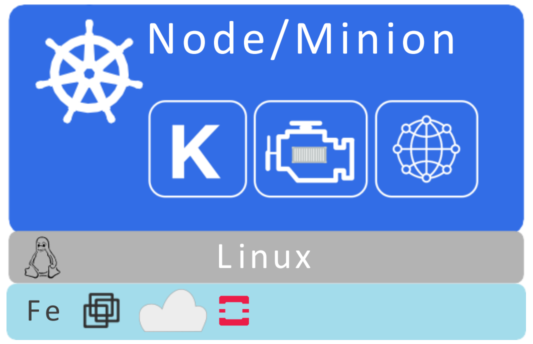
- Cluster state and config
- Stateful



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