Kubernetes Arabic Course

Kubernetes From Scratch - Hands-on

01 - Introduction

Course Contents

0 Introduction

- Course Introduction
- What is Kubernetes
- Kubernetes Features

1 Kubernetes Architecture

- Cluster Architecture
- Master Components
- Worker Components

0 Setup Lab Cluster

- Setup Single Node Cluster (Minikube)
- Setup Kind Cluster
- Setup Internal Docker Registry

2 Basic Concepts

- Namespaces
- Resource Quota
- Labels and Selectors
- Nodes

0

Course Contents

0 Pods

- Pods Explained
- Pods demo
- Multi-container pod

5 Deployments

- Deployment Explained
- Deployment Demo
- Rolling Update

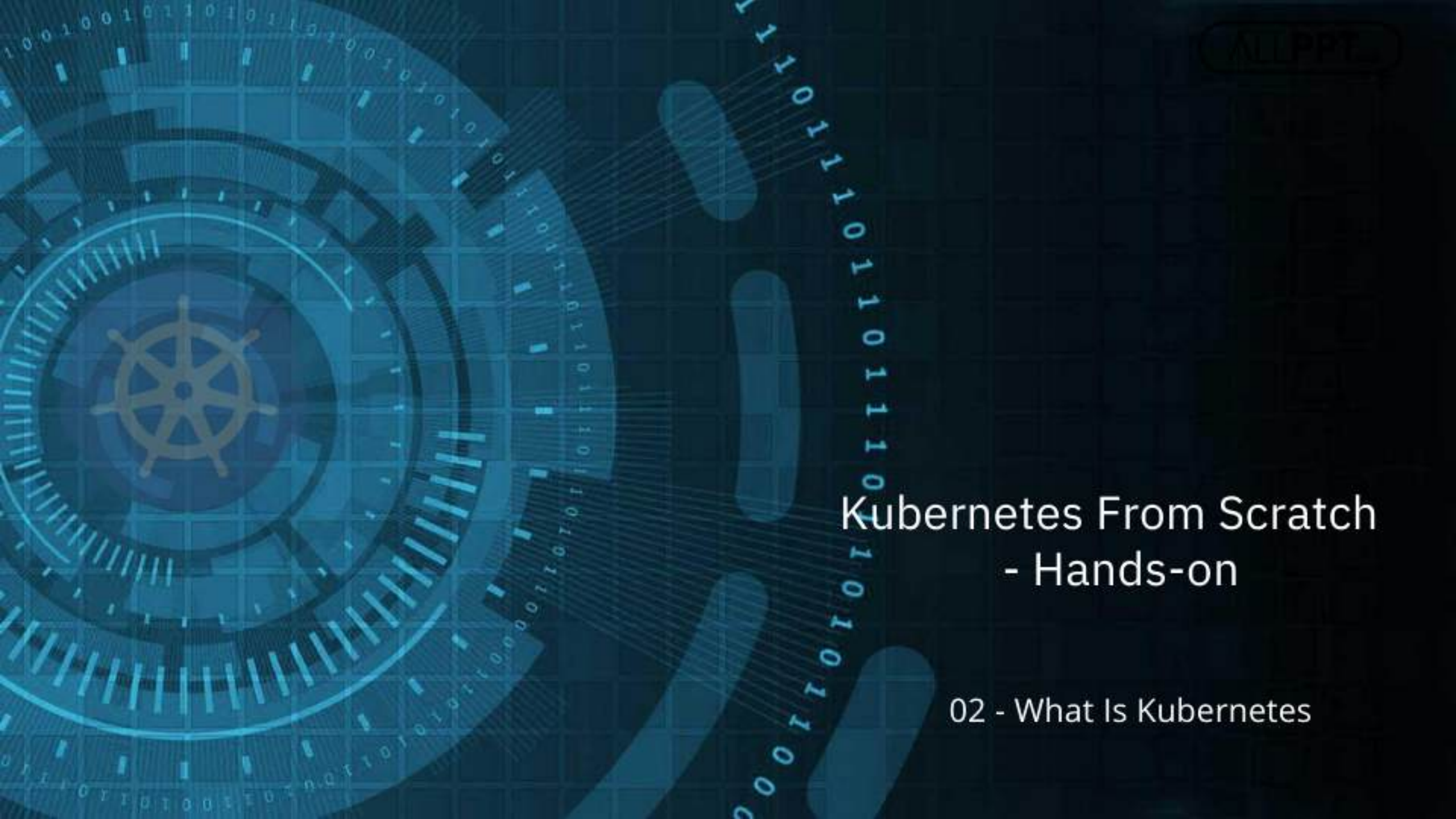
0 Services

- Service Explained
- Service Types
- Service Demo

6 Ingress

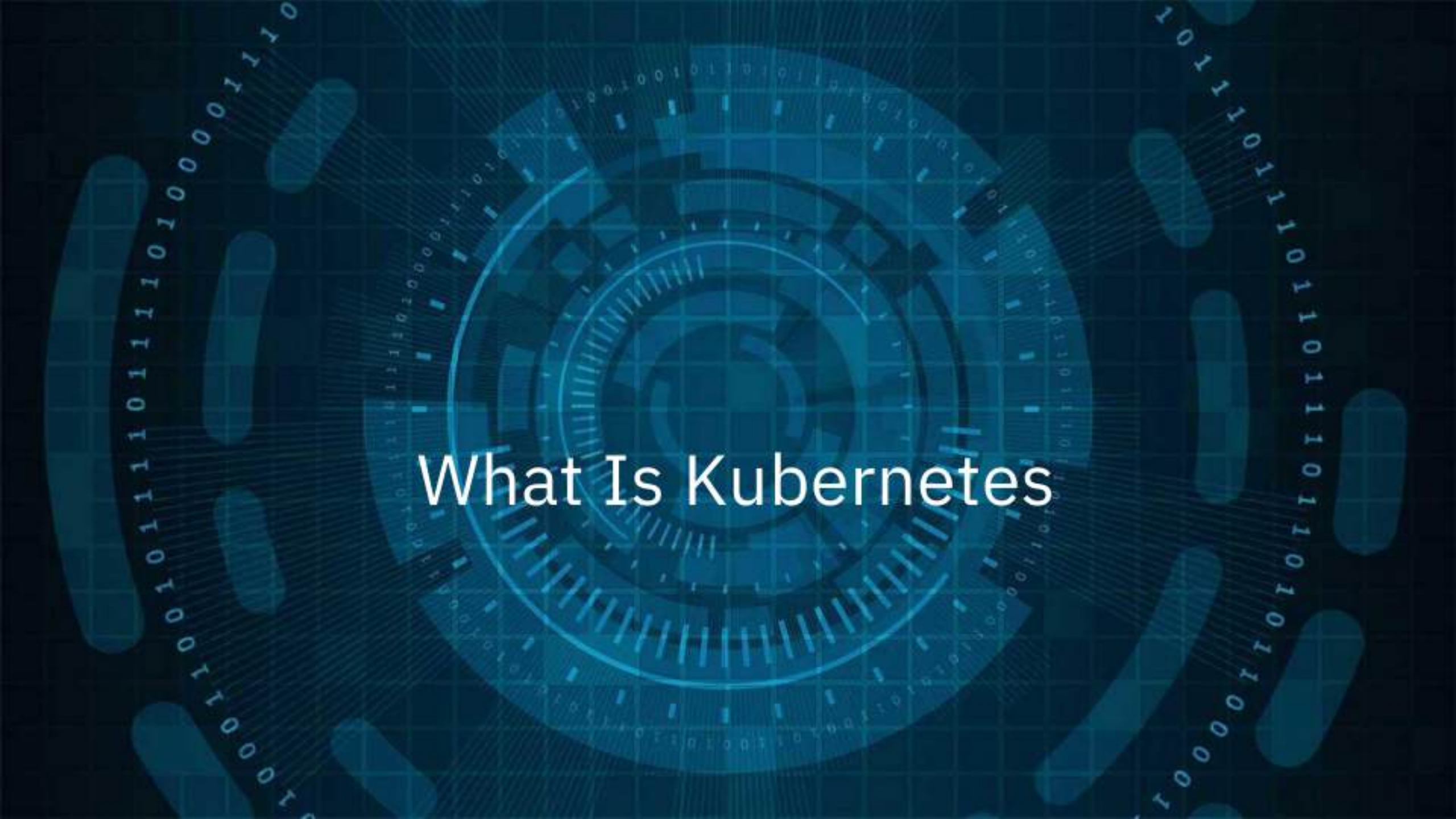
- Ingress Explained
- Deploy Nginx Ingress Controller
- Ingress Demo

0



Kubernetes From Scratch - Hands-on

02 - What Is Kubernetes


What Is Kubernetes

What is Kubernetes

Container Orchestration

It's an open source container orchestration platform, to automate and manage containerized applications.

Developed at Google

It was developed at Google and released as open source in 2014, So it combines the best practices of running containerized workloads at google for 15 years.

Cloud Agnostic

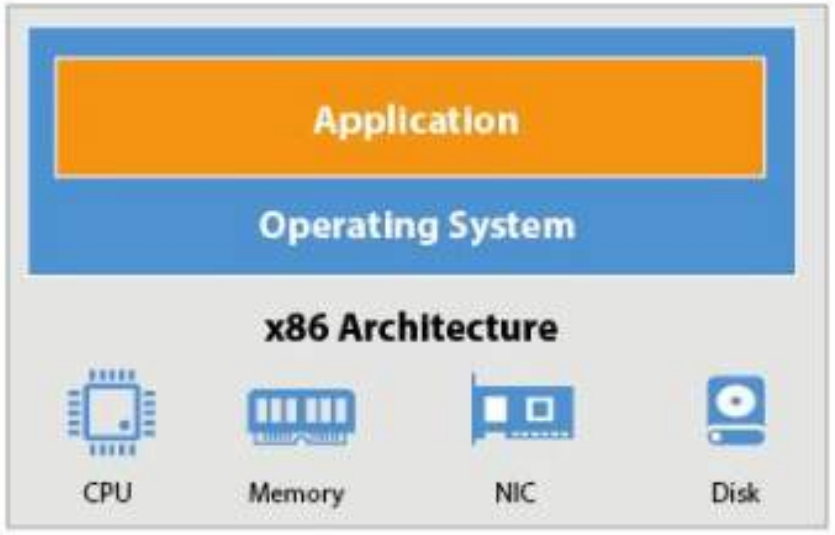
you can run it anywhere on bare metal or in any cloud provider infrastructure.

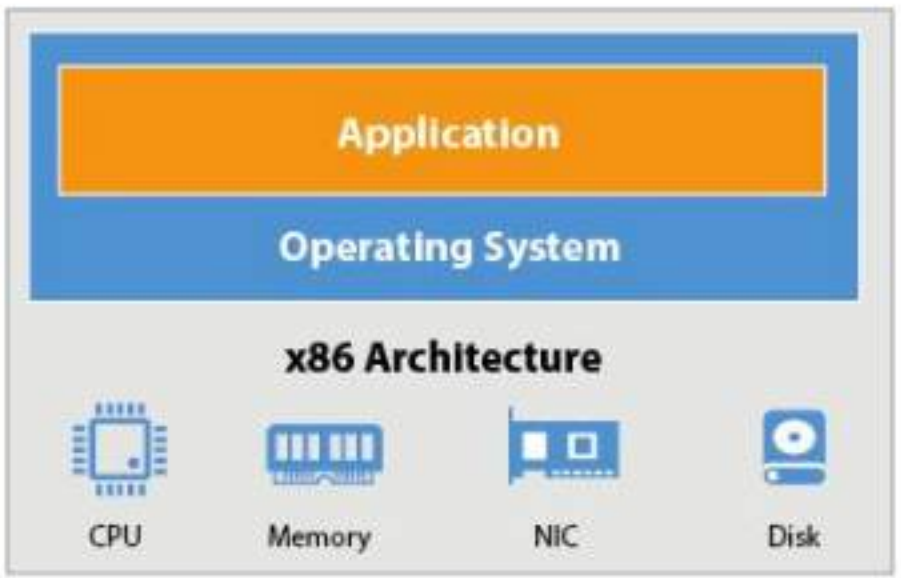
Zero Downtime Deployment

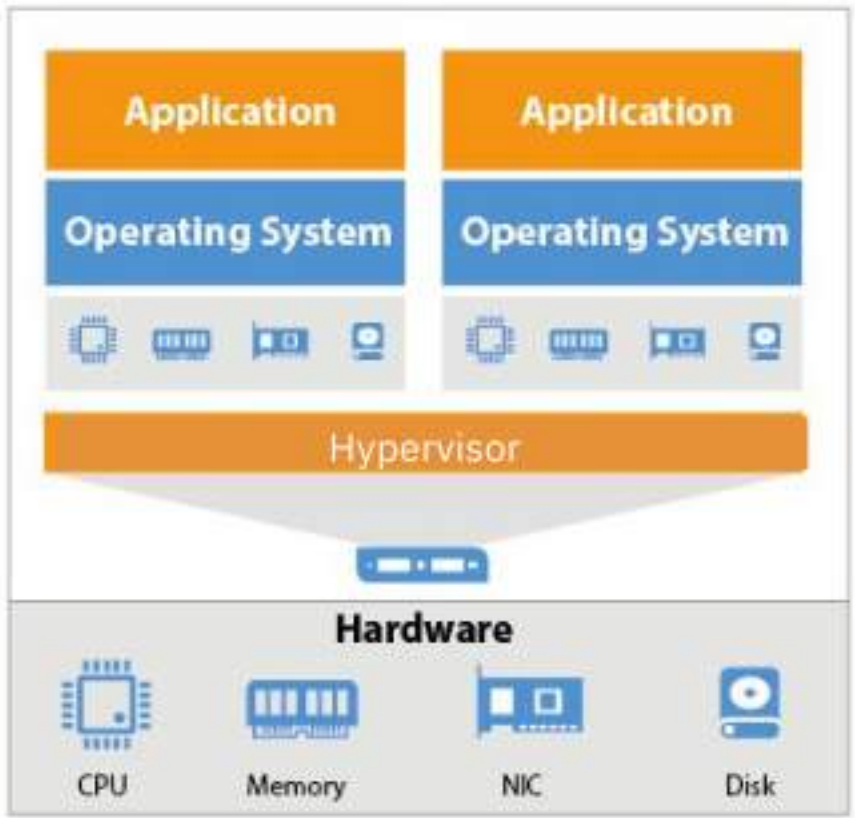
It became the industry standard for deploying containers in production

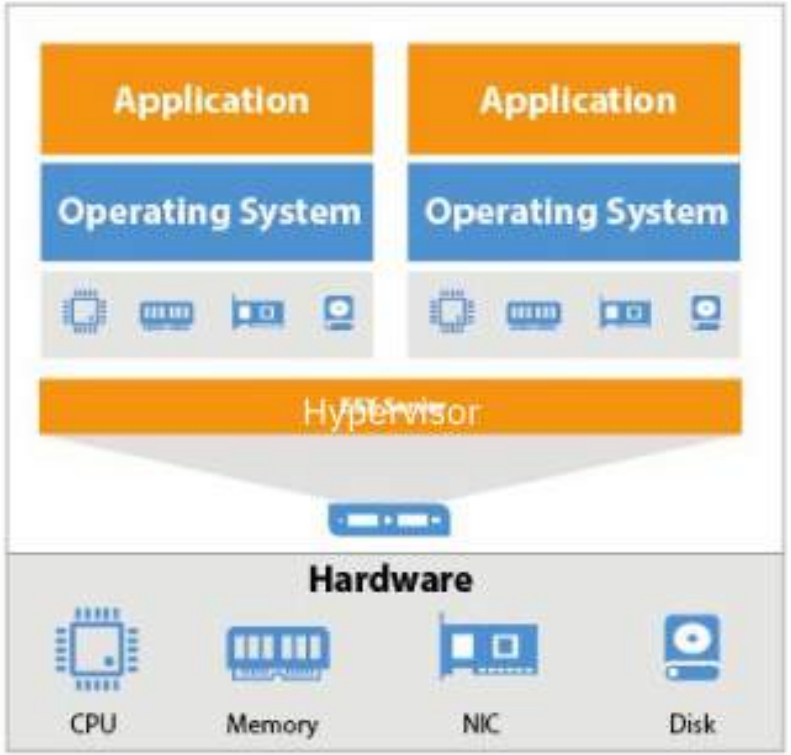
Container Orchestration

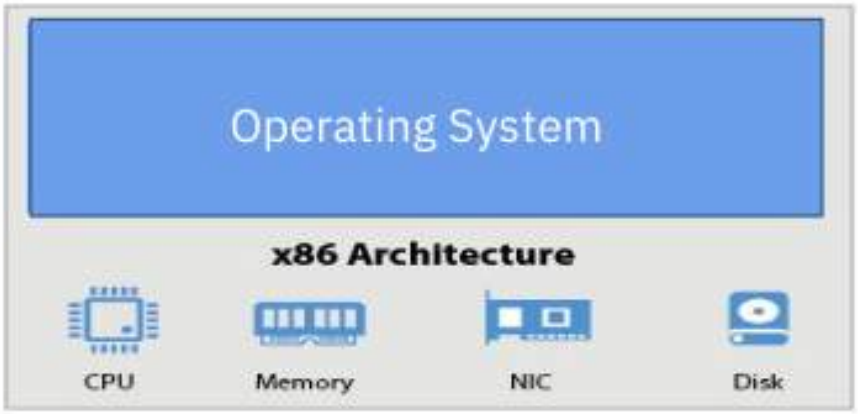
It's an open source container orchestration platform, to automate and manage containerized applications.

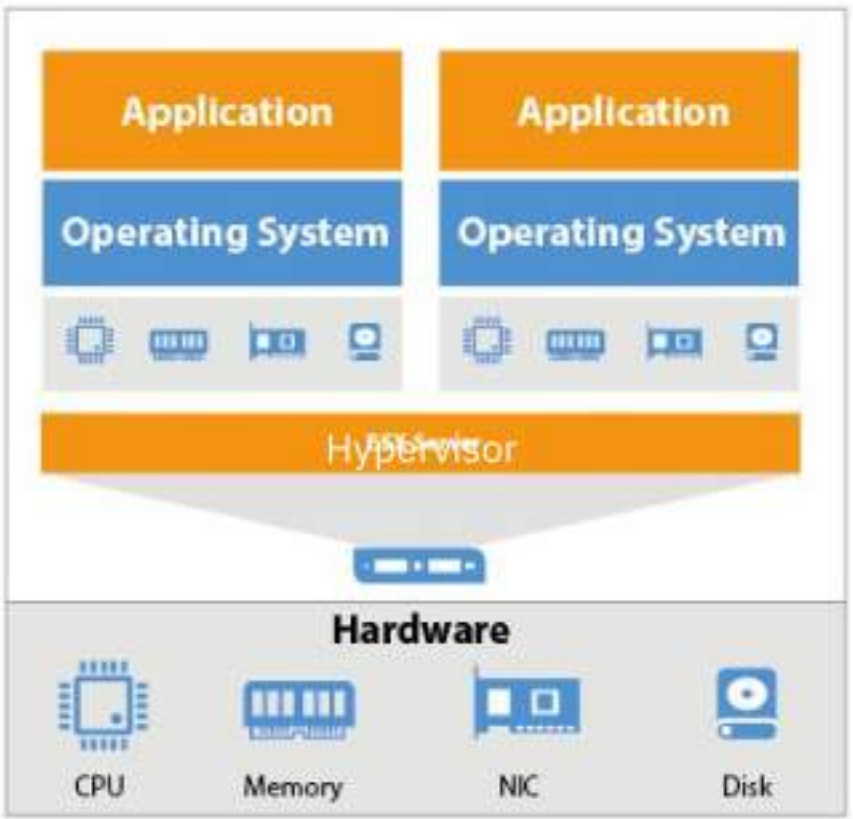

Application
Operating System
x86 Architecture
CPU
Memory
NIC
Disk

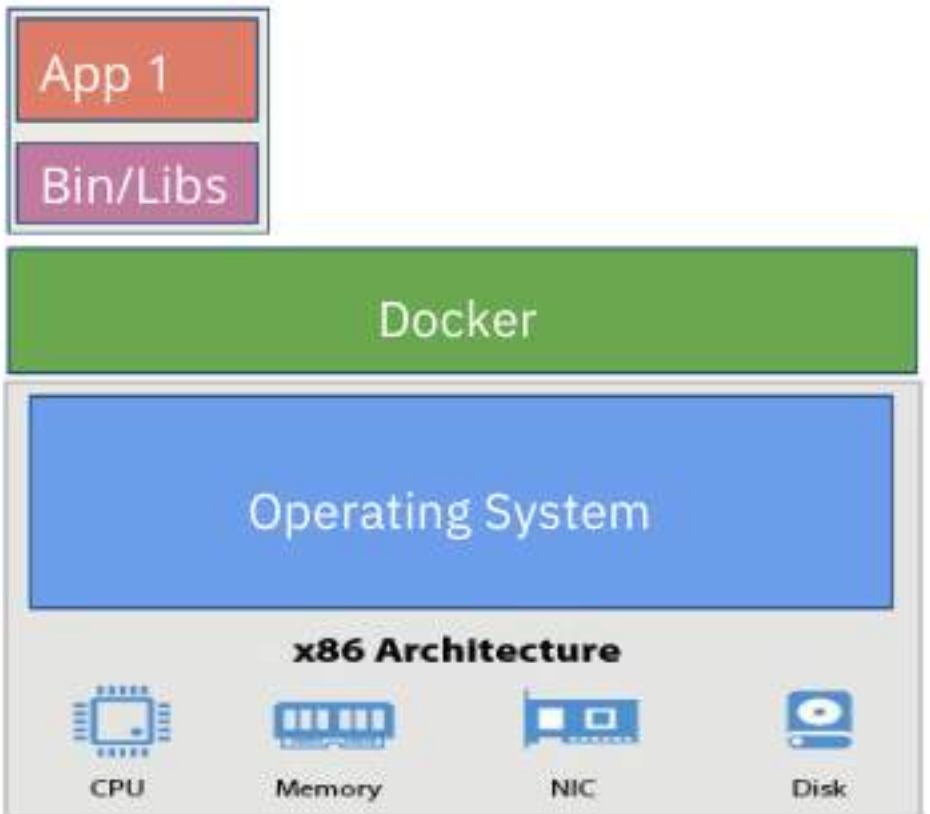

Application
Operating System
x86 Architecture
CPU
Memory
NIC
Disk

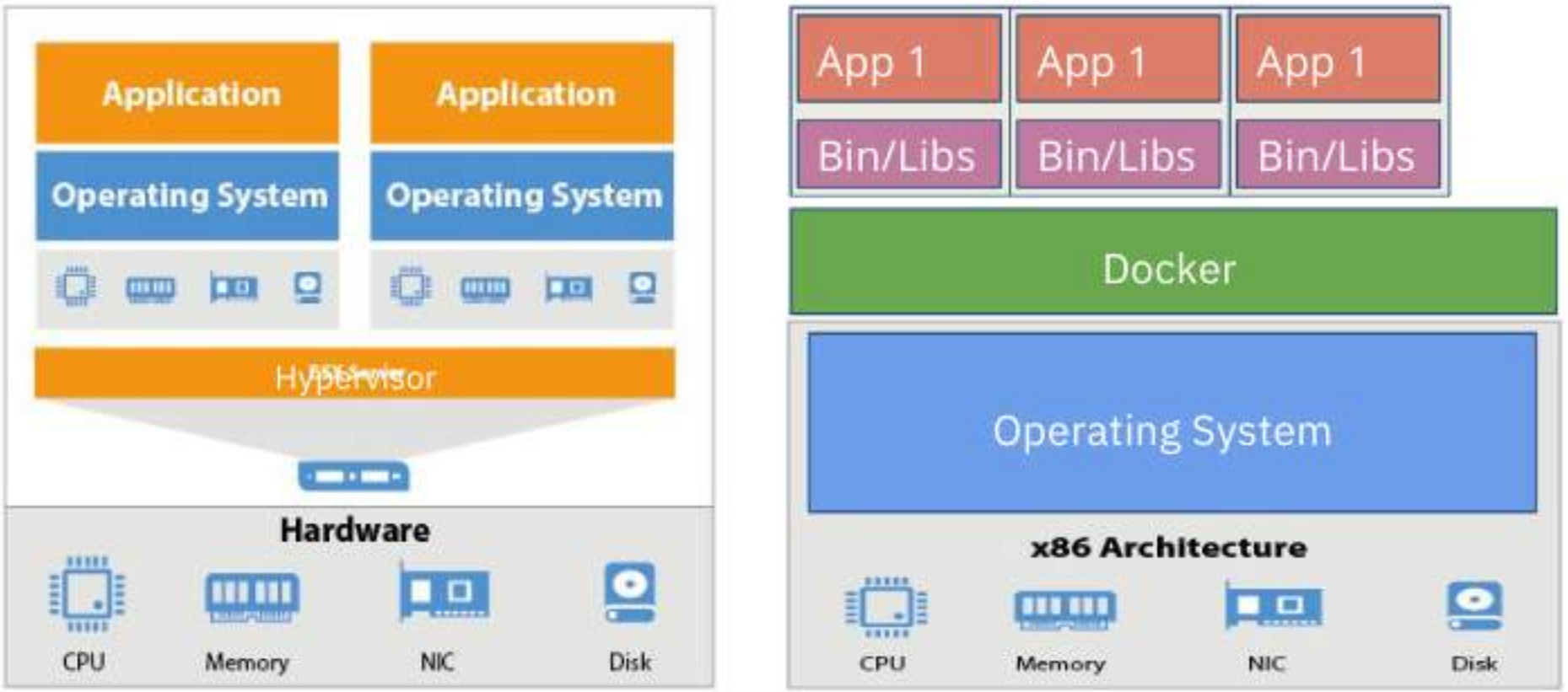

Application
Application
Operating System
Operating System
Hypervisor
Hardware
CPU
Memory
NIC
Disk


Application
Application
Operating System
Operating System
Hypervisor
Hardware
CPU
Memory
NIC
Disk

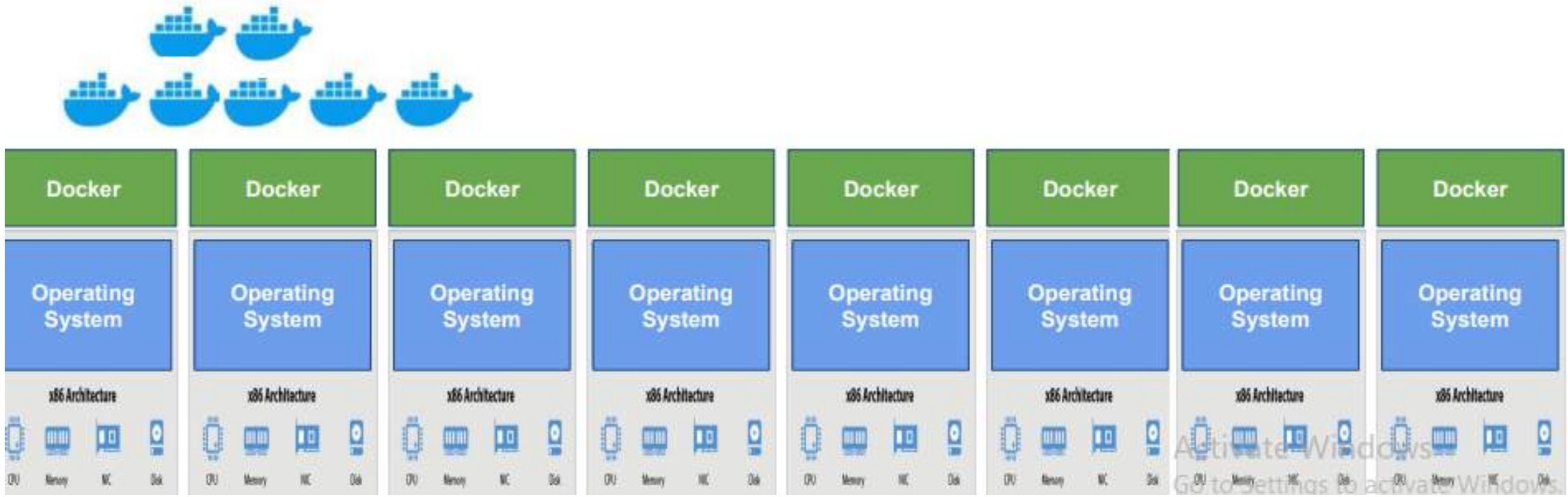

Operating System
x86 Architecture
CPU
Memory
NIC
Disk


Application
Application
Operating System
Operating System
Hypervisor
Hardware
CPU
Memory
NIC
Disk

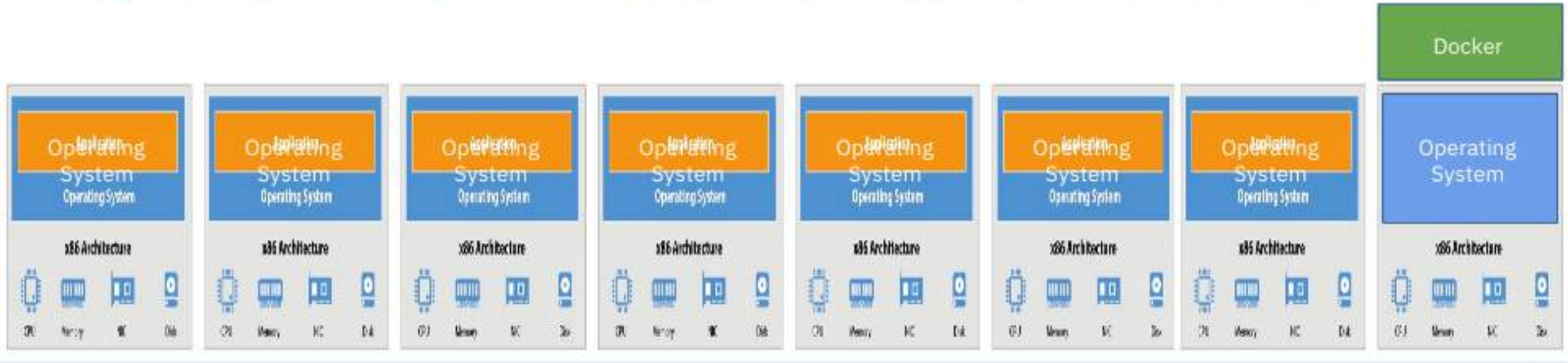

App 1
Bin/Libs
Docker
Operating System
x86 Architecture
CPU
Memory
NIC
Disk

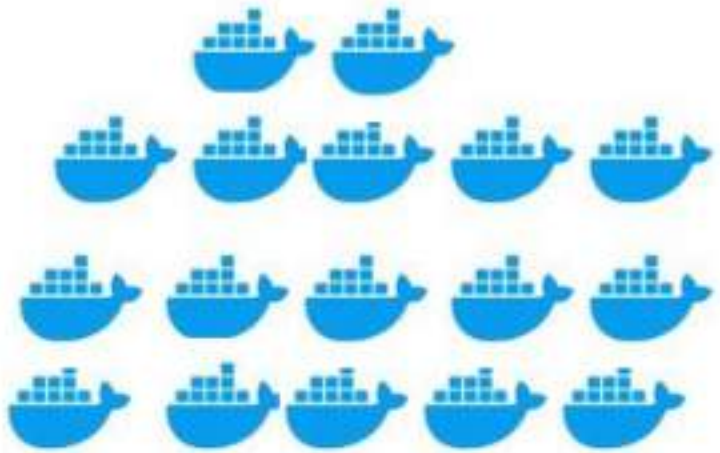

Application
Application
Operating System
Operating System
Hypervisor
Hardware
CPU
Memory
NIC
Disk
App 1
App 1
App 1
Bin/Libs
Bin/Libs
Bin/Libs
Docker
Operating System
x86 Architecture
CPU
Memory
NIC
Disk

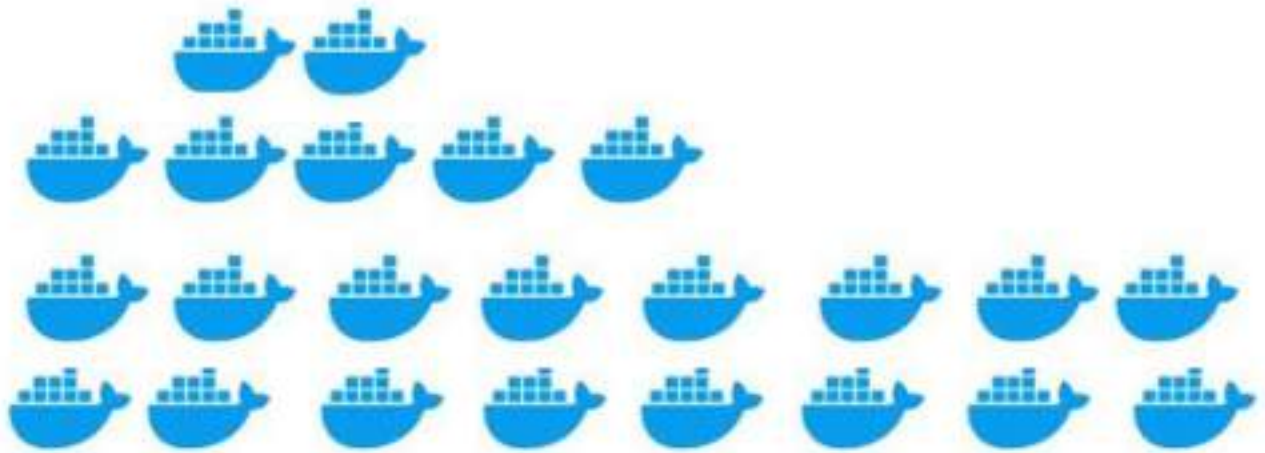

Docker
Operating System
x86 Architecture
CPU
Memory
NIC
Disk



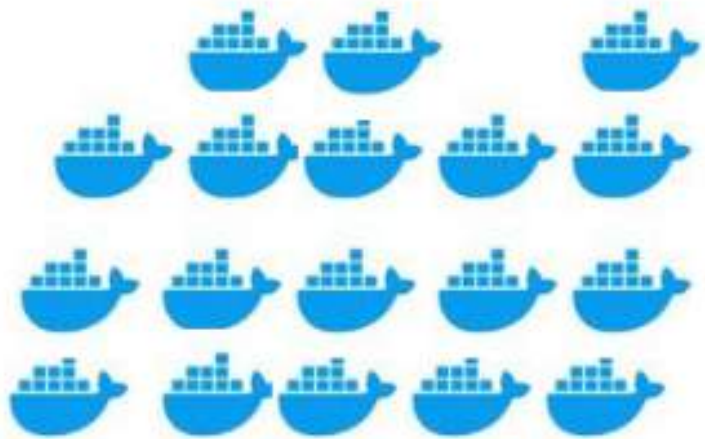

Docker
Operating System
x86 Architecture

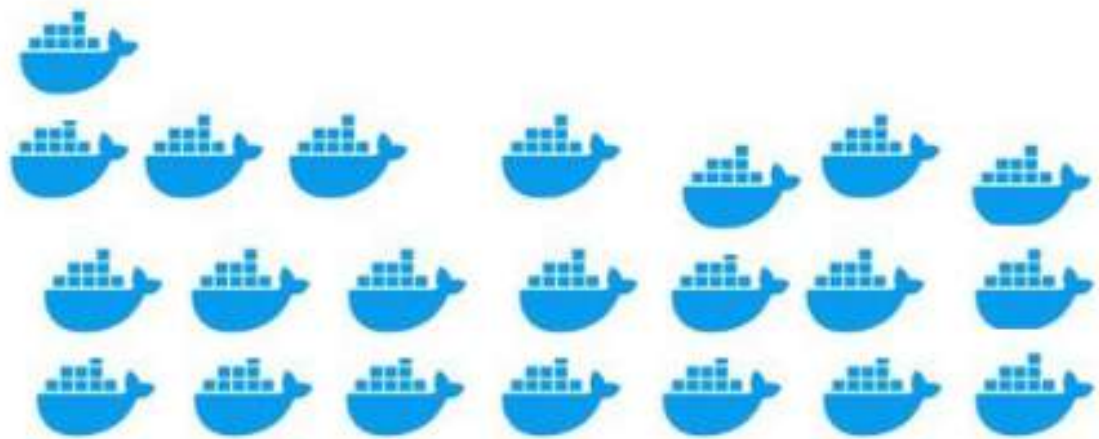

Docker
Operating System
x86 Architecture



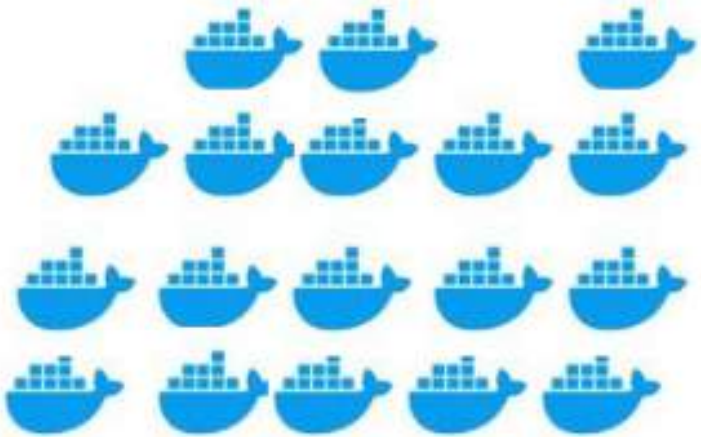


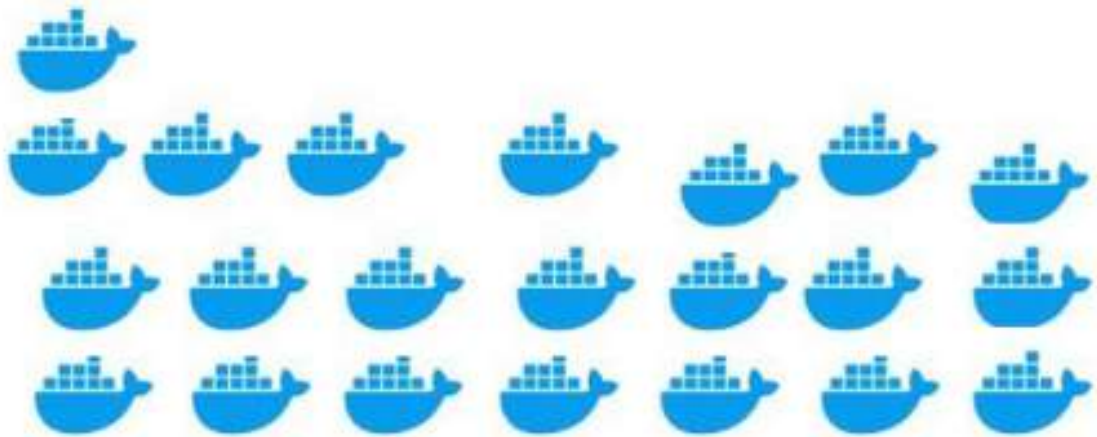

Docker
Operating System
x86 Architecture






Docker
Operating System
x86 Architecture






Docker
Operating System
x86 Architecture

Final Note

01 What does Kubernetes Provide for the Infrastructure

Automated Infrastructure

Compute Networking Storage Firewall Monitoring Alerting

02 What does Kubernetes Provide for the Developer

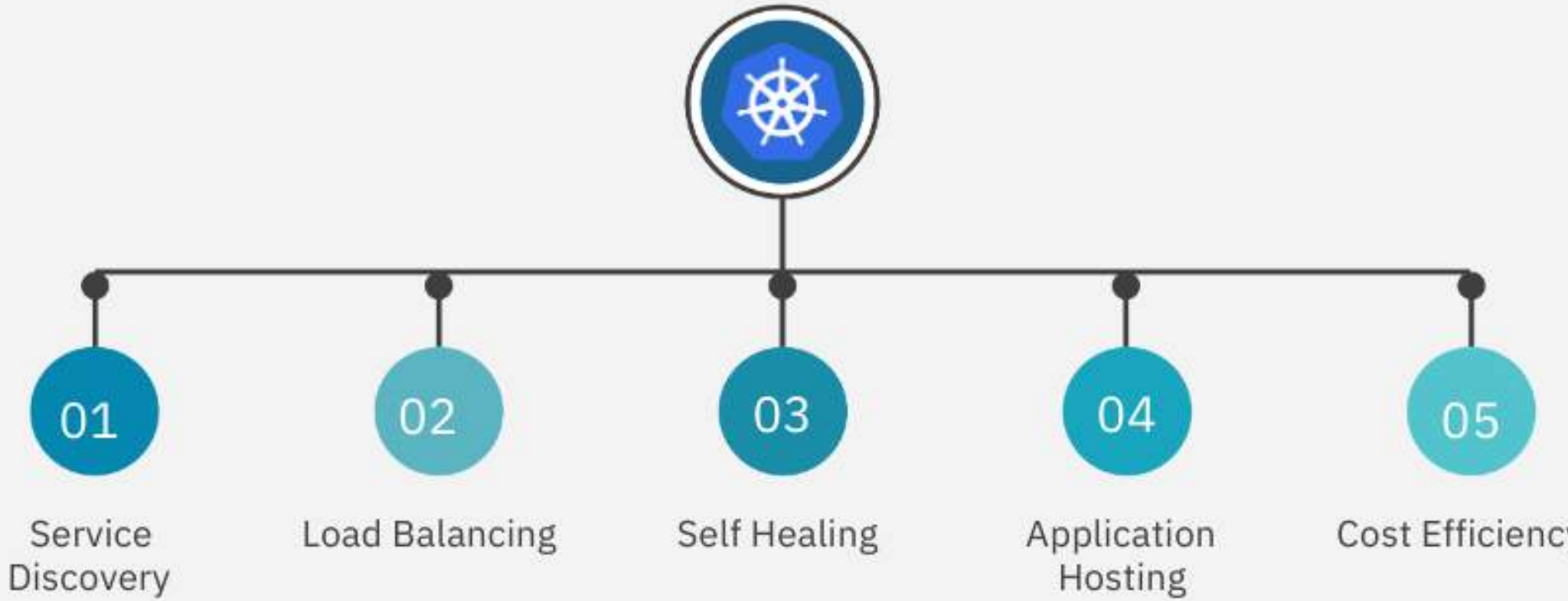
Automated Deployment

Deployment Self Healing Load Balancing Service Discovery Monitoring Logging Alerting

Kubernetes From Scratch - Hands-on

03 - Kubernetes Features

Kubernetes Features



Kubernetes Features

01

Service Discovery

A way to connect different services or components with a single stable endpoint

02

Load Balancing

Kubernetes automatically distributes traffic between different application backends to optimize performance and availability.

03

Self Healing

Kubernetes ensures that the Desired state and Actual state are always in sync by monitoring the container healthcheck and restarting the containers which fail this healthcheck.

04

Application Hosting

Kubernetes hosts containerized applications. So the developers doesn't care whether kubernetes is running on bare metal or on the gloud

Kubernetes Features

05

Faster Deployment

Kubernetes allows you to deliver a self-service Platform-as-a-Service (PaaS) that creates a hardware layer abstraction for development teams.

06

Fine-grained Access Control

Kubernetes RBAC allows the cluster maintainers to configure fine-grained access control to Kubernetes resources

07

Batch Jobs

Kubernetes supports running automated jobs with the CronJob object. They are similar to the Unix/Linux Cron tasks.

08

Autoscaling

One of the cool features of Kubernetes is that it can monitor your workload and scale it up or down based on the CPU utilization or memory consumption

Kubernetes Features

09

Cost Efficiency

Kubernetes optimizes infrastructure cost by efficiently managing the infrastructure resources
Reduction of administration time, etc.

10

Declarative Configuration

We provide the API server with the manifest files (YAML or JSON) describing how we want the Kubernetes cluster to look.

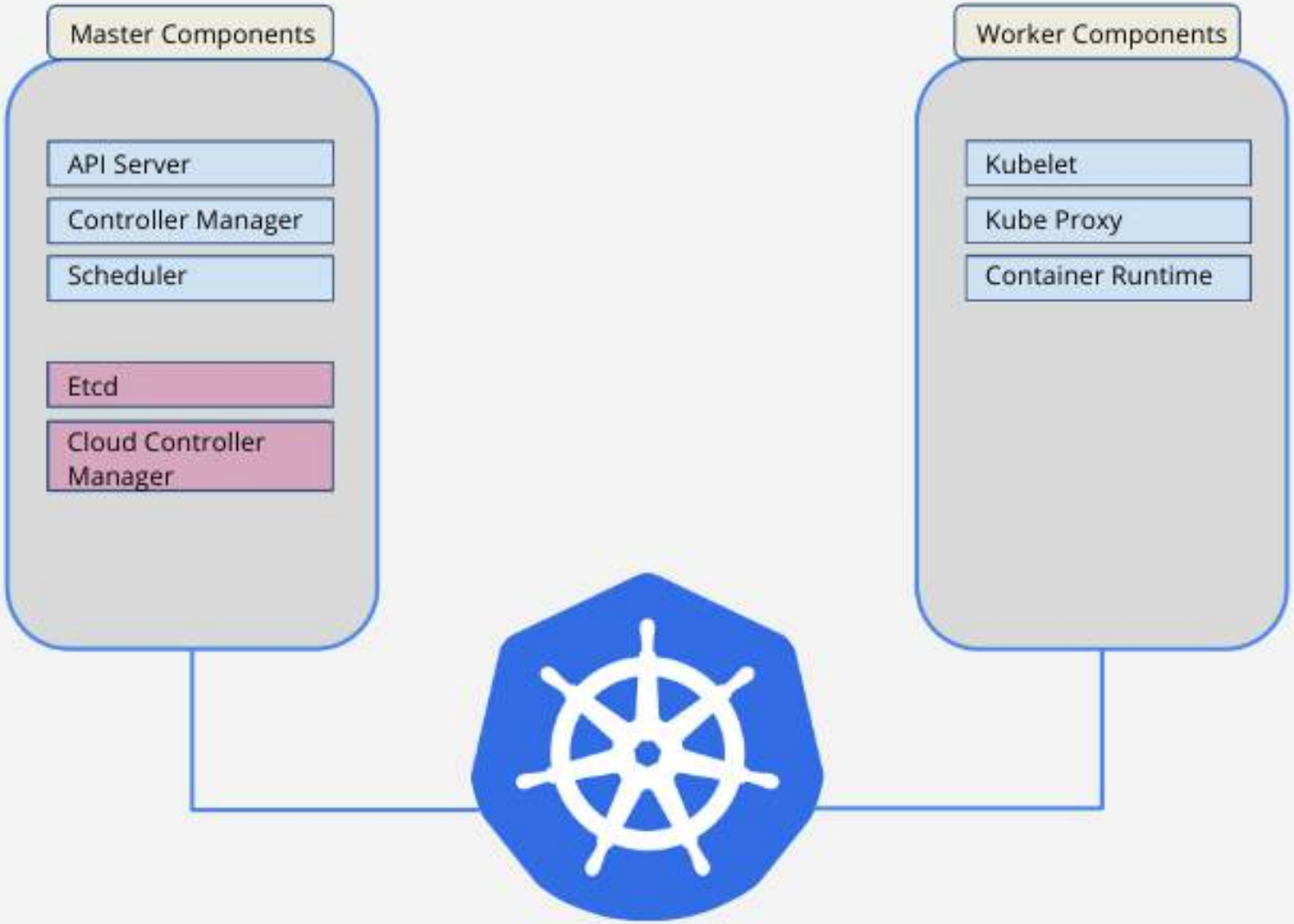
Module 2

Kubernetes Architecture

Kubernetes From Scratch - Hands-on

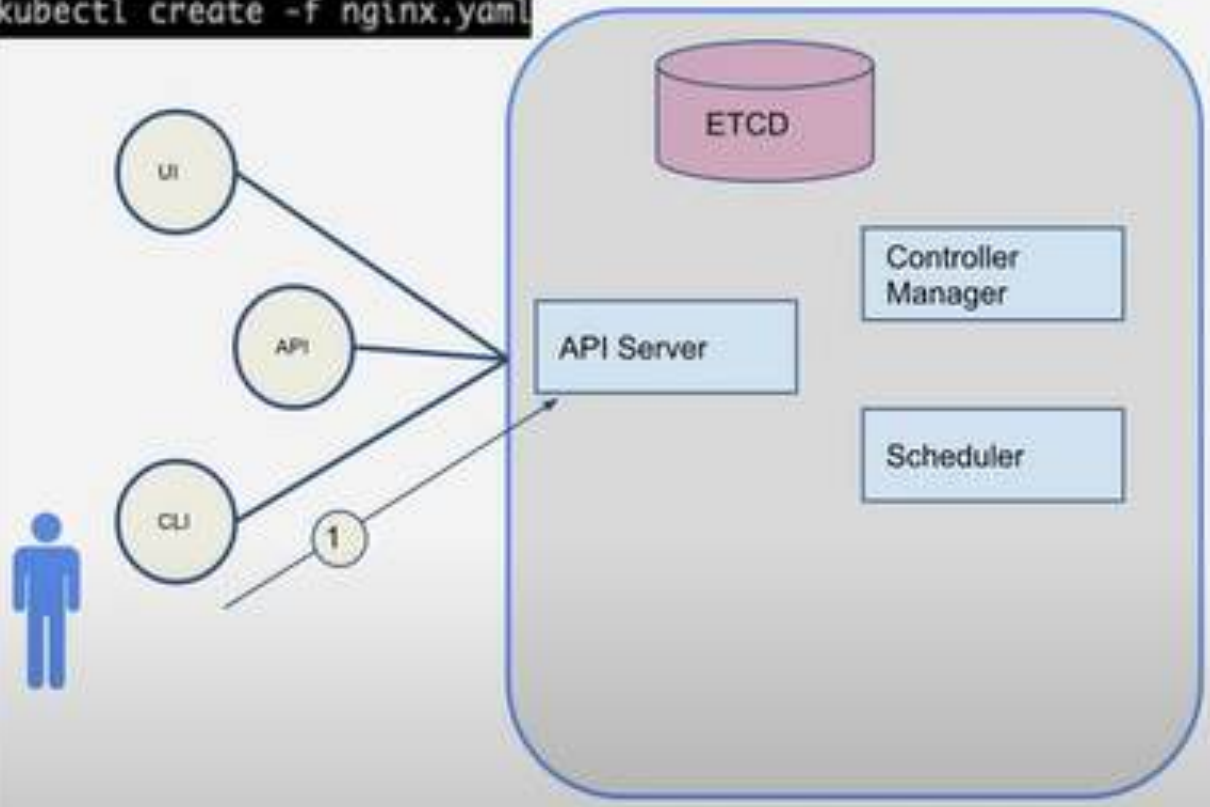
04 - Cluster Architecture

Cluster Architecture



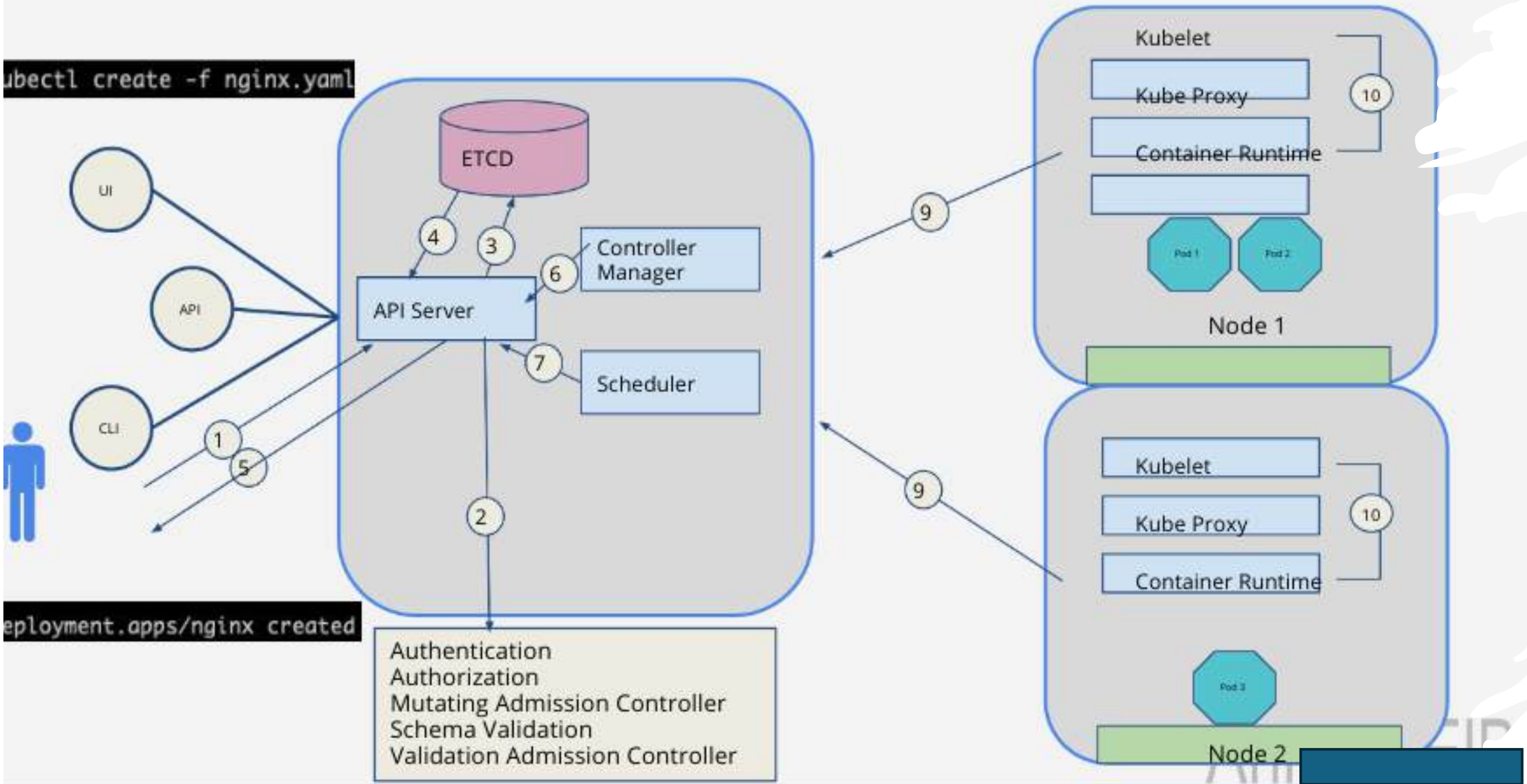
Cluster Architecture

```
kubectl create -f nginx.yaml
```



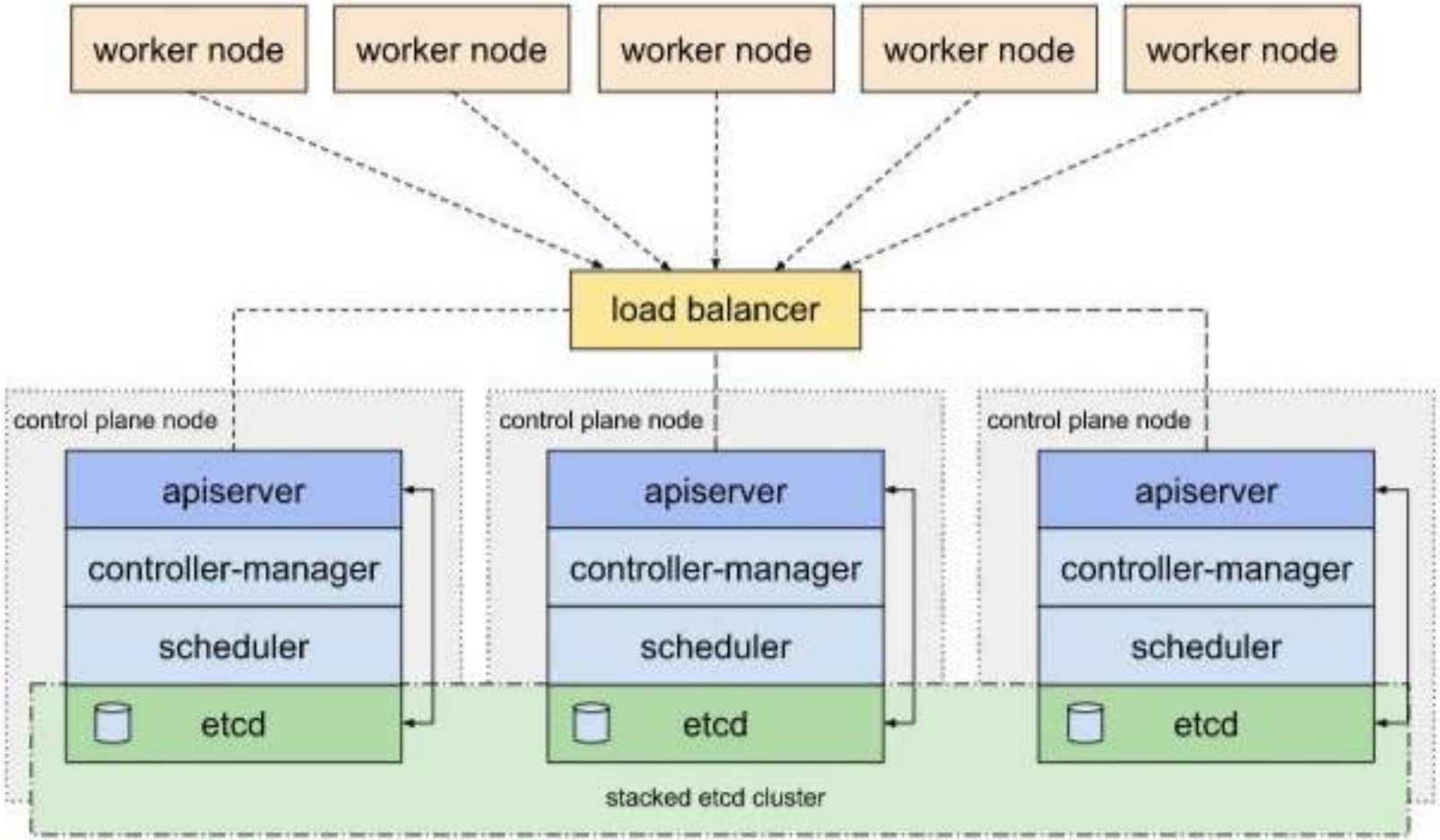

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
```

Cluster Architecture

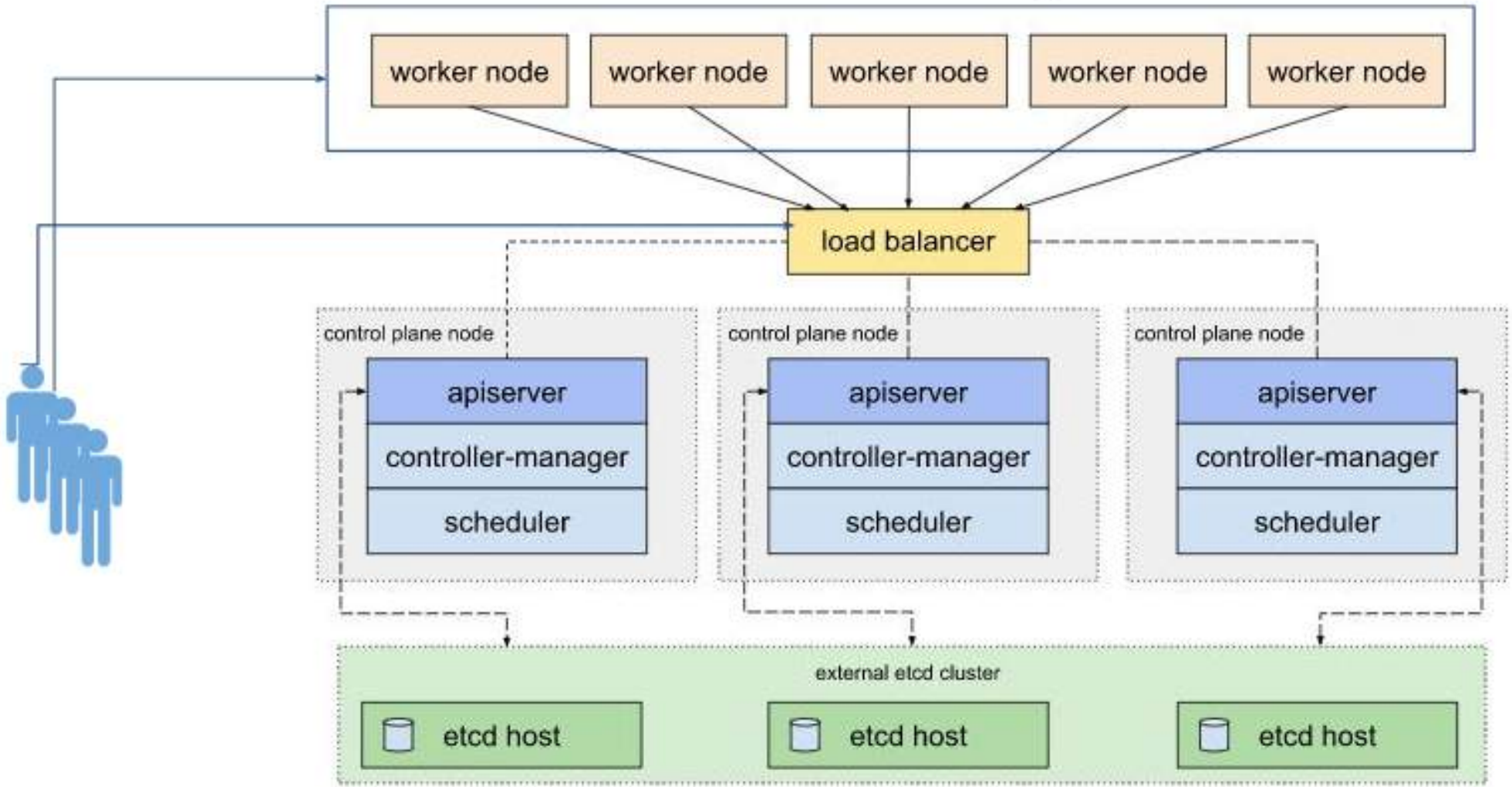



THANK YOU

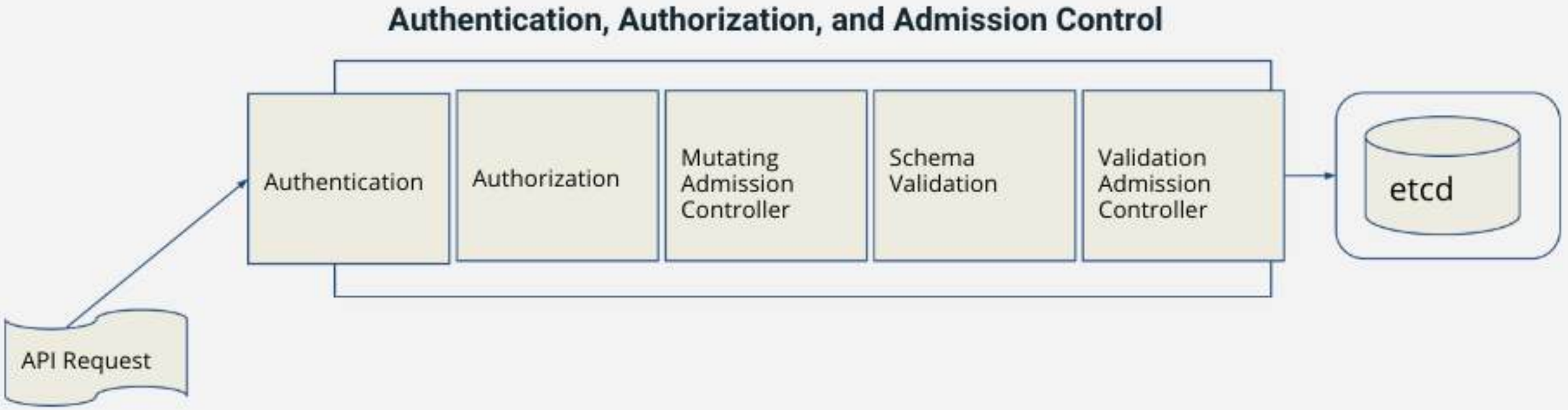
Master Components



Master Components



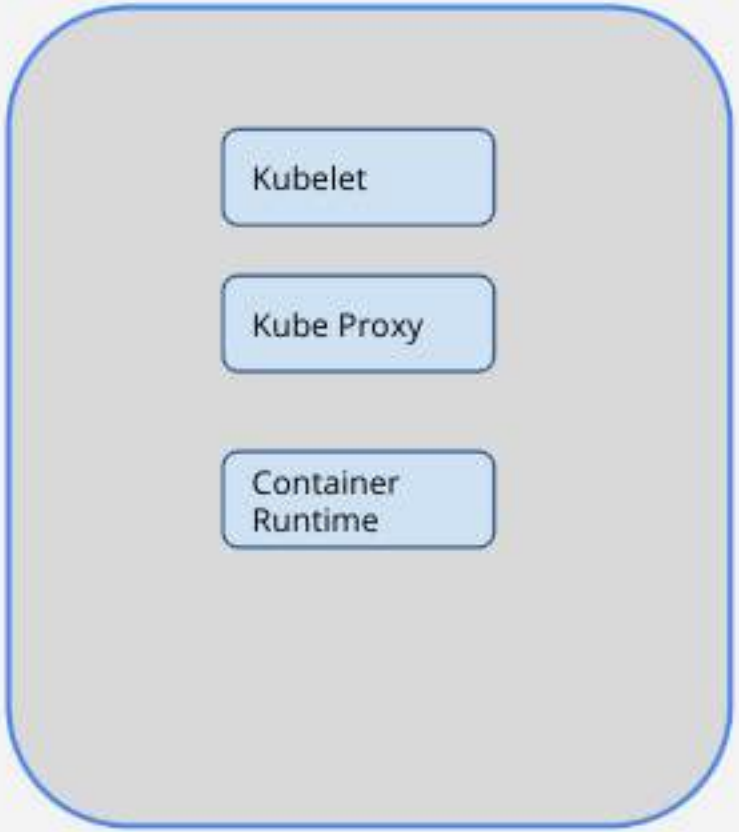
Master Components



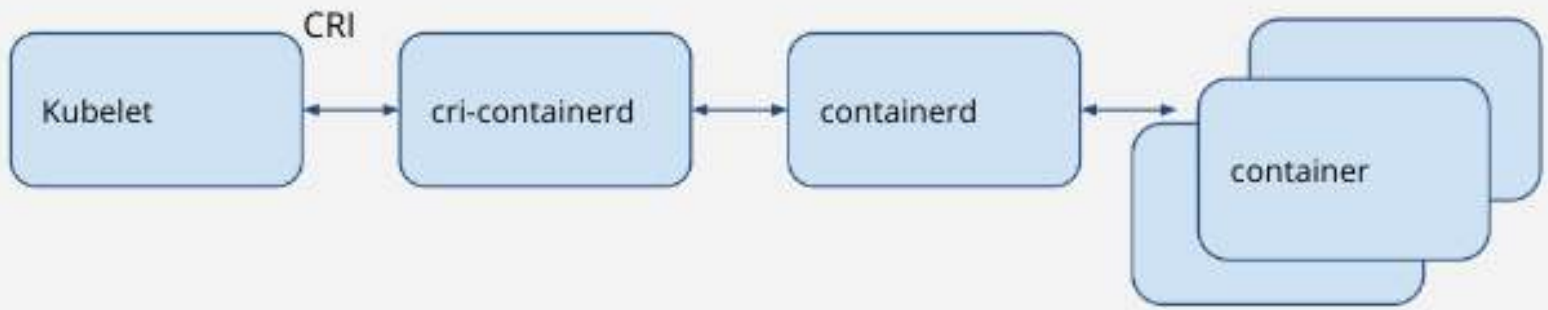
Kubernetes From Scratch - Hands-on

06 - Worker Components

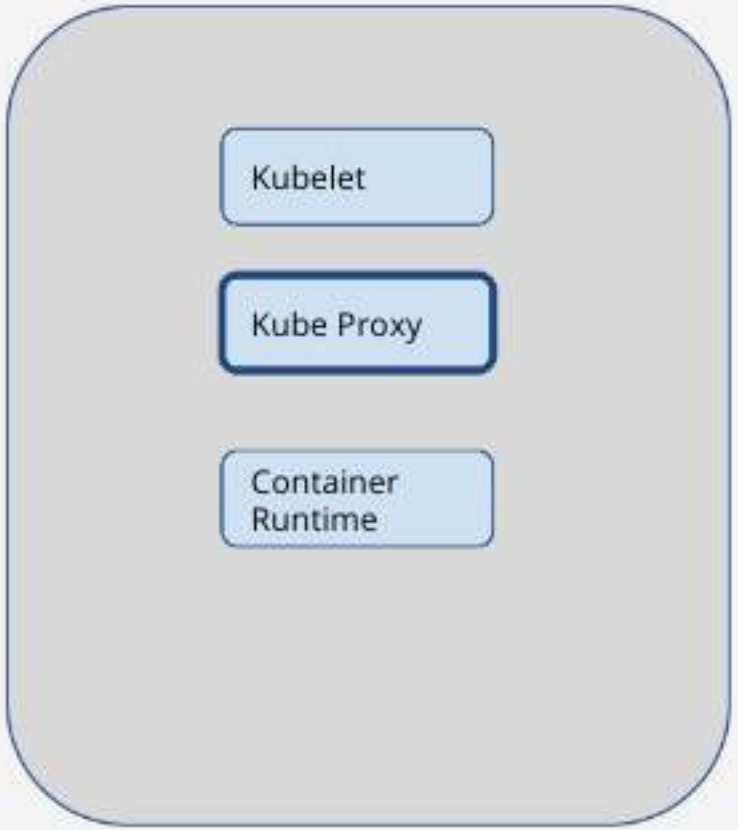
Worker Components



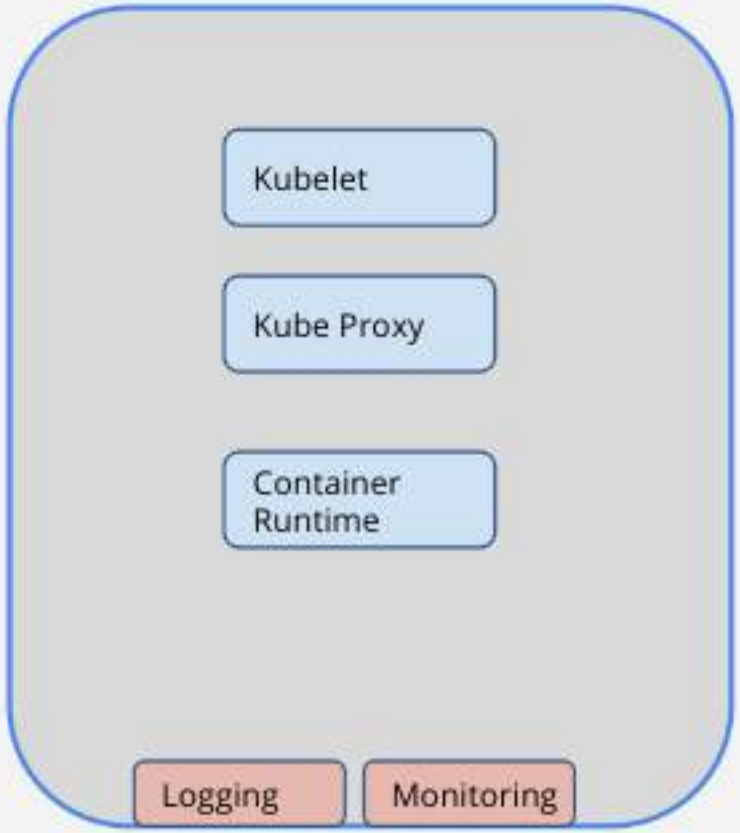
Worker Components



Worker Components



Worker Components

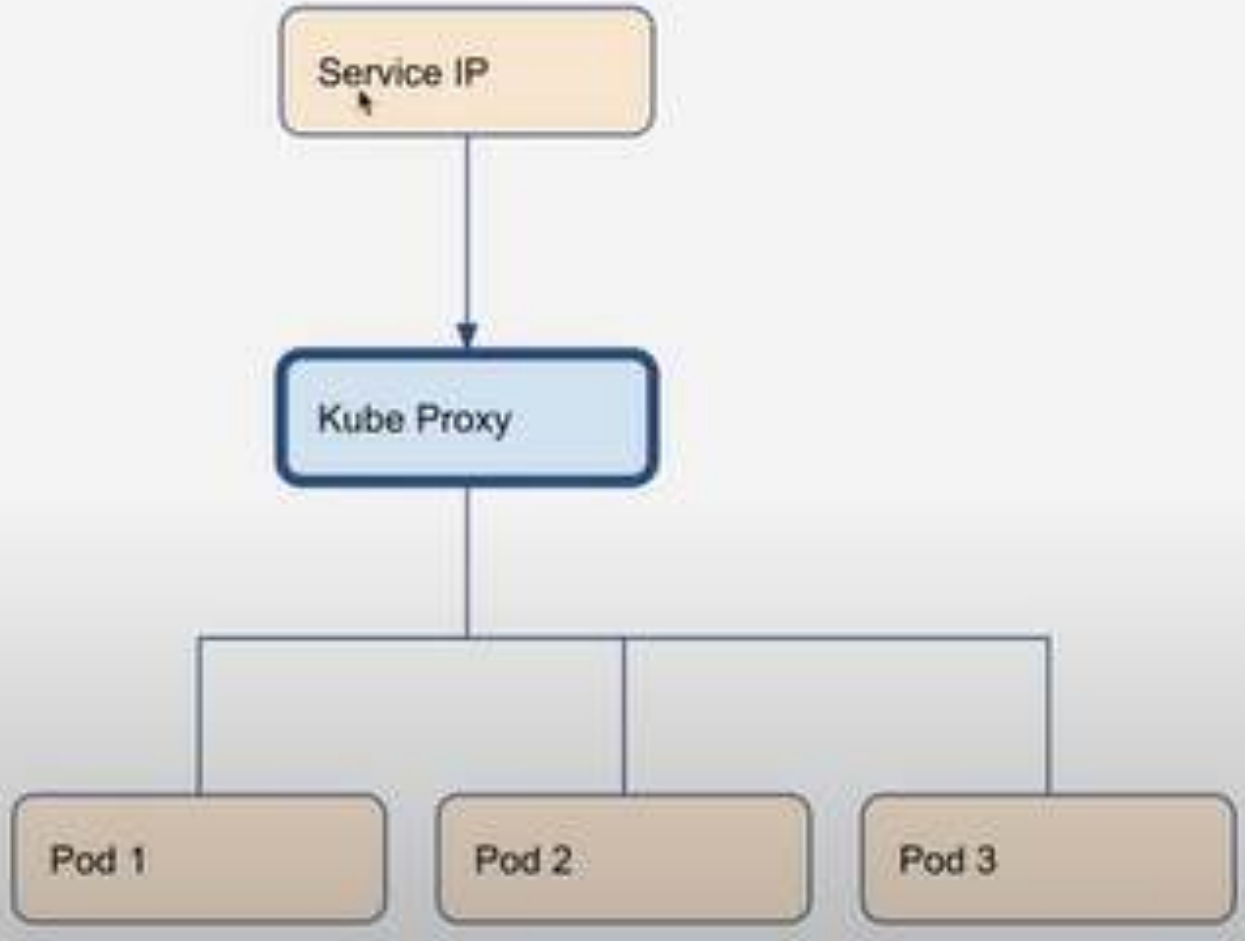



THANK YOU

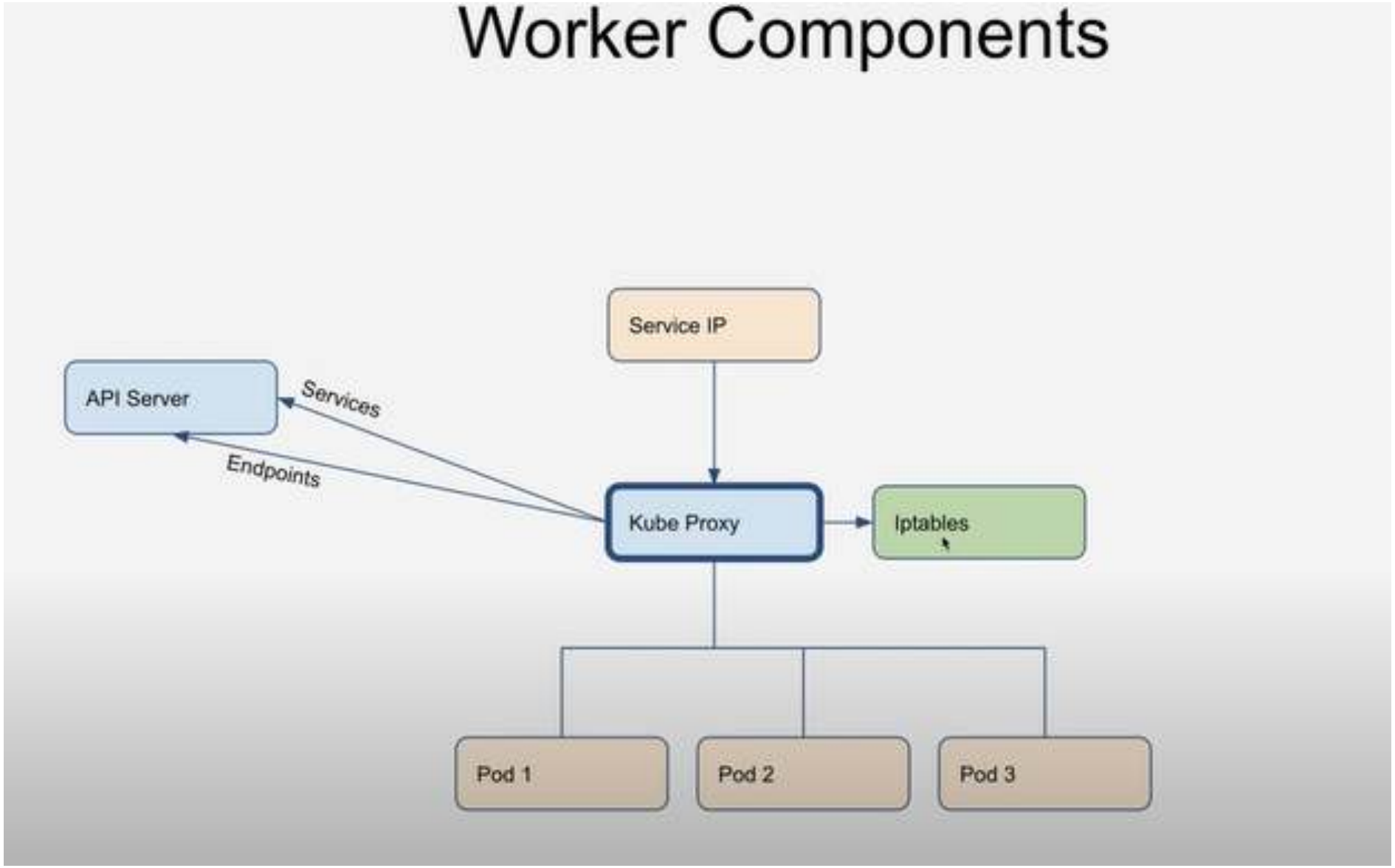
Kubernetes From Scratch - Hands-on

07 - Cluster Architecture Demo

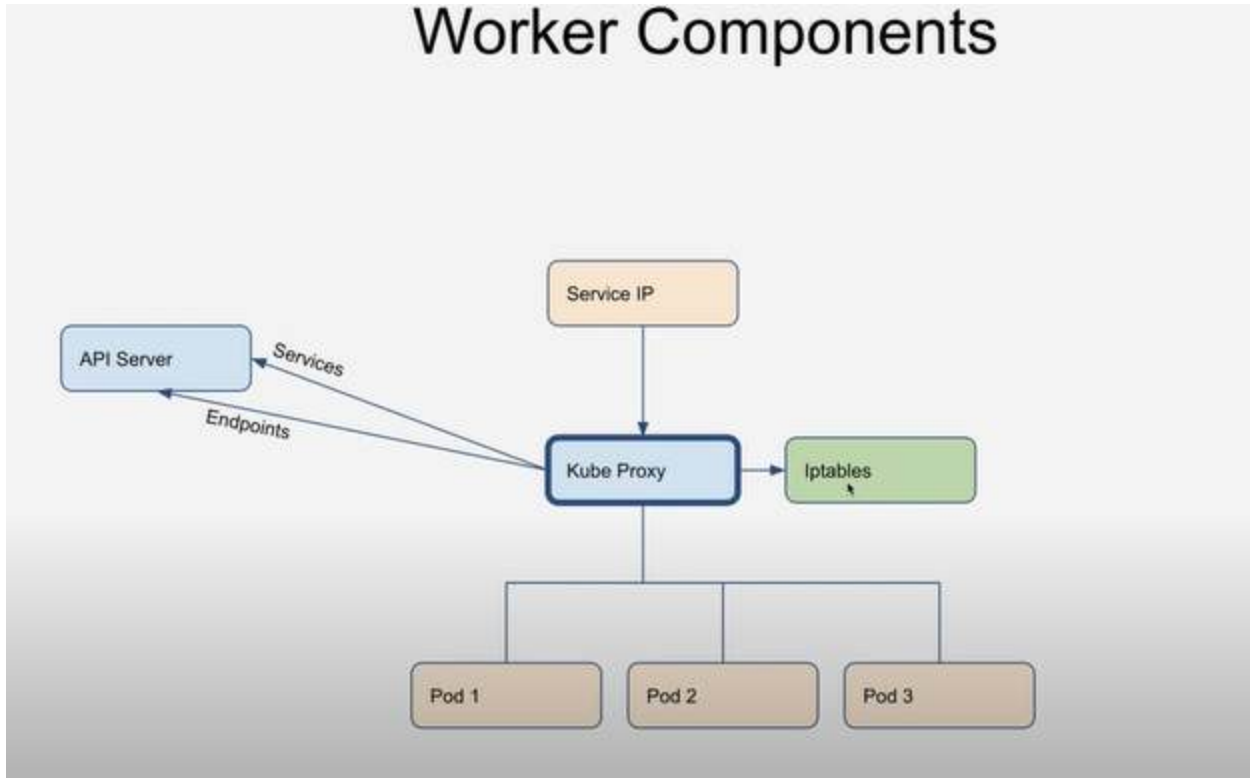
Worker Components



Worker Components



Worker Components



What is Minikube

- Minikube is a tool that makes it easy to run Kubernetes locally
- Minikube runs a single-node Kubernetes cluster inside a Virtual Machine (VM)
- Supports the following features

  - DNS
  - NodePorts
  - ConfigMaps and Secrets
  - Dashboards
  - Container Runtime: Docker, CRI-O, and containerd
  - Enabling CNI (Container Network Interface)
  - Ingress
- Requirements:
  - 2 CPU or more
  - 2GB of free memory
  - 20GB of free disk space
  - Internet connection
  - Container or virtual machine manager, such as: Docker, Hyperkit, Hyper-V, KVM, Parallels, Podman, VirtualBox, or VMWare

What is Kind

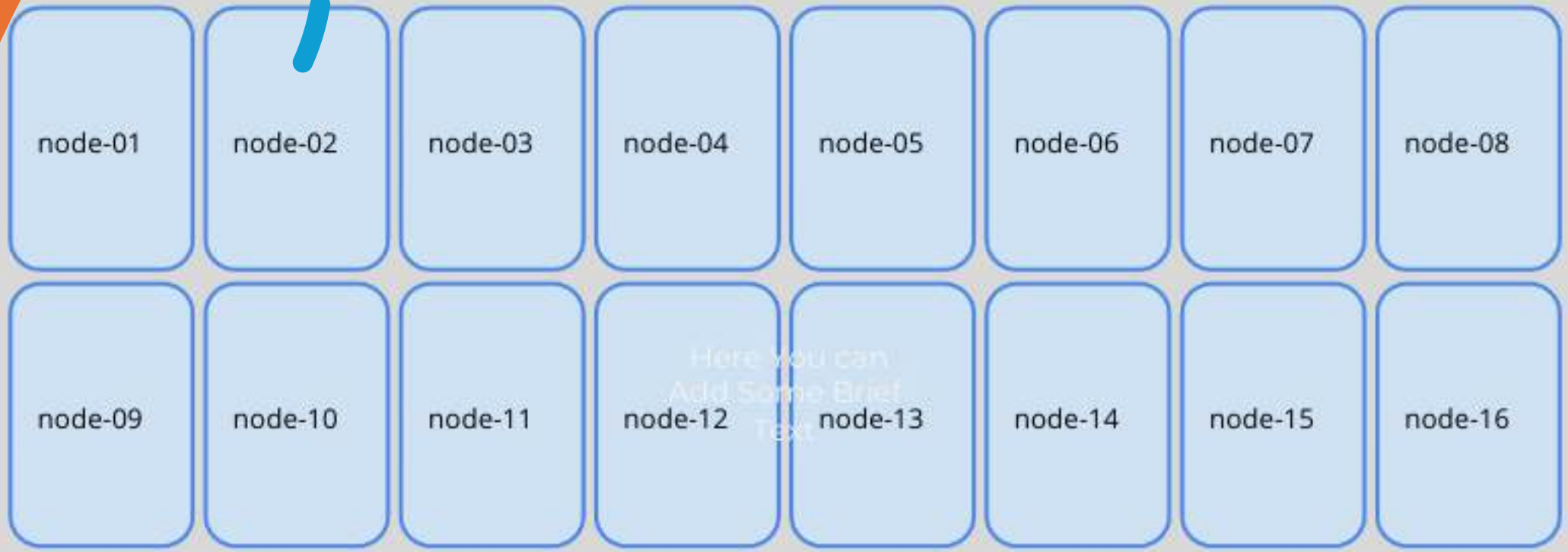
- kind is a tool for running local Kubernetes clusters using Docker container "nodes". kind supports Linux, macOS and Windows Run multiple clusters on different versions. Can be used for

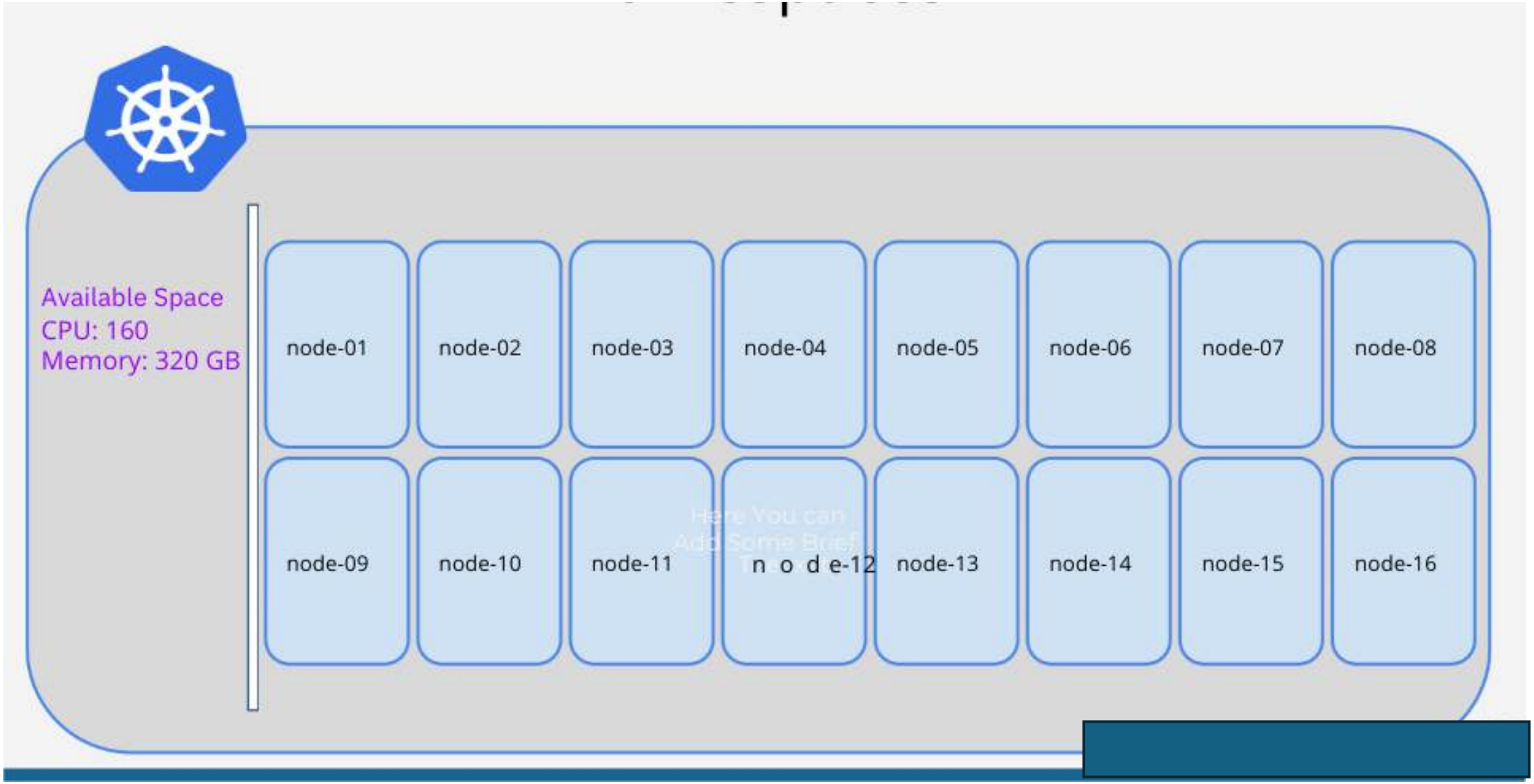
  - Testing Kubernetes upgrades
  - Testing components with Kubernetes new versions
  - Testing with CI/CD

Namespaces

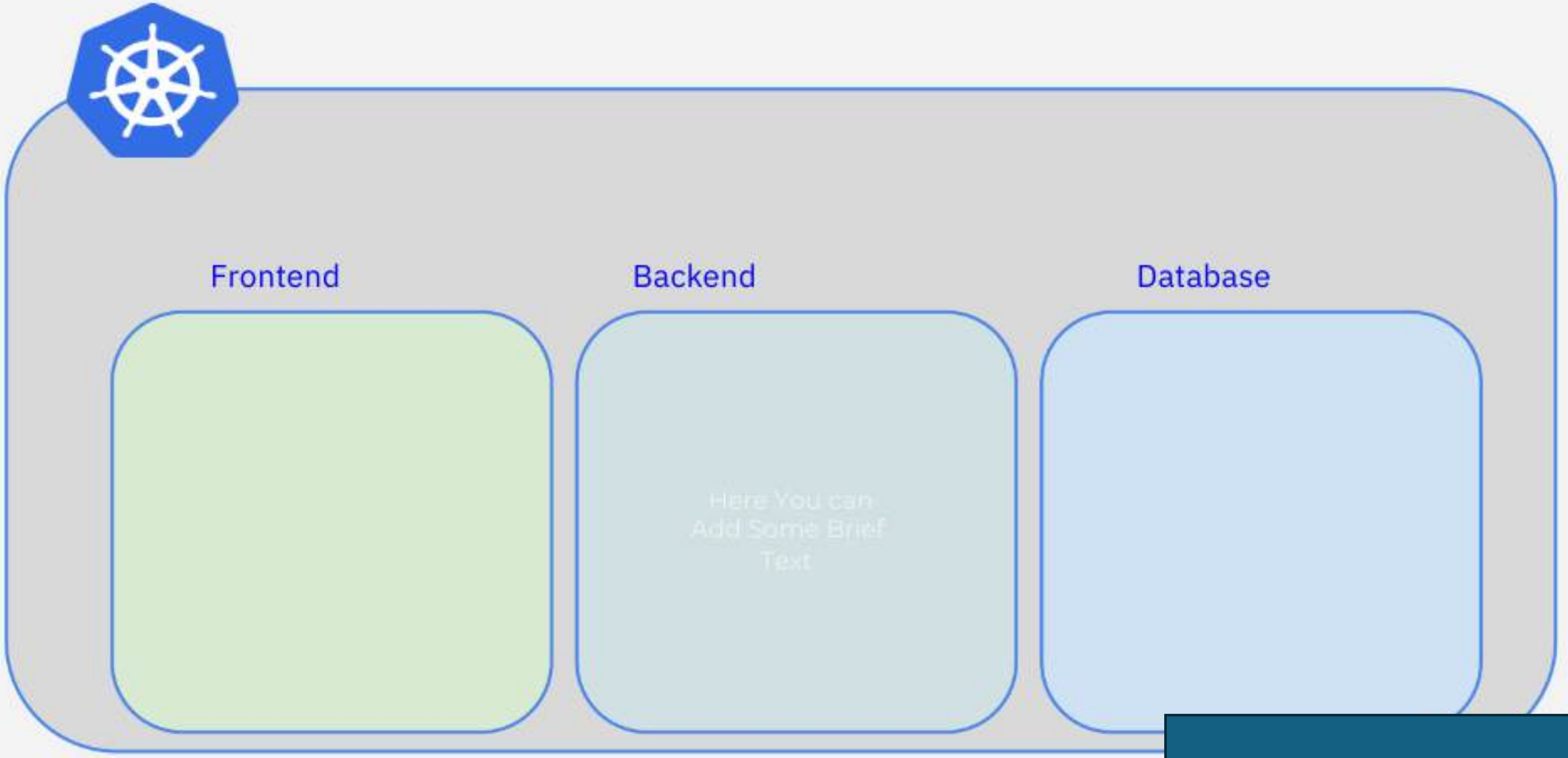
Here You can
Add Some Brief
Text

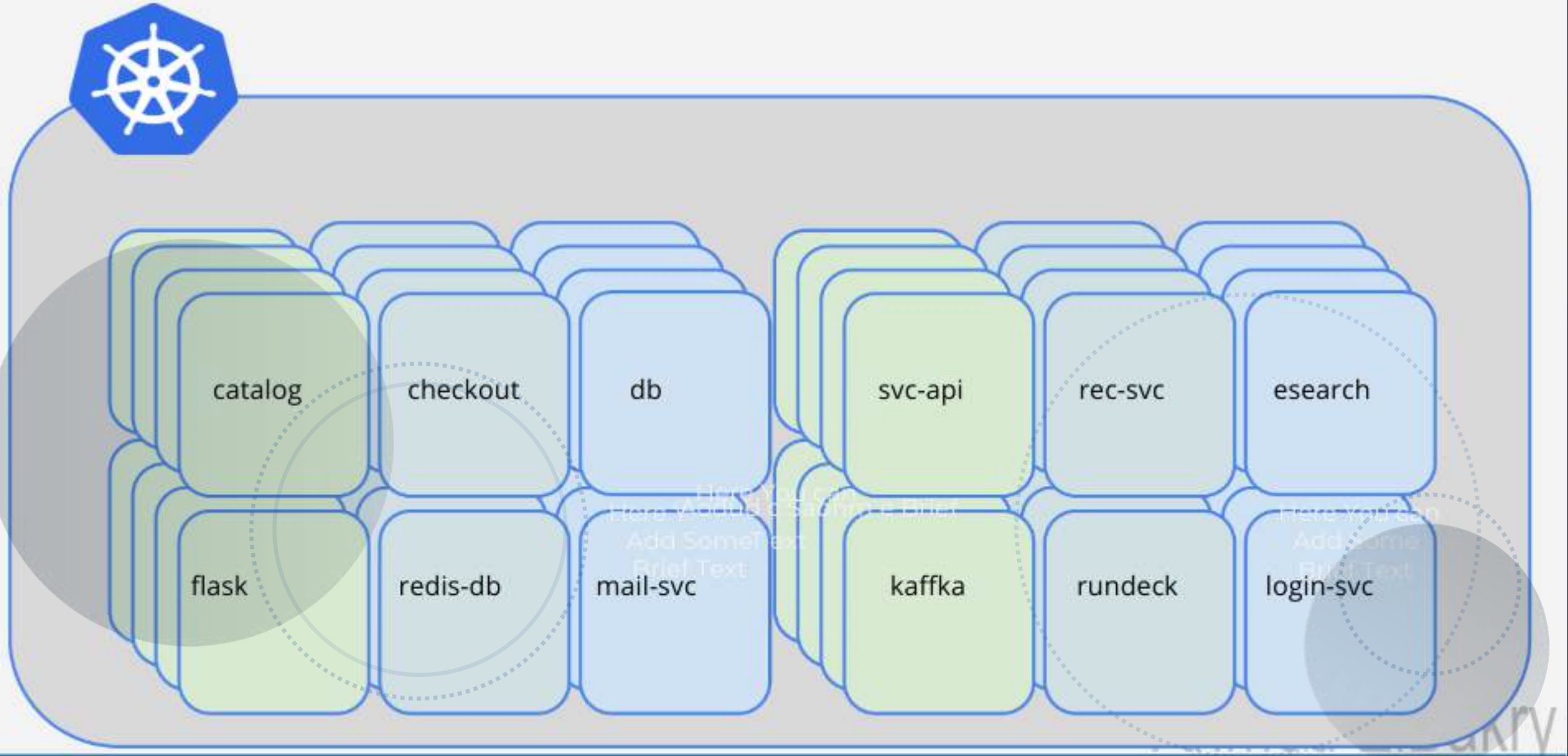
Namespaces


node-01
node-02
node-03
node-04
node-05
node-06
node-07
node-08
node-09
node-10
node-11
node-12
node-13
node-14
node-15
node-16


Available Space
CPU: 160
Memory: 320 GB
node-01
node-02
node-03
node-04
node-05
node-06
node-07
node-08
node-09
node-10
node-11
node-12
node-13
node-14
node-15
node-16

Namespaces

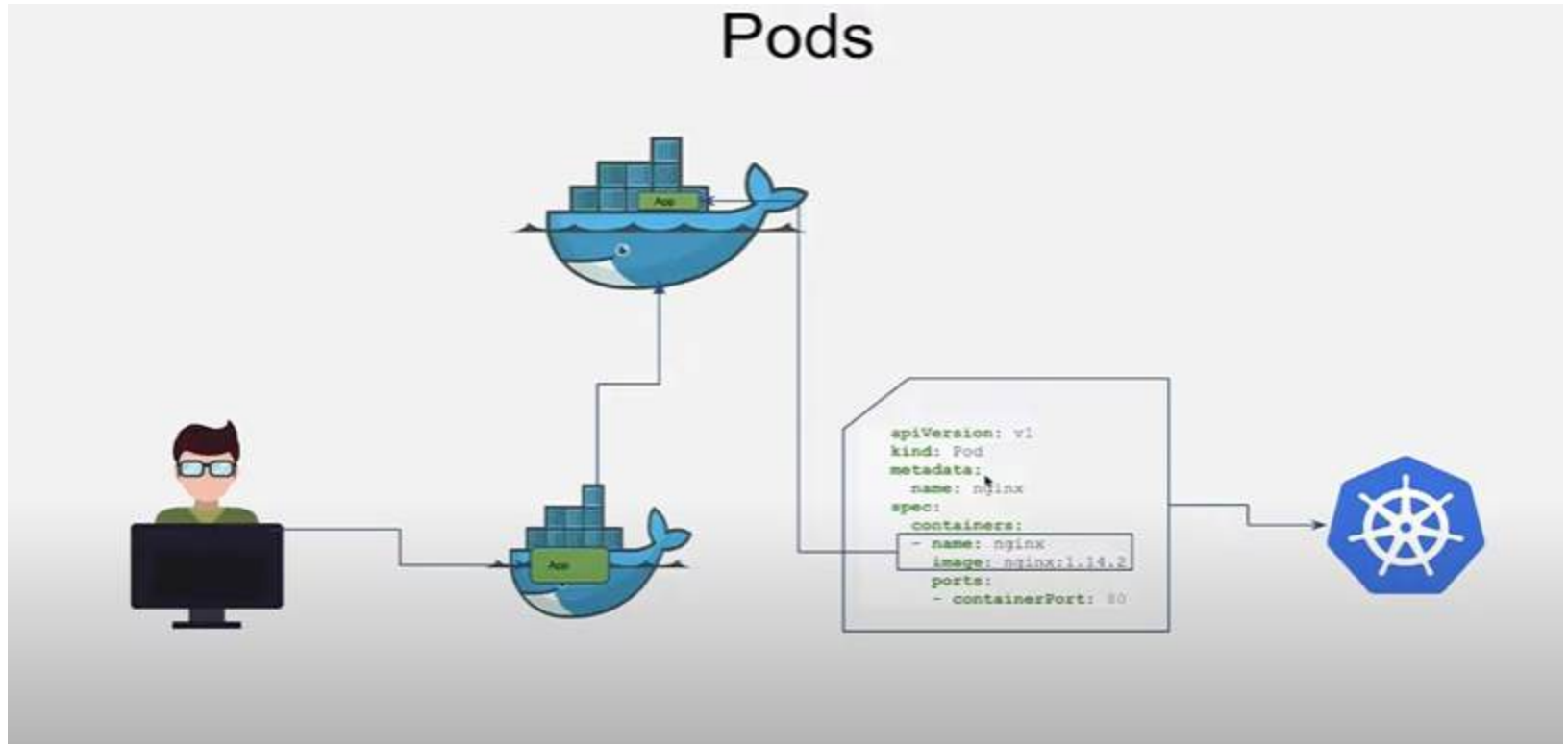



catalog
checkout
db
svc-api
rec-svc
esearch
flask
redis-db
mail-svc
kaffka
rundeck
login-svc

Pods

- Pods are the smallest deployable units in Kubernetes.
- A Pod can be one or more containers.

Pods


App
App
apiVersion: v1
kind: Pod
metadata:
name: nginx
spec:
containers:
- name: nginx
image: nginx:1.14.2
ports:
- containerPort: 80

Pods

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```

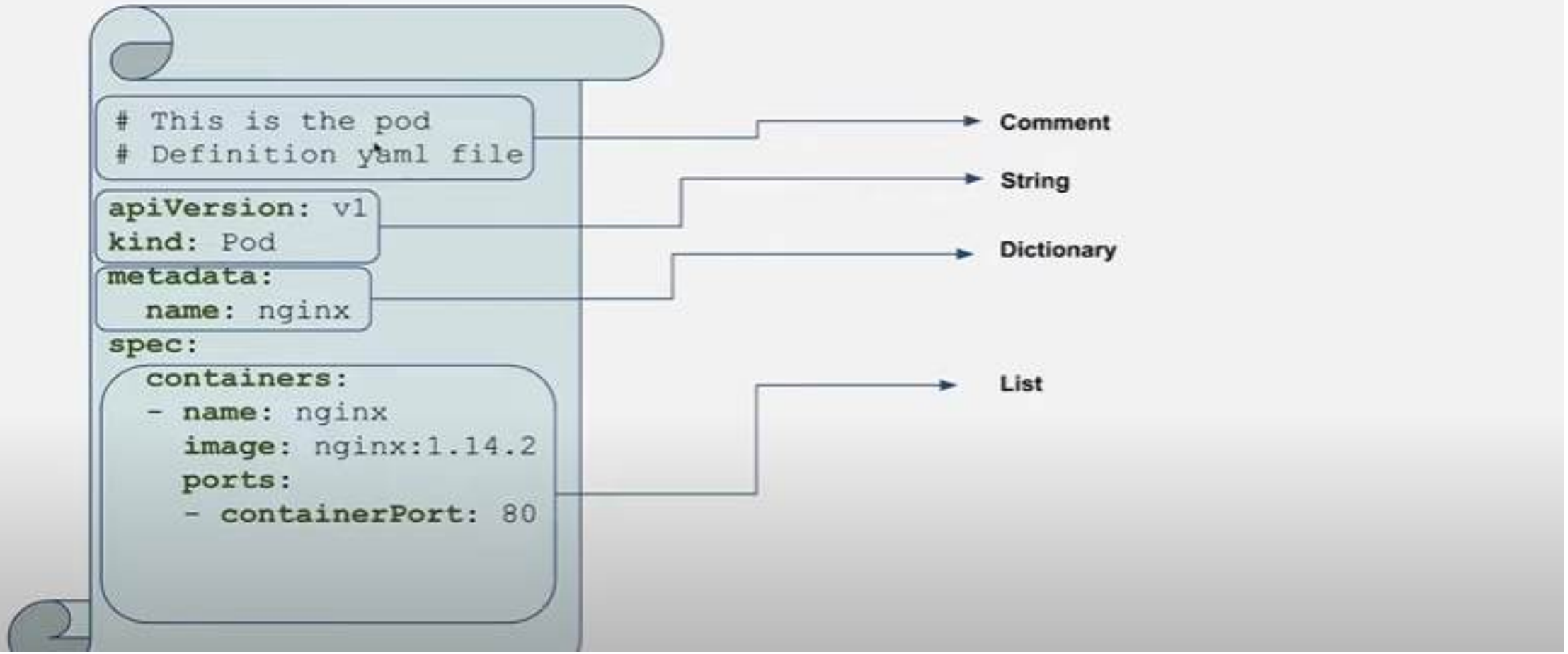
Pods

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```

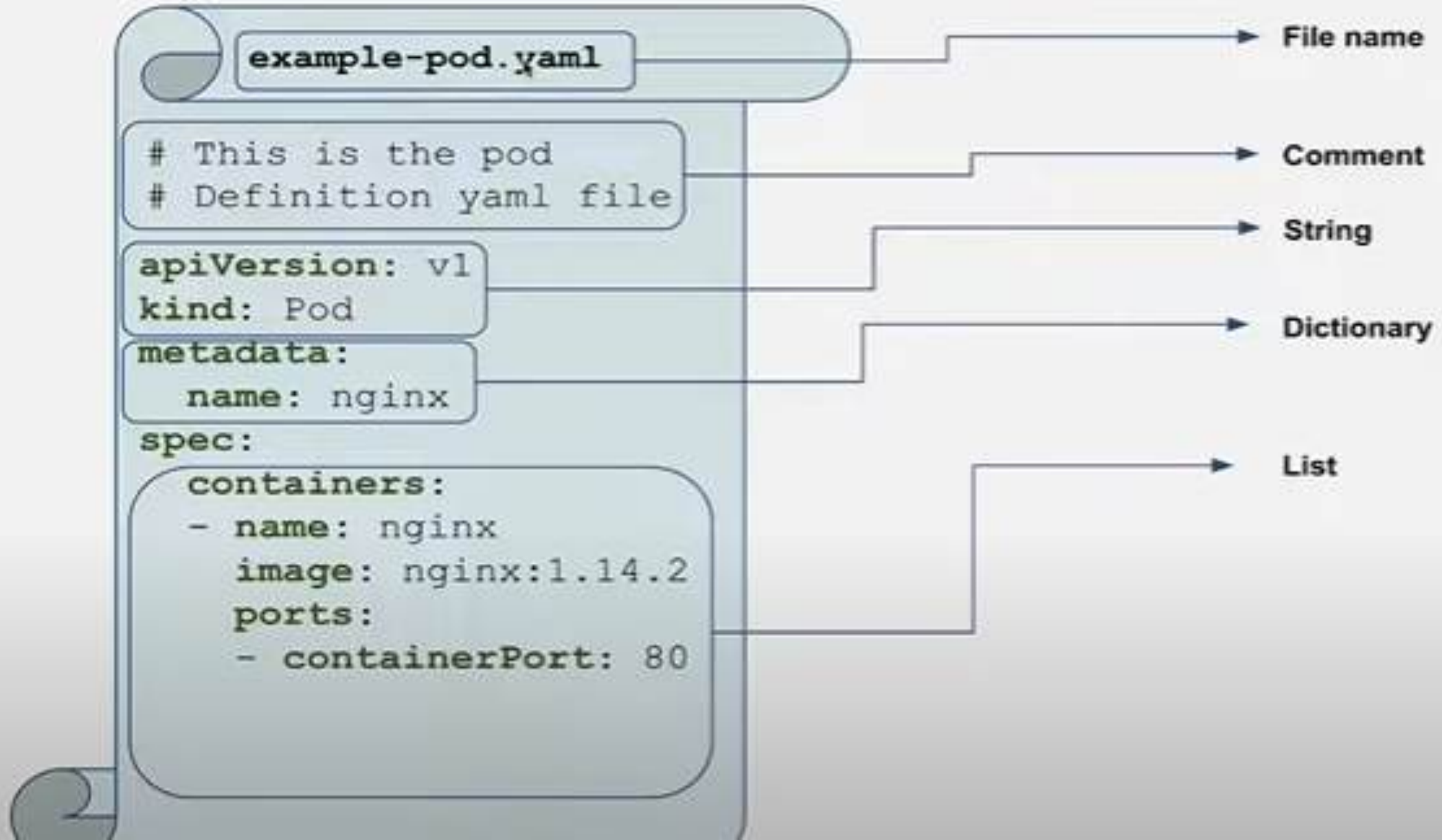
(⎈ |kind-c11:default)-> kubectl create -f pod.yaml

(⎈ |kind-c11:default)-> kubectl get pod

Pods


This is the pod
Definition yaml file
apiVersion: v1
kind: Pod
metadata:
name: nginx
spec:
containers:
- name: nginx
image: nginx:1.14.2
ports:
- containerPort: 80
Comment
String
Dictionary
List

Pods


example-pod.yaml
This is the pod
Definition yaml file
apiVersion: v1
kind: Pod
metadata:
name: nginx
spec:
containers:
- name: nginx
image: nginx:1.14.2
ports:
- containerPort: 80
File name
Comment
String
Dictionary
List

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```

Pods

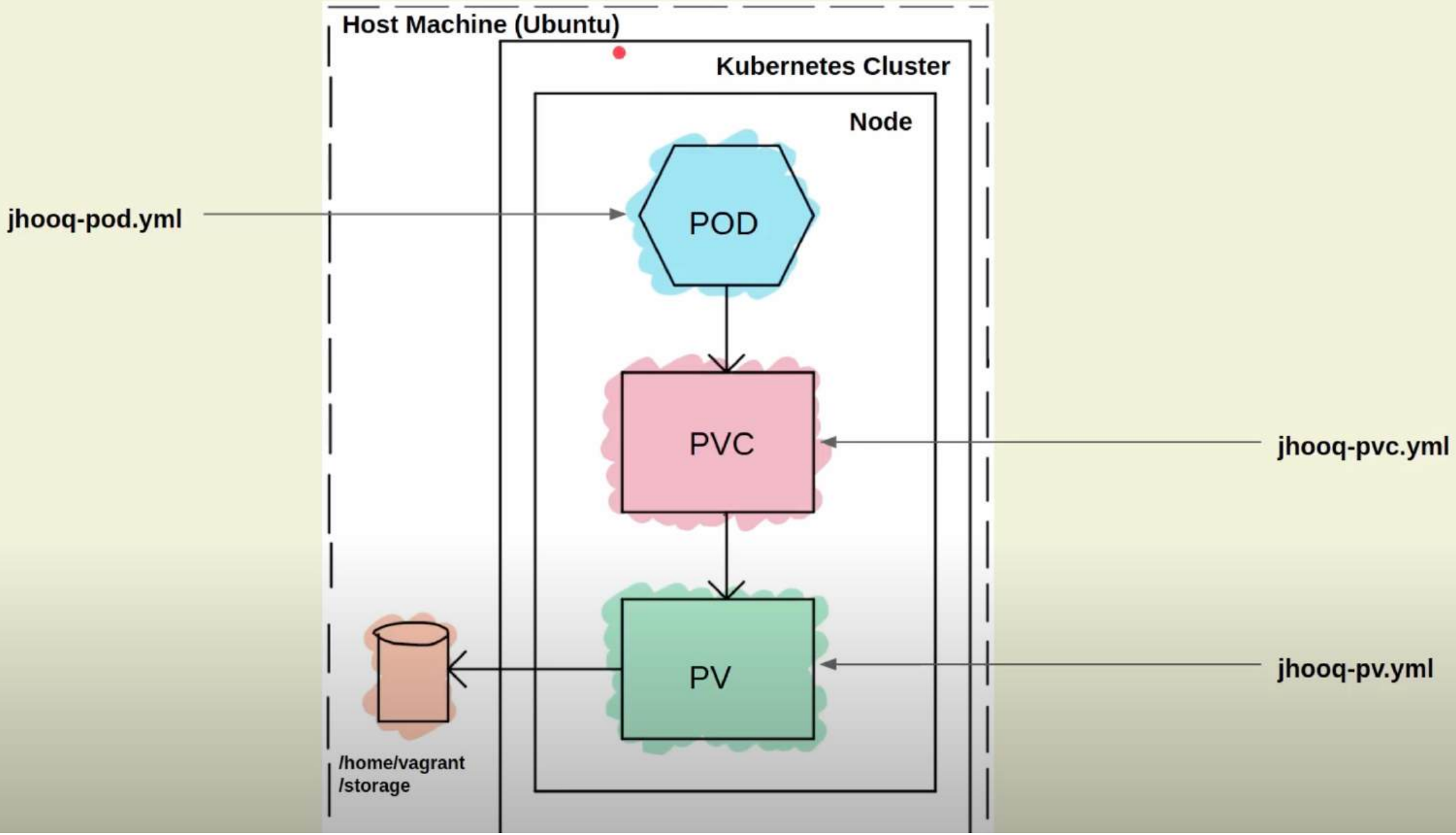
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - image: nginx:1.14.2
    name: nginx
    ports:
    - containerPort: 80
```

Pods

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
  - name: php-fpm
    image: php-fpm:1.12
    resources:
      limits:
        memory: "200Mi"
      requests:
        memory: "100Mi"
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: php-fpm
    image: php-fpm:1.12
    resources:
      limits:
        memory: "200Mi"
      requests:
        memory: "100Mi"
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```


Host Machine (Ubuntu)
Kubernetes Cluster
Node
jhooq-pod.yml
POD
PVC
jhooq-pvc.yml
PV
jhooq-pv.yml
/home/vagrant
/storage

Persistent Volume (PV)

Persistent Volume (PV)

A **Persistent Volume (PV)** is a **piece of storage** in a **cluster** that an **administrator** has **provisioned**. It is a **resource** in the cluster, just as a node is a cluster resource. A persistent volume is a volume **plug-in** that has a **lifecycle independent** of any **individual pod** that **uses the persistent volume.**

Access Modes

There are currently **four access modes** for PVs in Kubernetes:

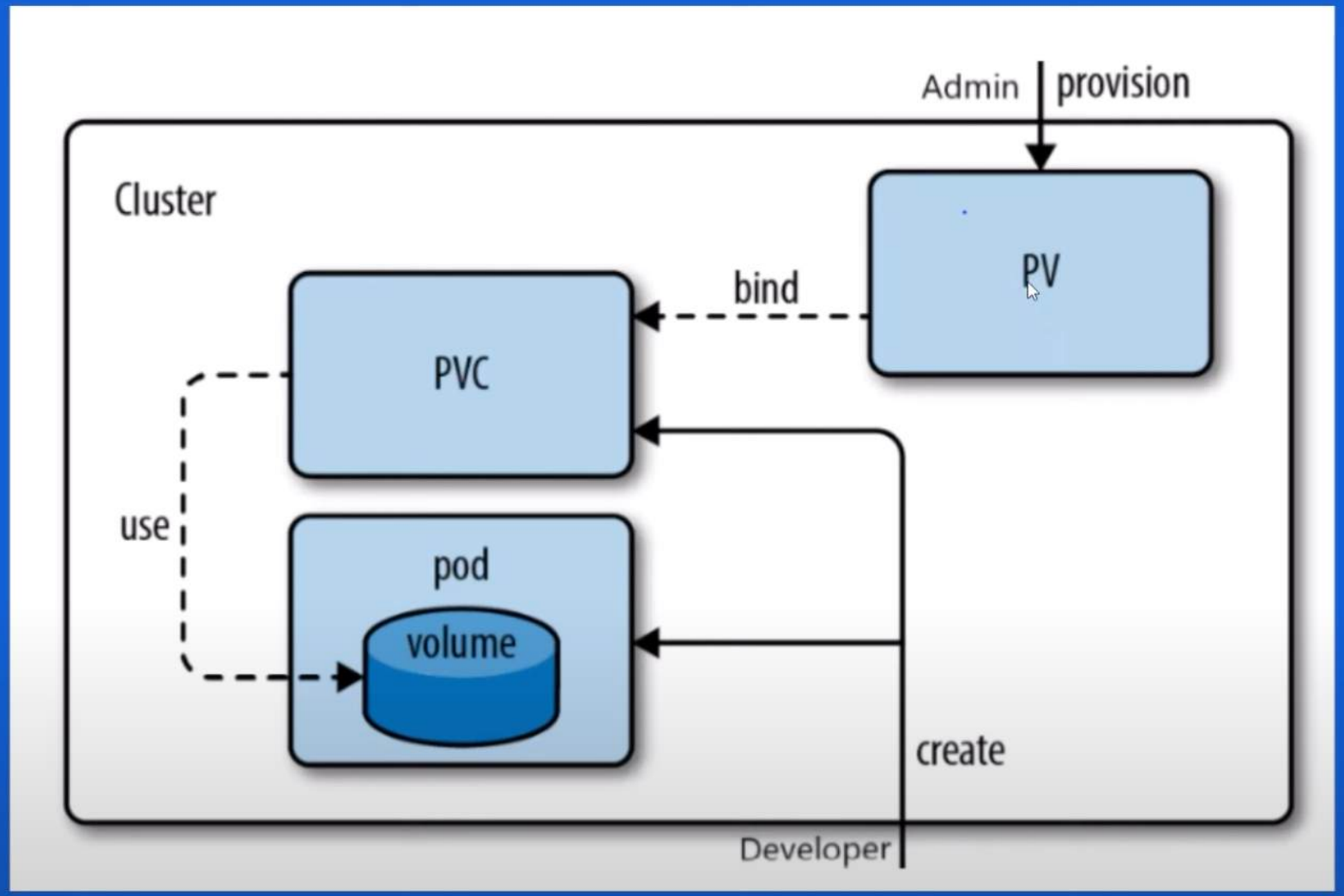
- **ReadWriteOnce**: This allows only a single node to access the volume in read-write mode. Furthermore, all pods in that single node can read and write to such volumes.
- **ReadWriteMany**: Multiple nodes can read and write to the volume.
- **ReadOnlyMany**: This means that the volume will be in a read-only mode and accessible by multiple nodes.
- **ReadWriteOncePod**: Only a single pod can gain access to the volume.

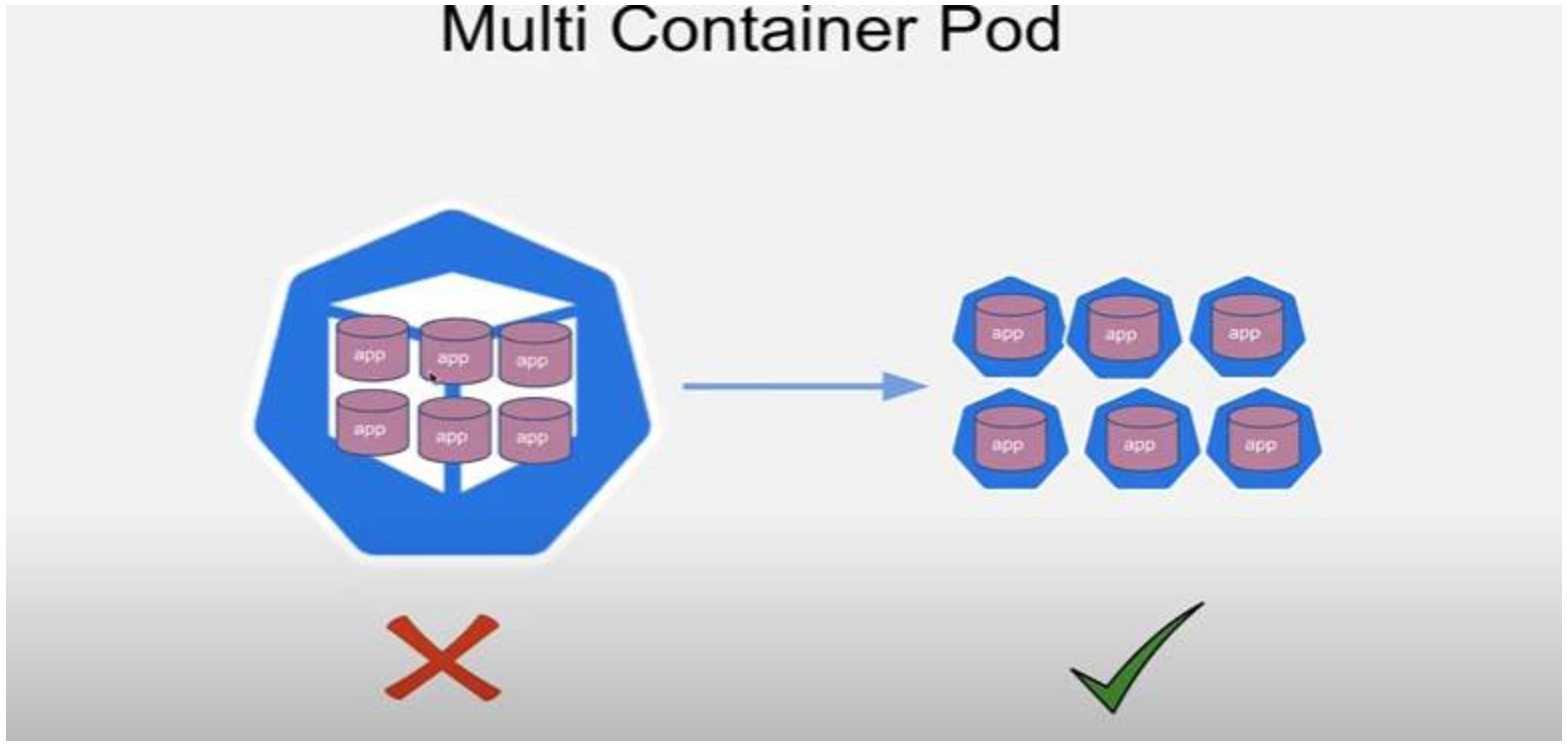
Persistent Volume Claim (PVC)

Persistent Volume Claim

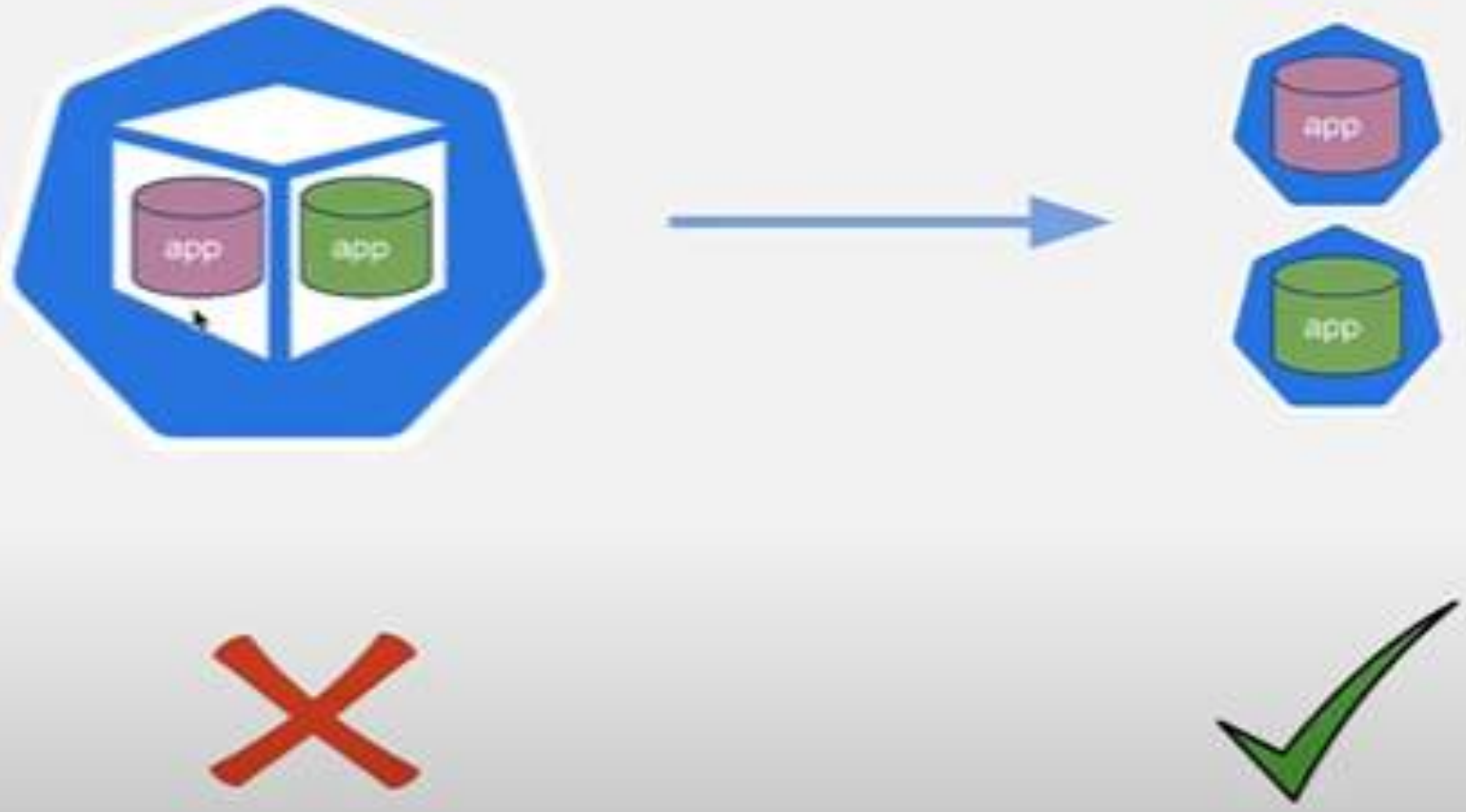
A **Persistent Volume Claim (PVC)** is a **request** for **storage by a user**. It is similar to a Pod. Pods consume node resources and PVCs consume PV resources.

Persistent Volume Claim (PVC)




Multi Container Pod
app
app
app
app
app
app
app
app
app
app
app
app

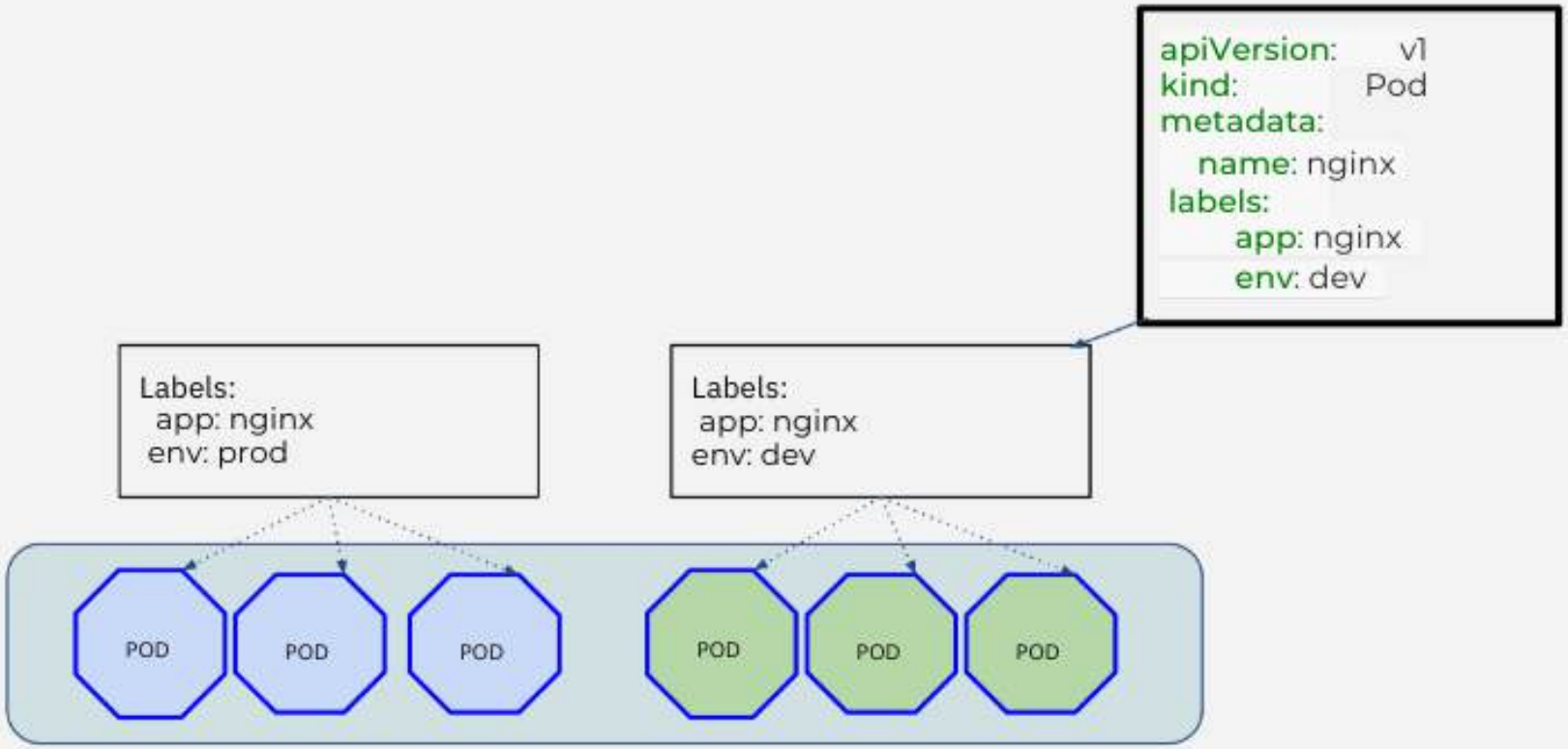
Multi Container Pod



Labels and Selectors

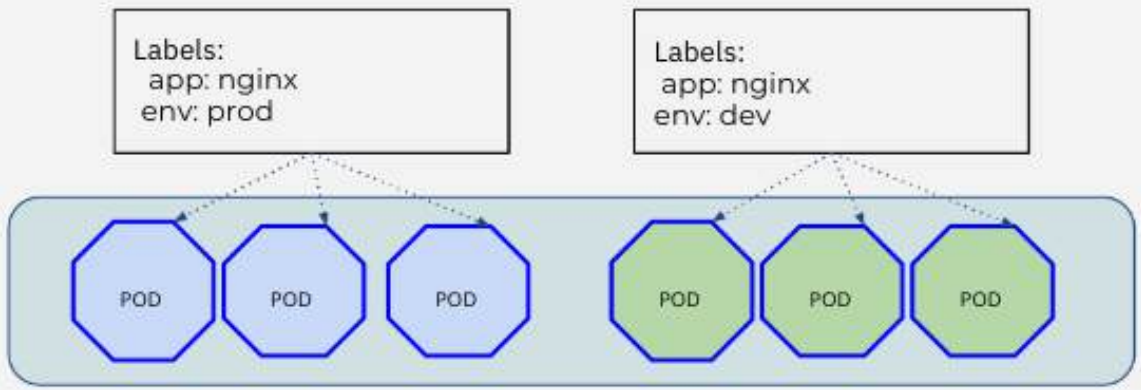
- Labels are key/value pairs used to identify objects
- Labels can be attached to almost any Kubernetes Object
- Labels are mutable, they can be modified at anytime.
- Selectors are a way to select objects based on their labels
- Types of Selectors
  - Equality-based selectors
    - environment = production
    - tier != frontend
  - Set-based selectors
    - environment in (production, qa)
    - tier notin (frontend, backend)

Labels and Selectors



Labels and Selectors

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
spec:
  selector:
    app: nginx
```



Deployments

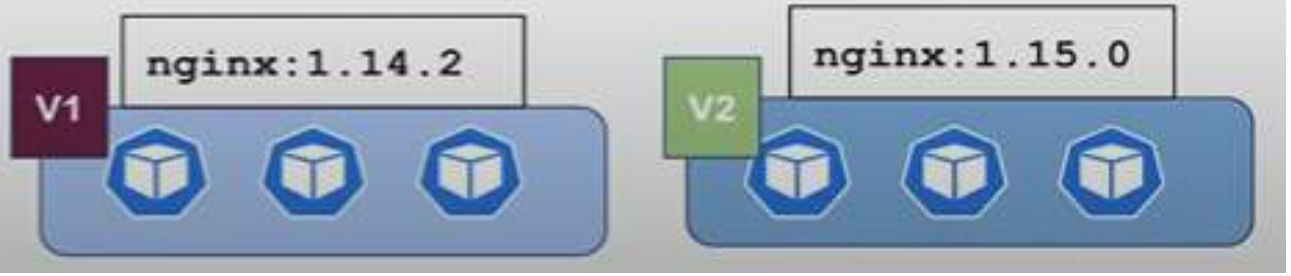
What is Kubernetes Deployment

Managing The Lifecycle of The Application:

- Create the Pods
- Rollout new version of the application
- Rollback to previous version
- Scale the application
- Self healing

Deployments

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.15.0
        ports:
```



Deployments

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.15.0
        ports:
        - containerPort: 80
```


```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.15.0
    ports:
    - containerPort: 80
```

Deployments

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

Create Deployment

Declarative	Imperative
(⎈ |kind-c11:default)-> kubectl create -f deploy.yaml	(⎈ |kind-c11:default)-> kubectl create deployment nginx-deployment --image=nginx:1.14.2 --replicas=3 --port=80

Deployments

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
```

Create Deployment

Declarative	Imperative
(⎈ |kind-c11:default)-> kubectl create -f deploy.yaml	(⎈ |kind-c11:default)-> kubectl create deployment nginx-deployment --image=nginx:1.14.2 --replicas=3 --port=80

Deployments

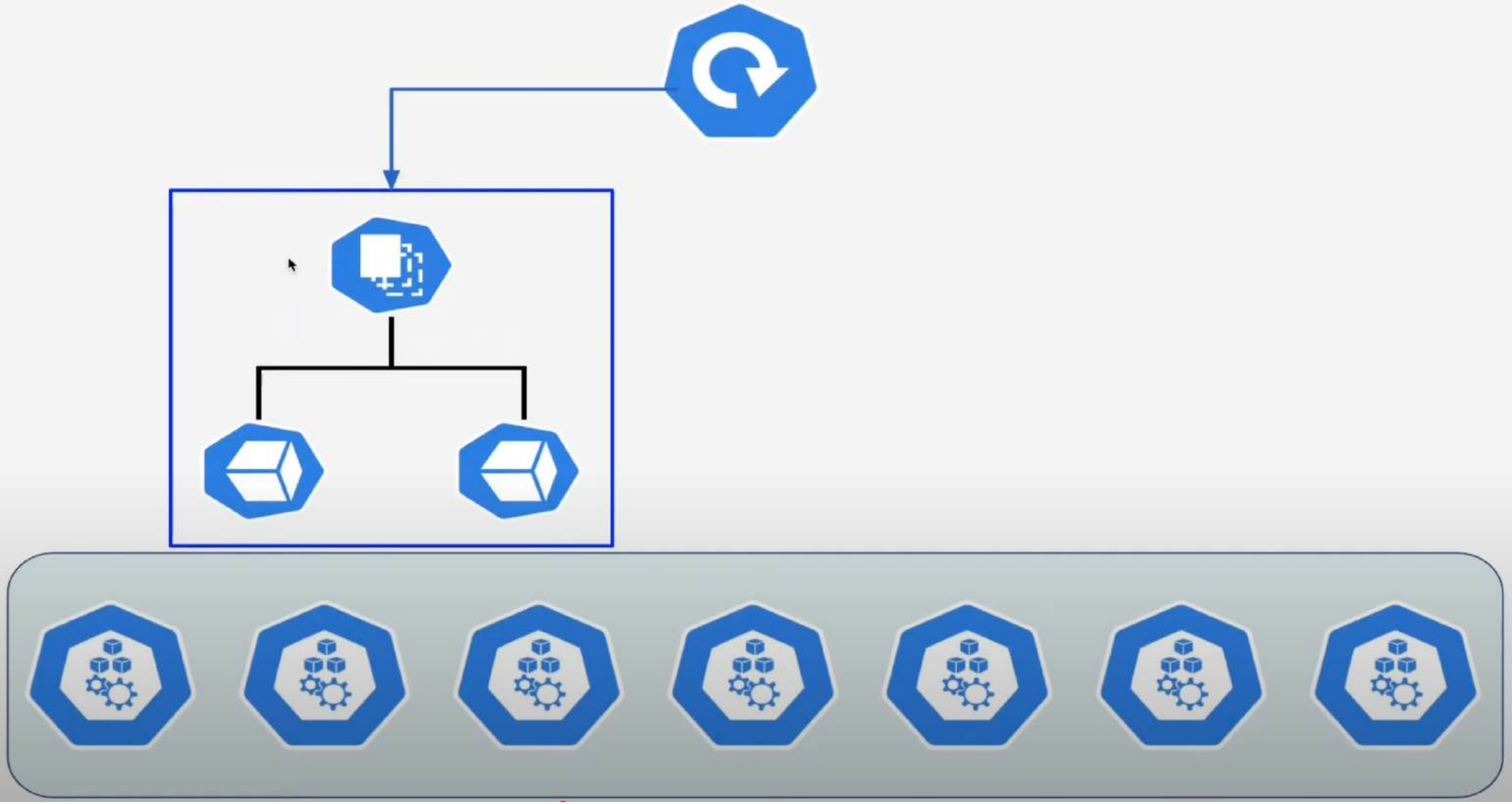
Demo

- ❑ Deployment YAML File
- ❑ Create Deployment
  - ❑ Declarative
  - ❑ Imperative
- ❑ Update Deployment
- ❑ Rollback Deployment
- ❑ Scale Deployment
- ❑ Revision History Limit

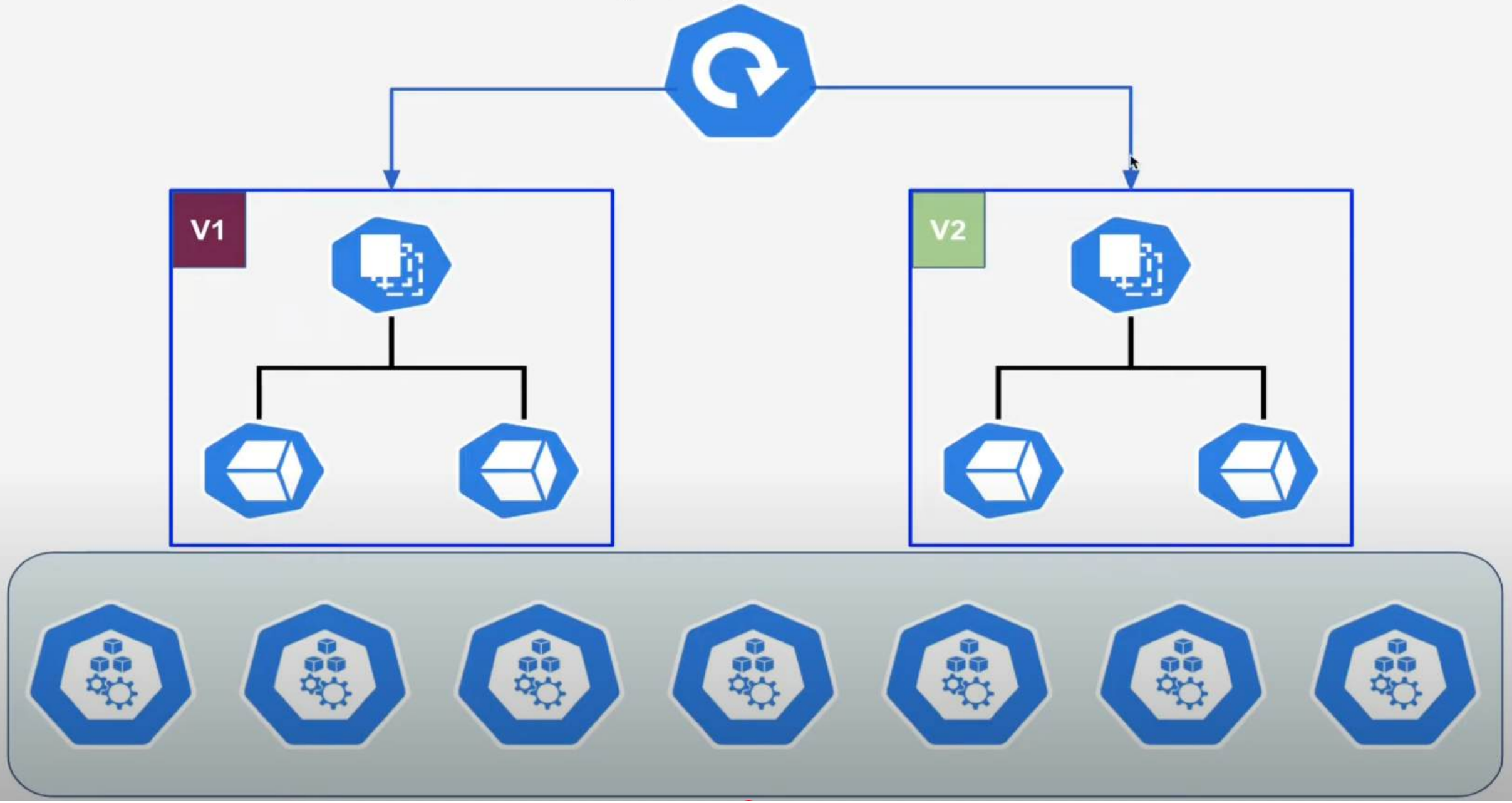
Reference:
https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

Deployments

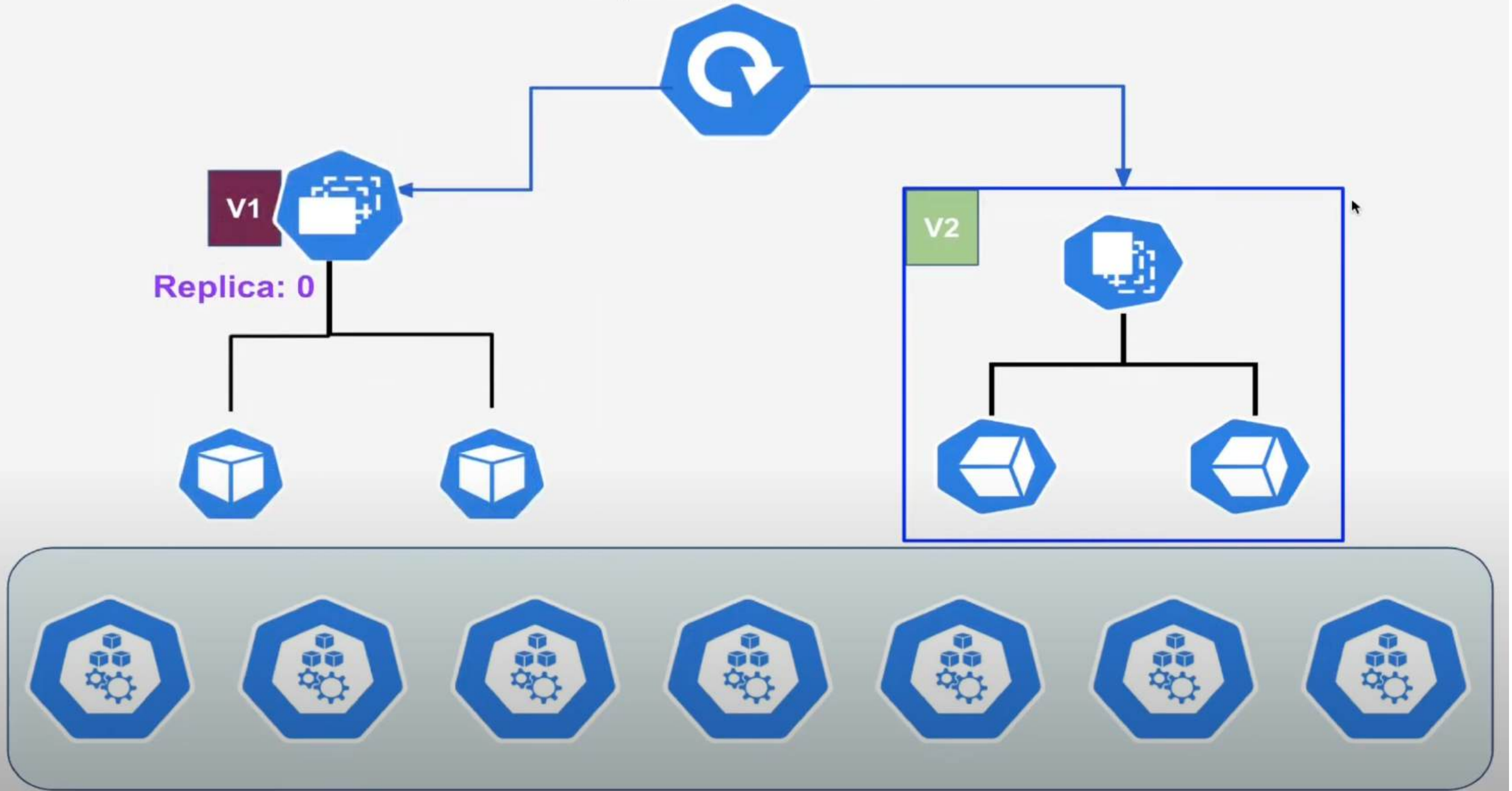
Deployments



Deployments



Deployments



Deployments

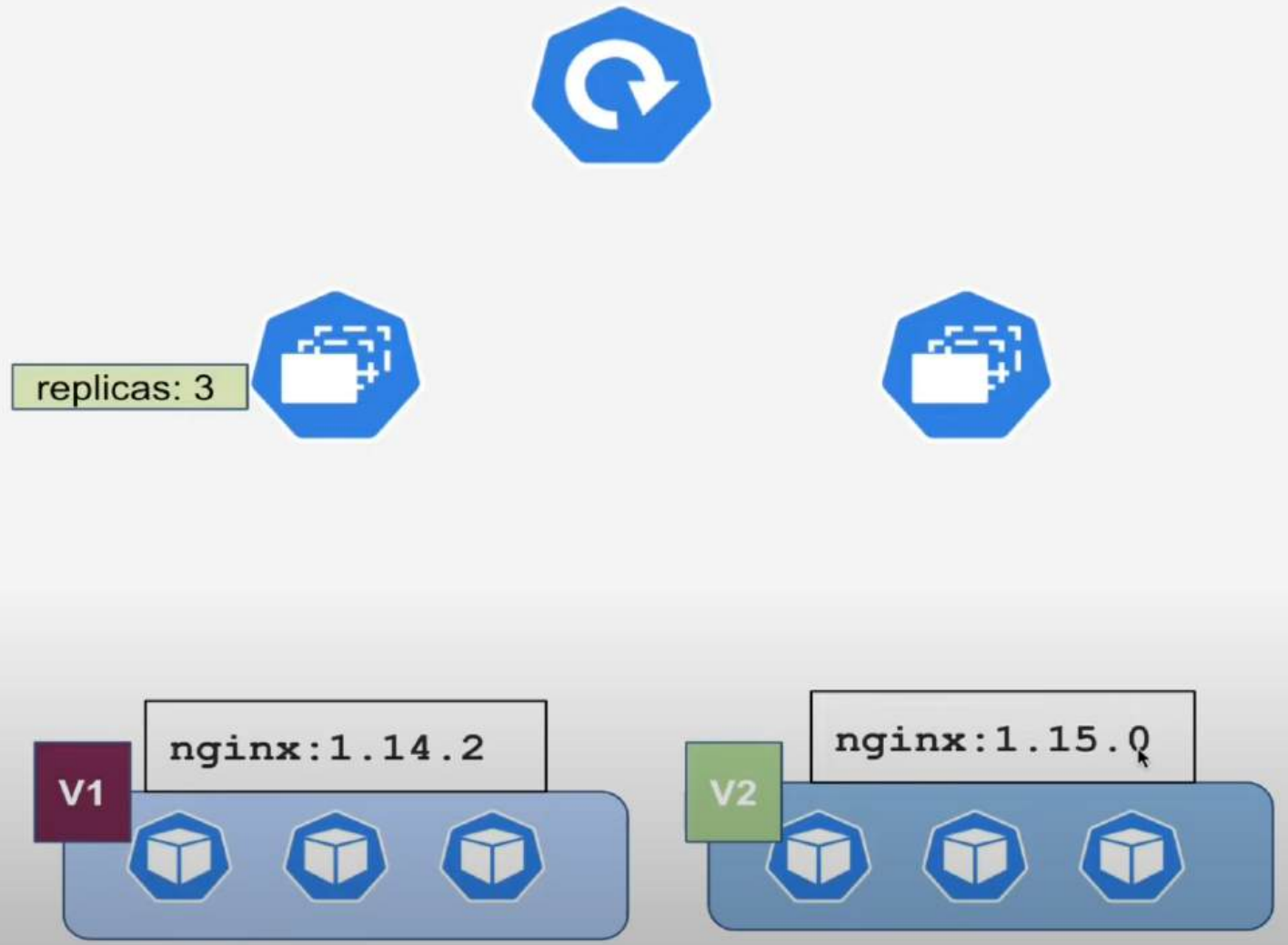
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

Deployments

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

Deployments

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.15.0
        ports:
        - containerPort: 80
```



Deployments

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

Deployment Strategies

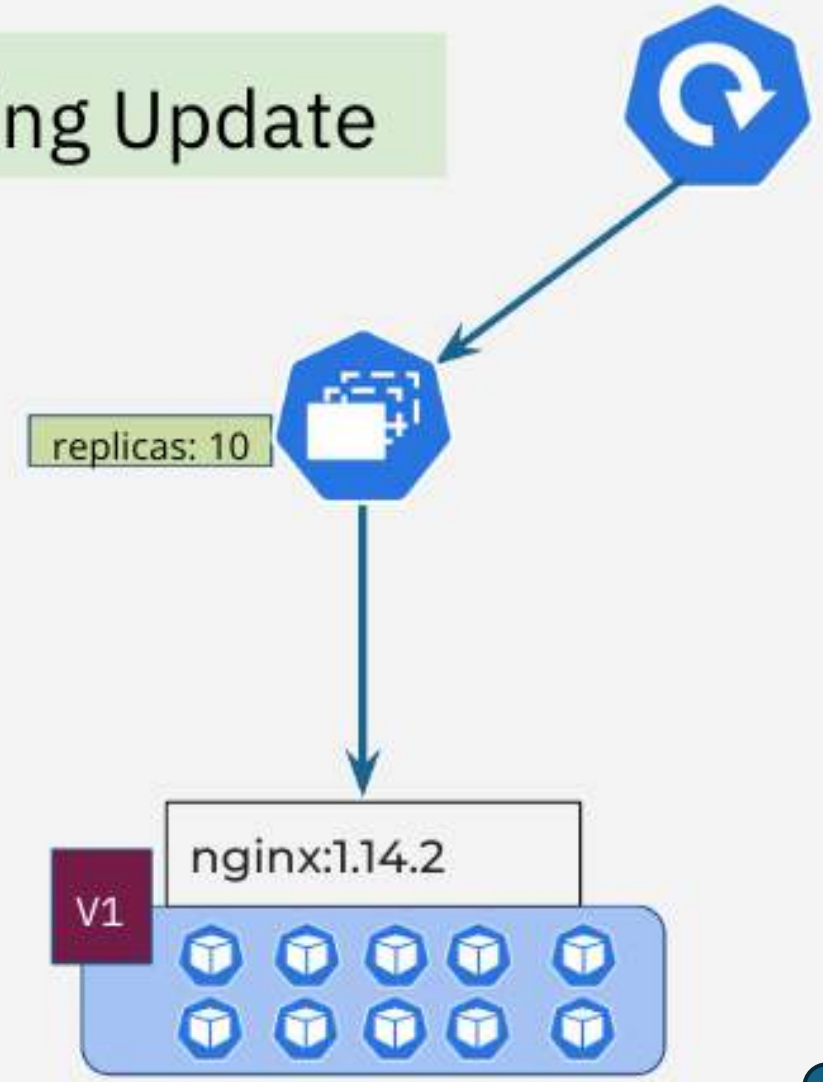
Rolling Update

```
apiVersion: apps/v1
kind: Deployment
metadata:
name: nginx-deployment
labels:
      app: nginx
spec:
replicas: 10
selector:
       matchLabels:
           app: nginx
strategy:
      rollingUpdate:
         maxSurge: 25%
         maxUnavailable: 25%
      type: RollingUpdate
template:
       metadata:
            labels:
            app: nginx
      spec:
         containers:
         - name: nginx
            image: nginx:1.14.2
            ports:
```

Deployment Strategies

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
name: nginx-deployment
labels:
      app: nginx
spec:
replicas: 10
selector:
       matchLabels:
           app: nginx
strategy:
      rollingUpdate:
        maxSurge: 30%
        maxUnavailable: 30%
      type: RollingUpdate
template:
       metadata:
            labels:
            app: nginx
      spec:
        containers:
        - name: nginx
            image: nginx:1.14.2
            ports:
            - containerPort: 80
```

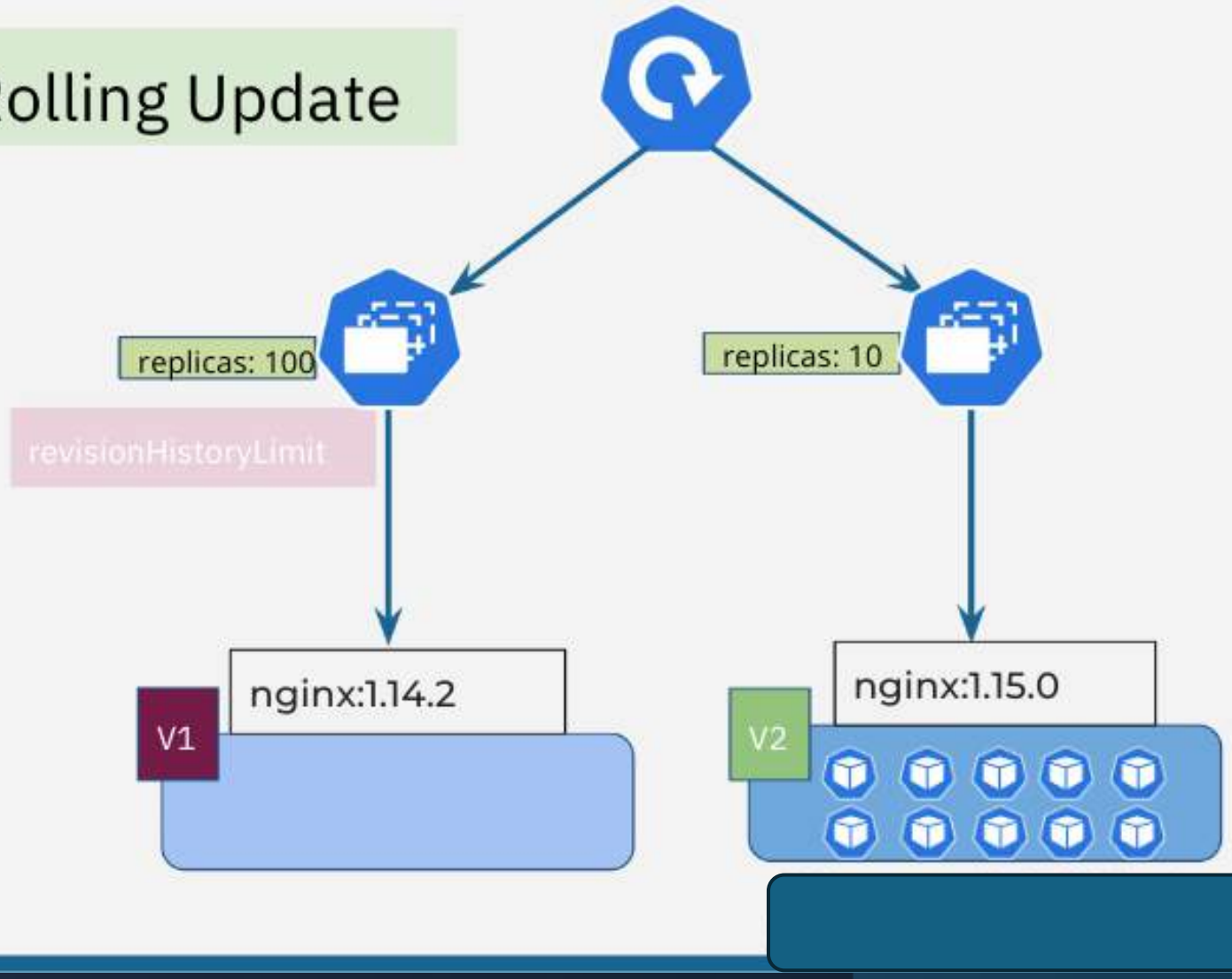
Rolling Update



Deployment Strategies

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
name: nginx-deployment
labels:
    app: nginx
spec:
replicas: 10
selector:
    matchLabels:
        app: nginx
strategy:
    rollingUpdate:
      maxSurge: 30%
      maxUnavailable: 30%
    type: RollingUpdate
template:
    metadata:
        labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.15.0
        ports:
        - containerPort: 80
```

Rolling Update



Services

01

Service Discovery

A way to connect different services or components with a single stable endpoint

02

Load Balancing

Kubernetes automatically distributes traffic between different application backends to optimize performance and availability.

Services

01

Service Discovery

A way to connect different services or components with a single stable endpoint

02

Load Balancing

Kubernetes automatically distributes traffic between different application backends to optimize performance and availability.

172.21.35.41

Services

01

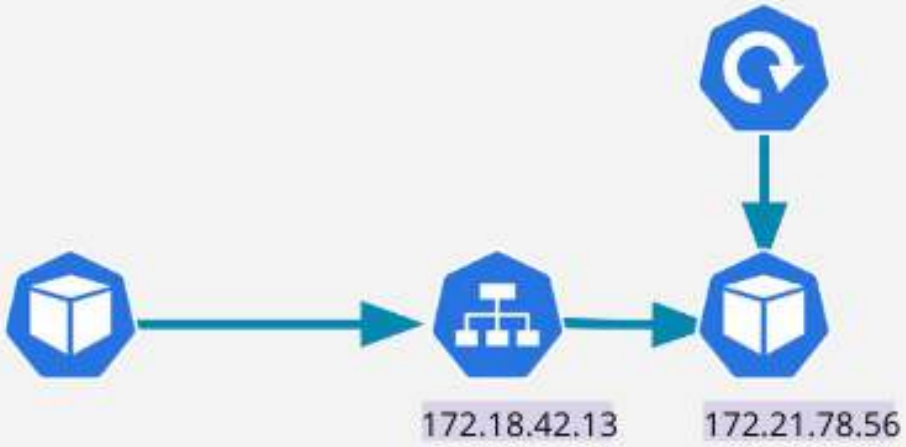
Service Discovery

A way to connect different services or components with a single stable endpoint

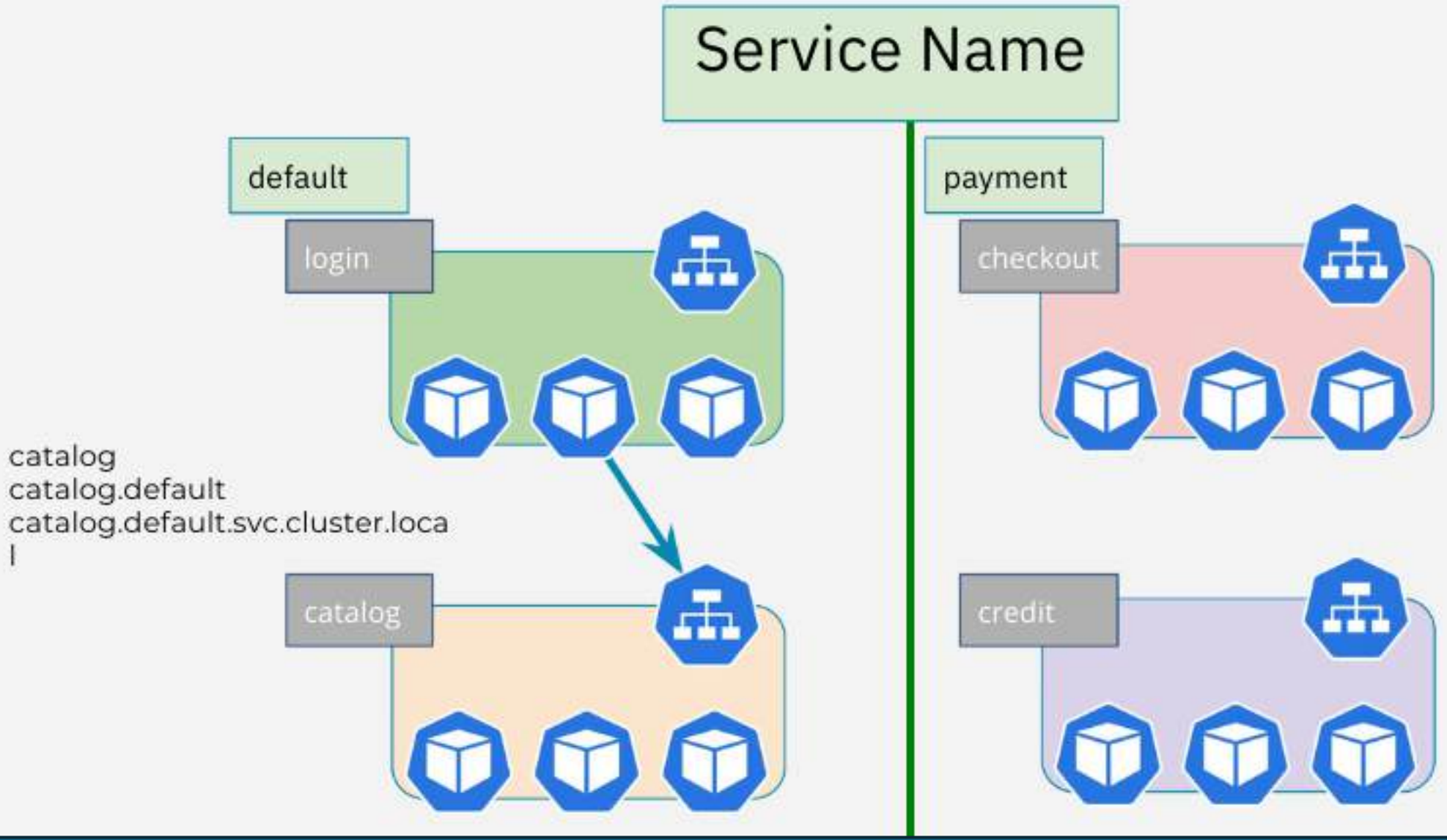
02

Load Balancing

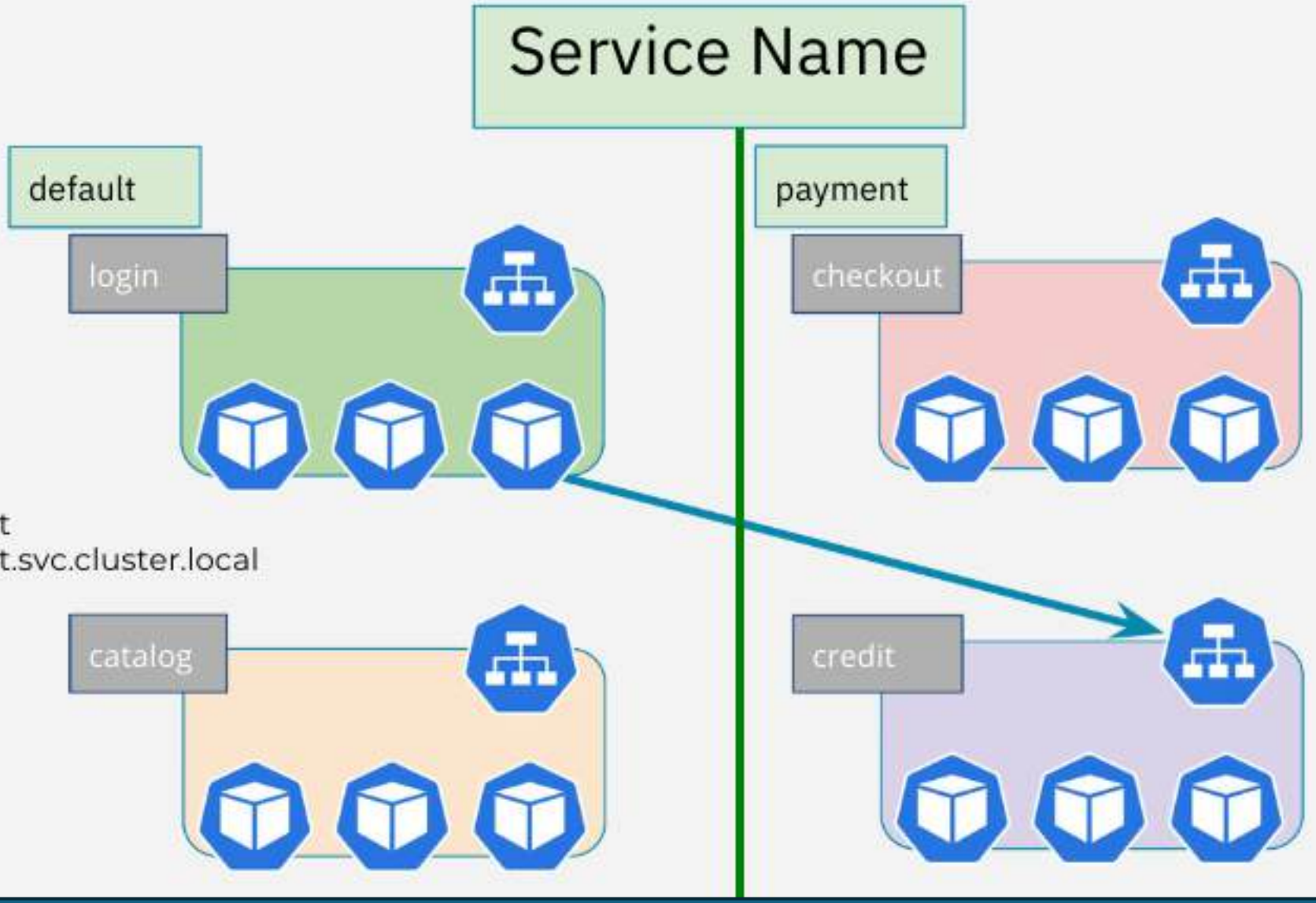
Kubernetes automatically distributes traffic between different application backends to optimize performance and availability.



Services


Service Name
default
login
payment
checkout
catalog
catalog.default
catalog.default.svc.cluster.loca
l
catalog
credit

Services


Service Name
default
payment
login
checkout
credit X
credit.payment
credit.payment.svc.cluster.local
catalog
credit

Services

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 80
```

Services

```
apiVersion: v1
kind: Service
metadata:
    name: my-service
Spec:
    type: ClusterIP
    selector:
        app: nginx
    ports:
        - protocol: TCP
            port: 80
            targetPort: 9376
```

Services

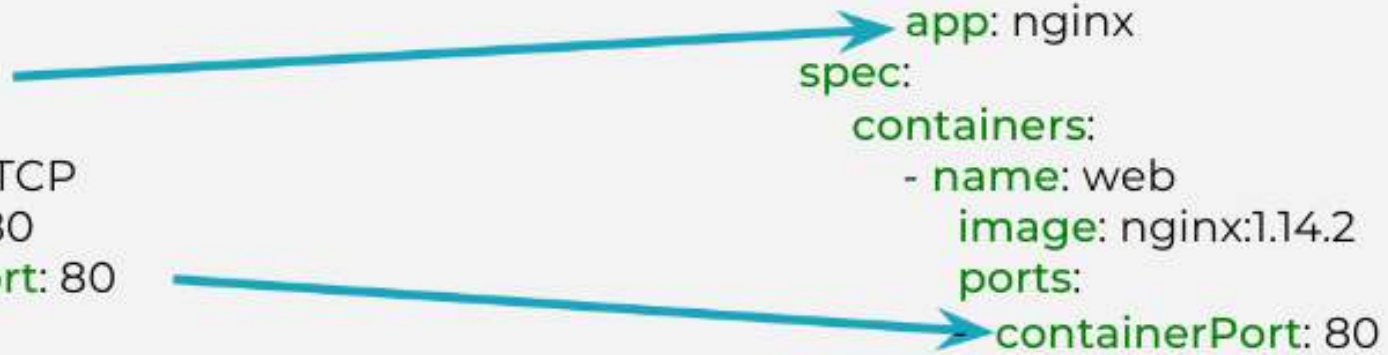
```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 80
```

```yaml
apiVersion:    v1
kind:          Pod
metadata:
  name: test-pod
  labels:
    app: nginx
spec:
  containers:
    - name: web
      image: nginx:1.14.2
      ports:
      -  containerPort: 80
```

Services

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 80
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: test-pod
  labels:
    app: nginx
spec:
  containers:
    - name: web
      image: nginx:1.14.2
      ports:
        - containerPort: 80
```



Services

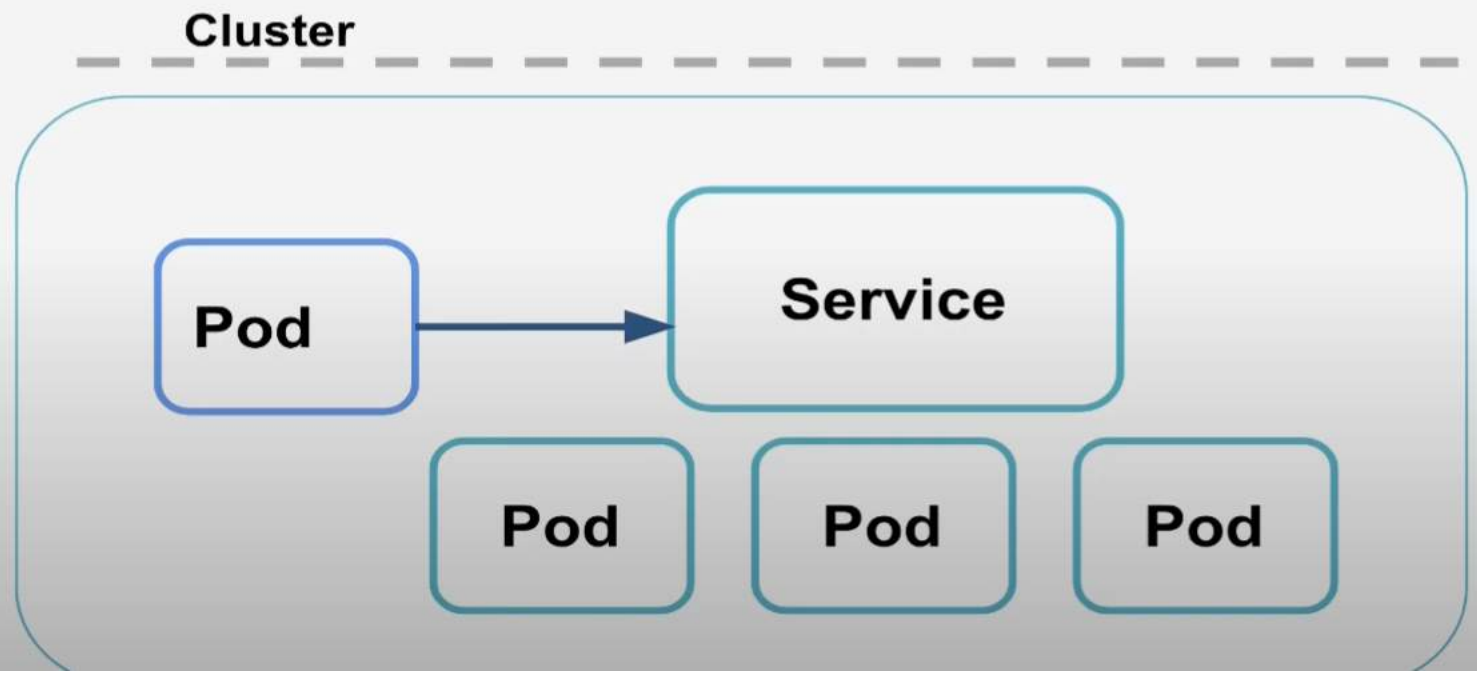
```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
Spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      name: http
      port: 80
      targetPort: 9376
    - name: https
      protocol: TCP
      port: 443
      targetPort: 9377
```

Service Types

- ClusterIP
- NodePort
- LoadBalancer

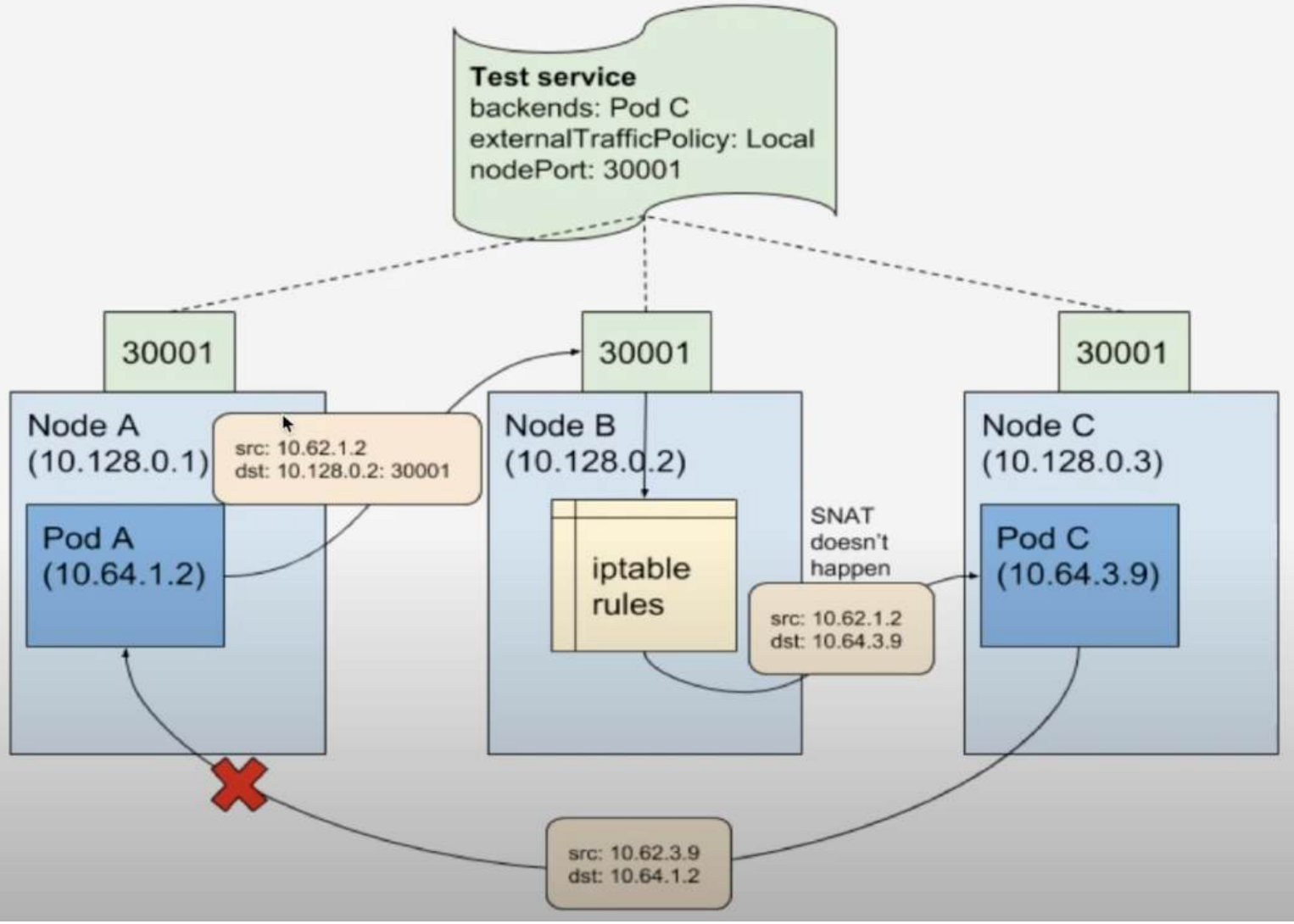
Services

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
Spec:
  type: ClusterIP
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
```



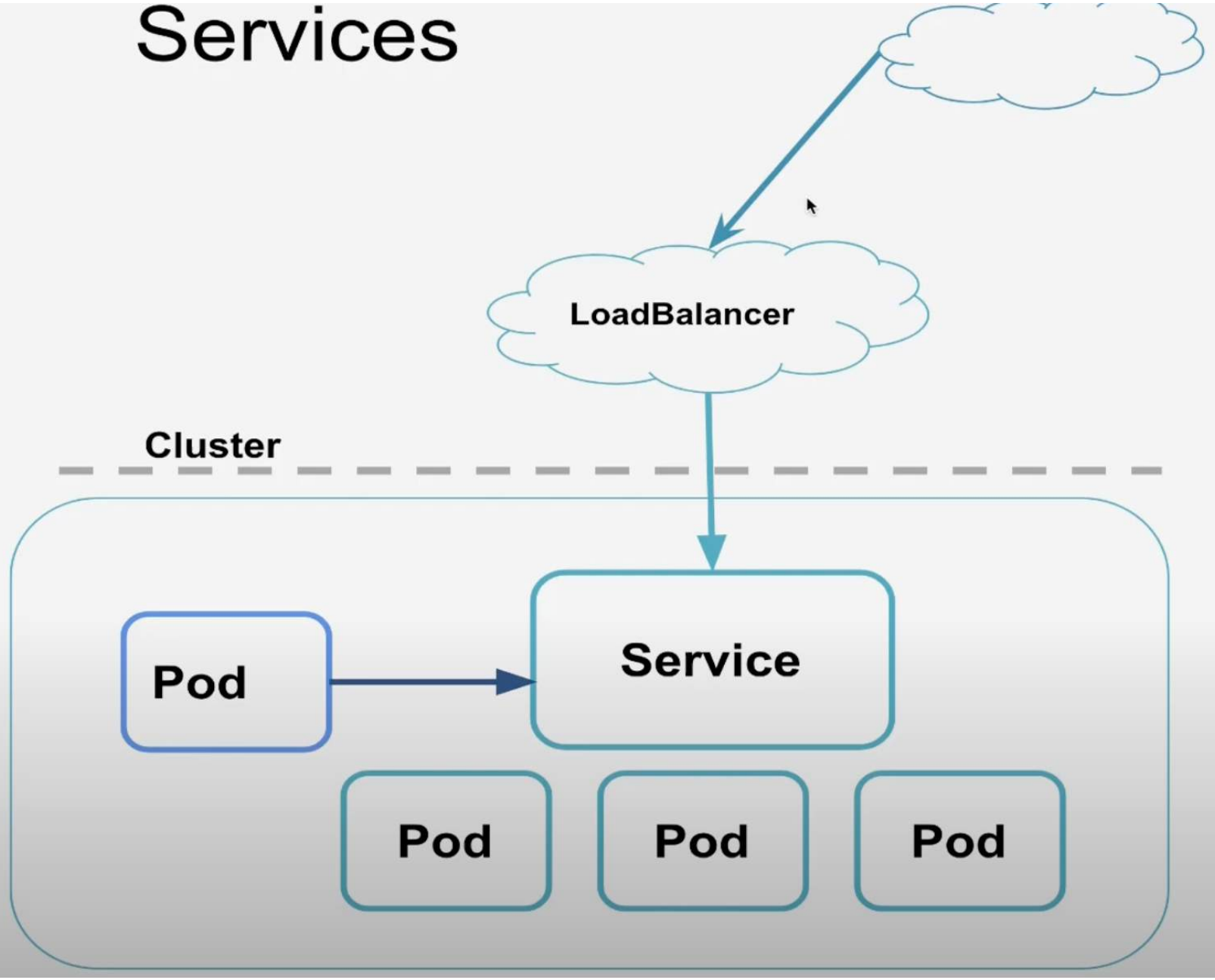
Services

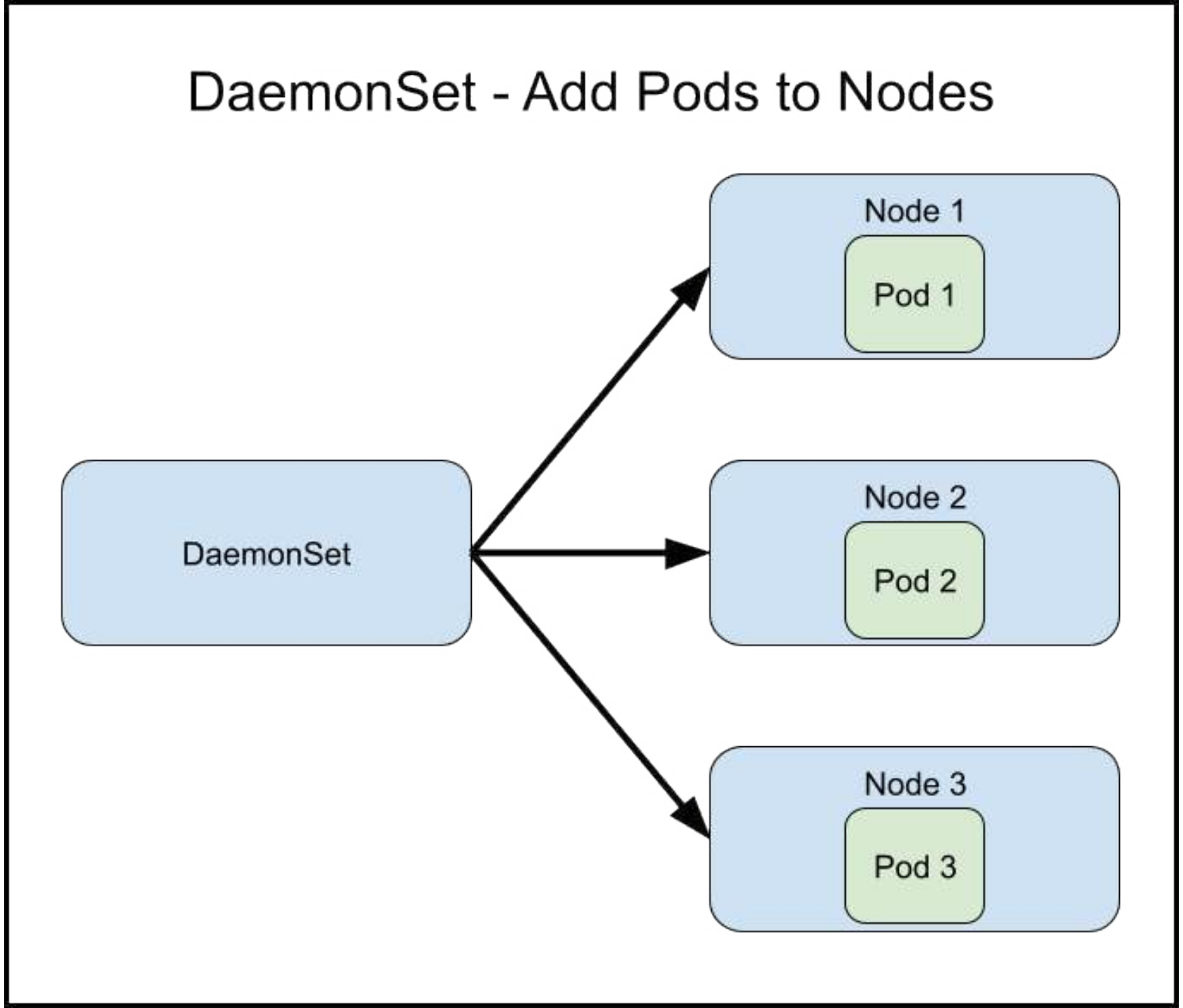
```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
Spec:
  type: NodePort
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
      nodePort: 30305
```

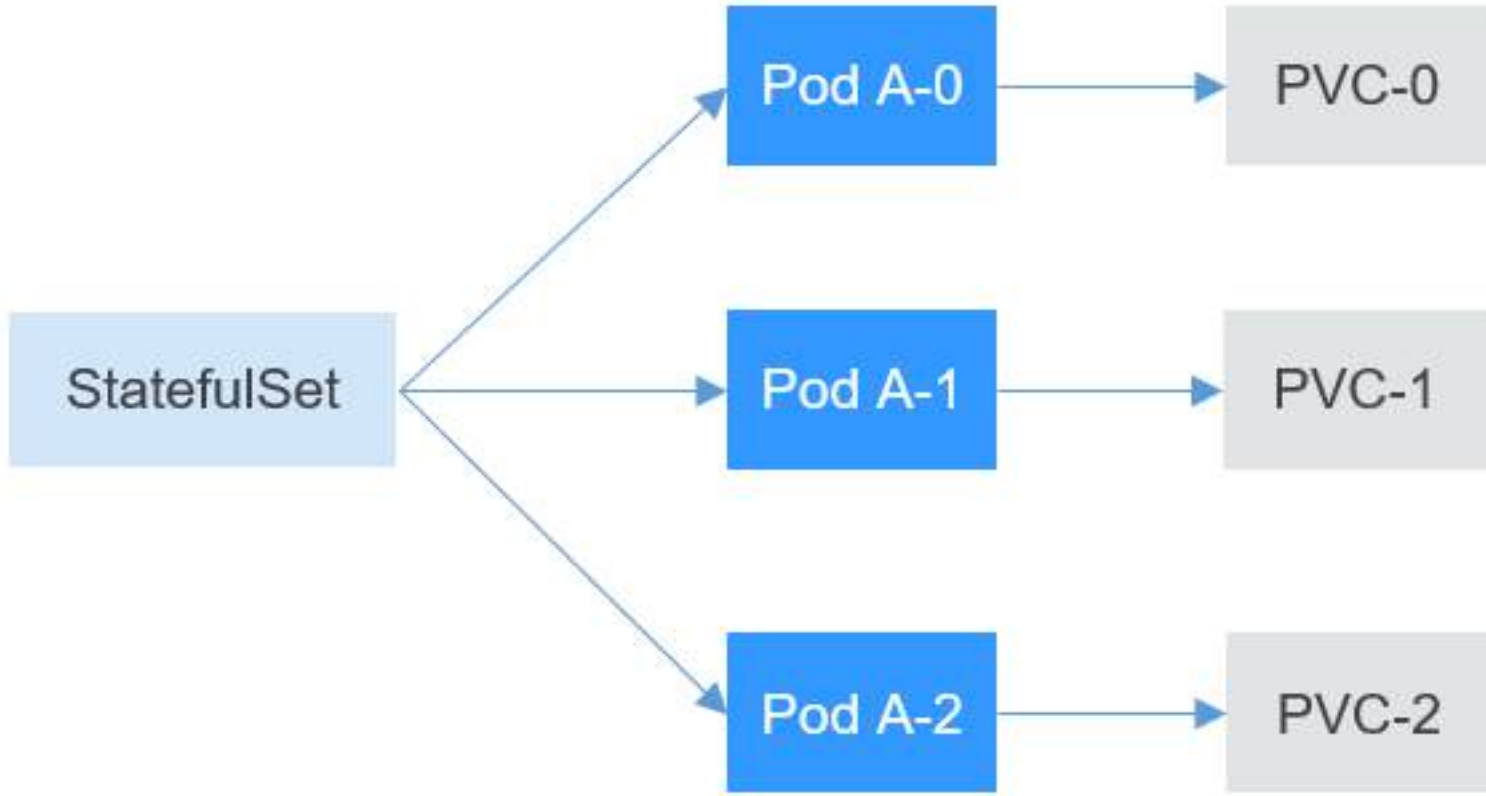


Services

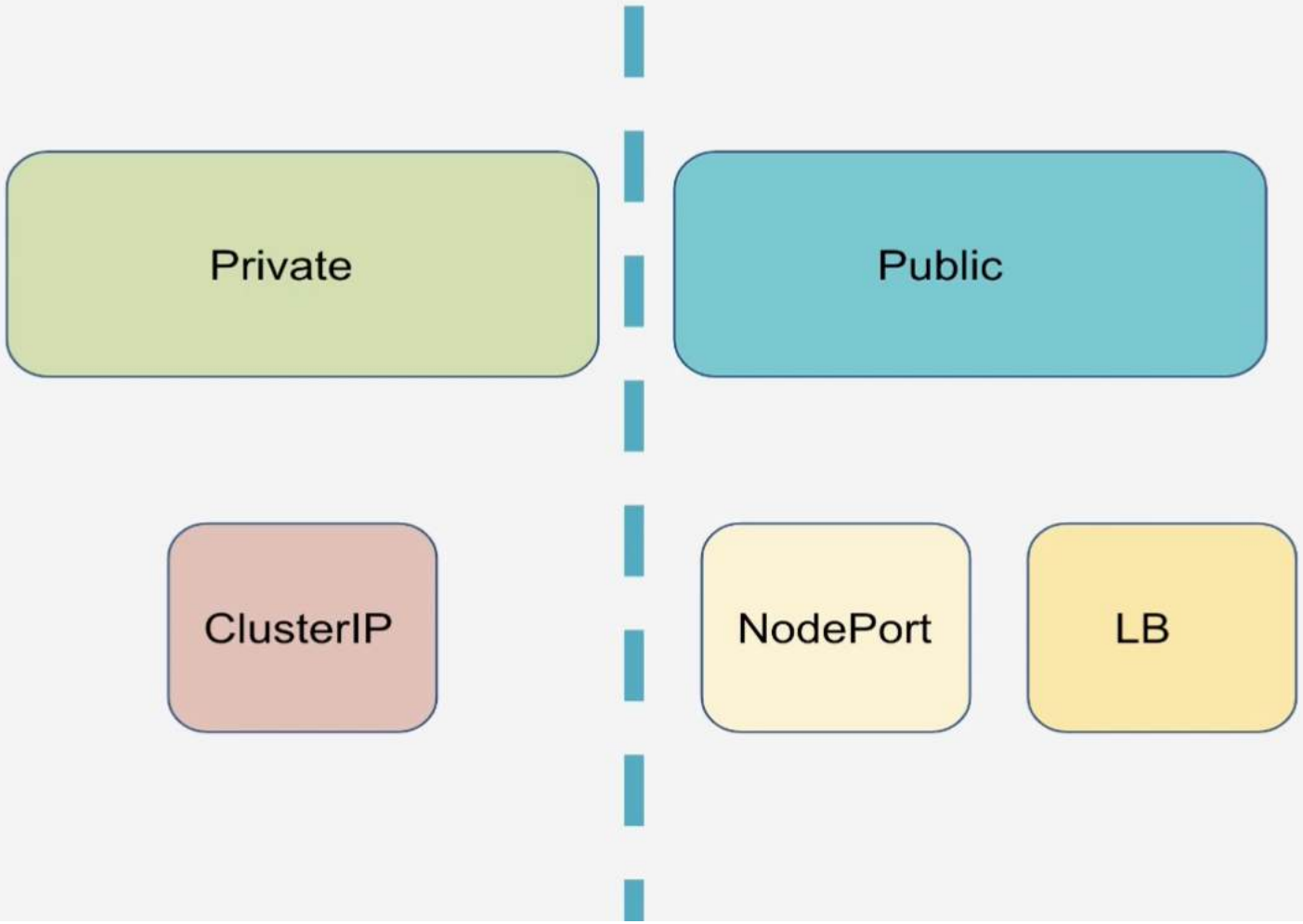
```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
Spec:
  type: LoadBalancer
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```



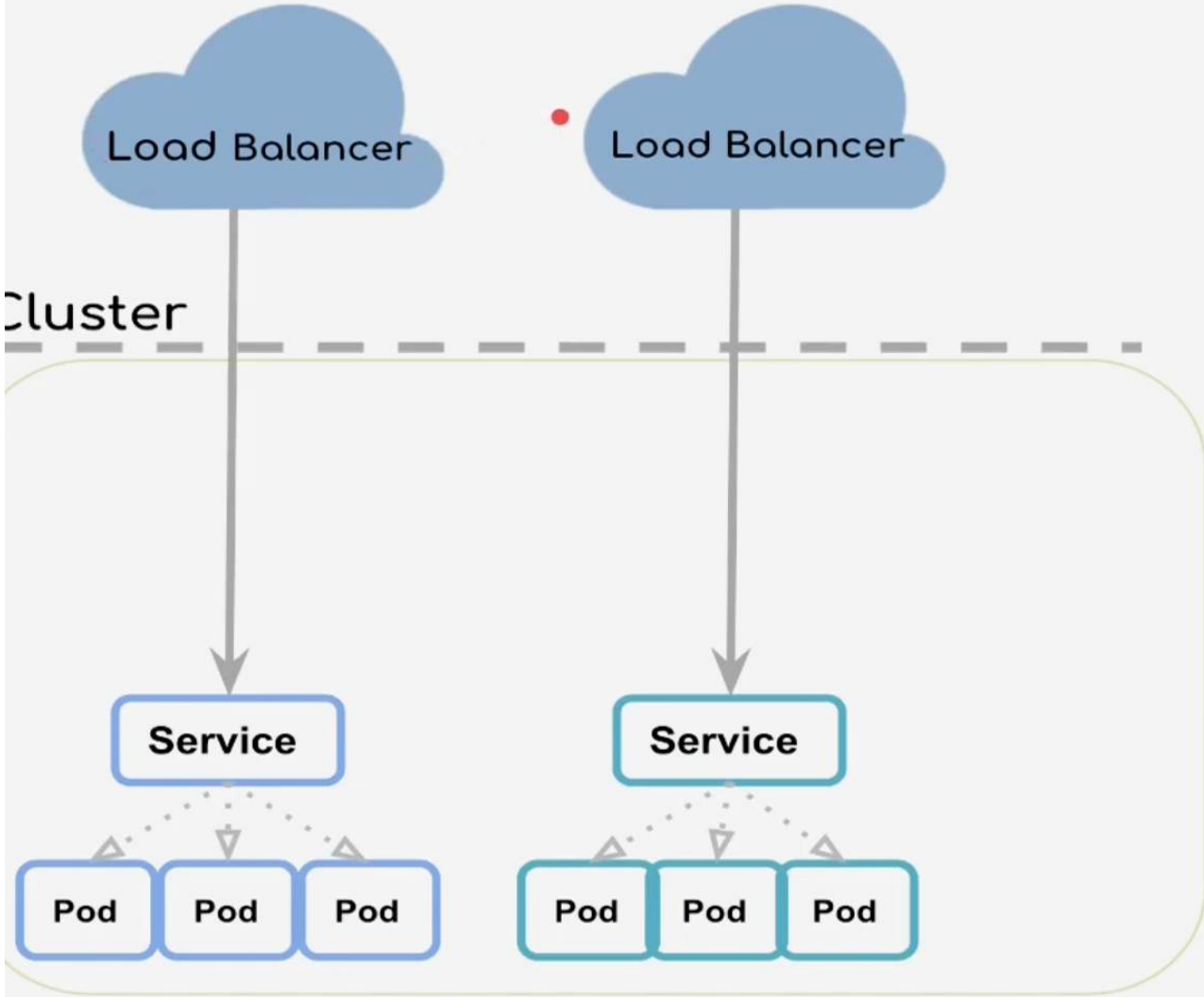

DaemonSet - Add Pods to Nodes
DaemonSet
Node 1
Pod 1
Node 2
Pod 2
Node 3
Pod 3


StatefulSet
Pod A-0
Pod A-1
Pod A-2
PVC-0
PVC-1
PVC-2

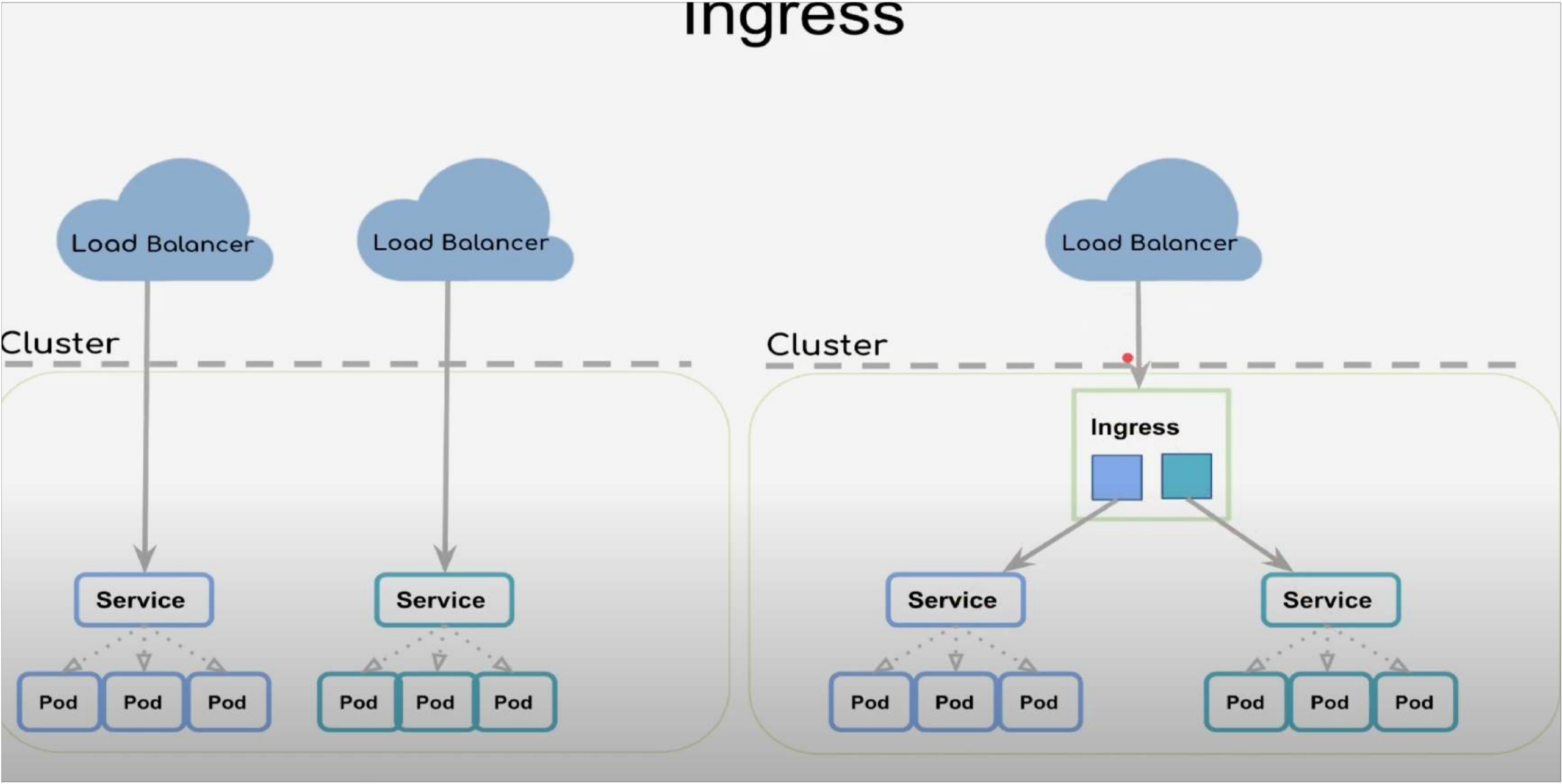
Service Types

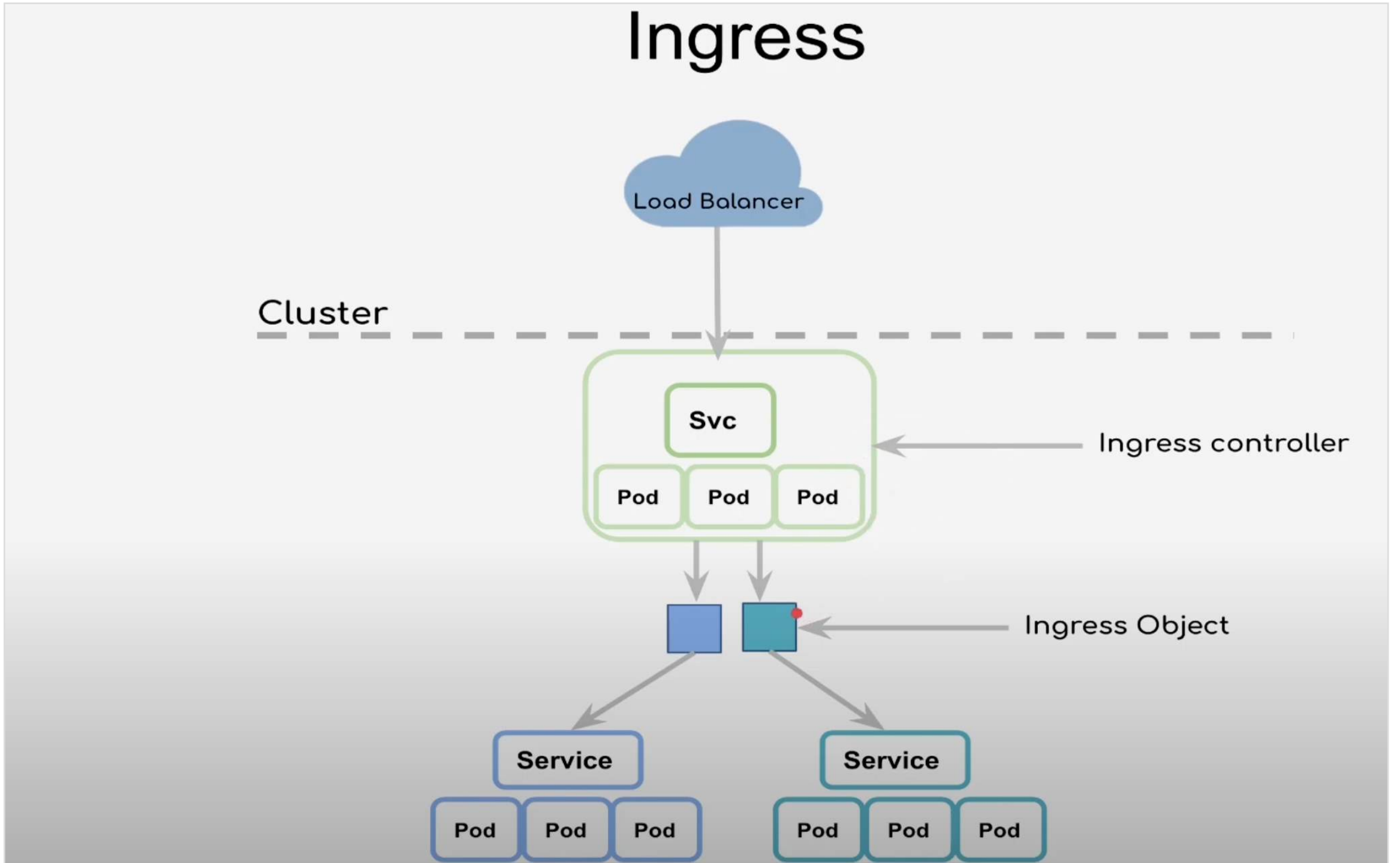


Ingress

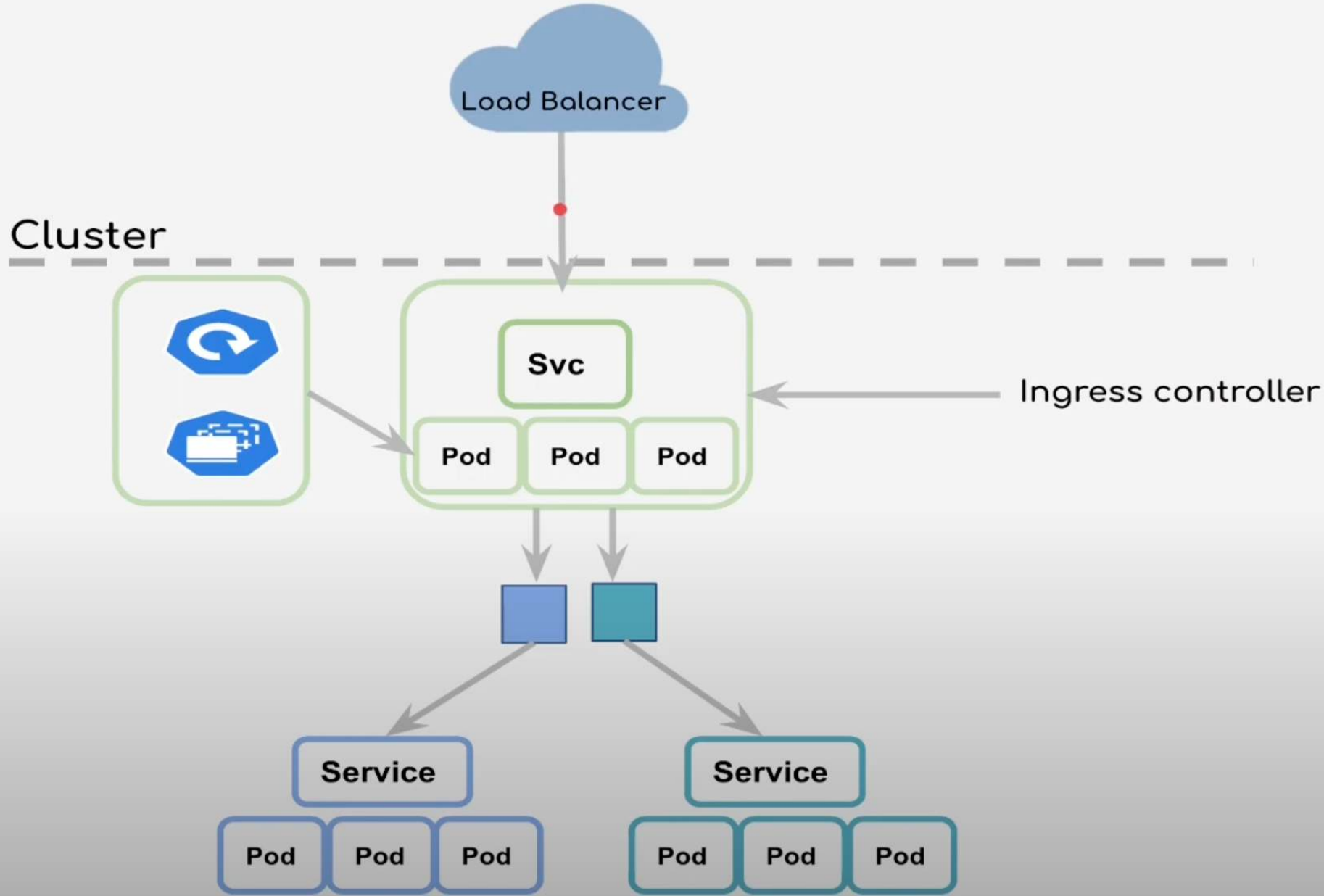

Load Balancer
Load Balancer
Cluster
Service
Service
Pod
Pod
Pod
Pod
Pod
Pod

Ingress

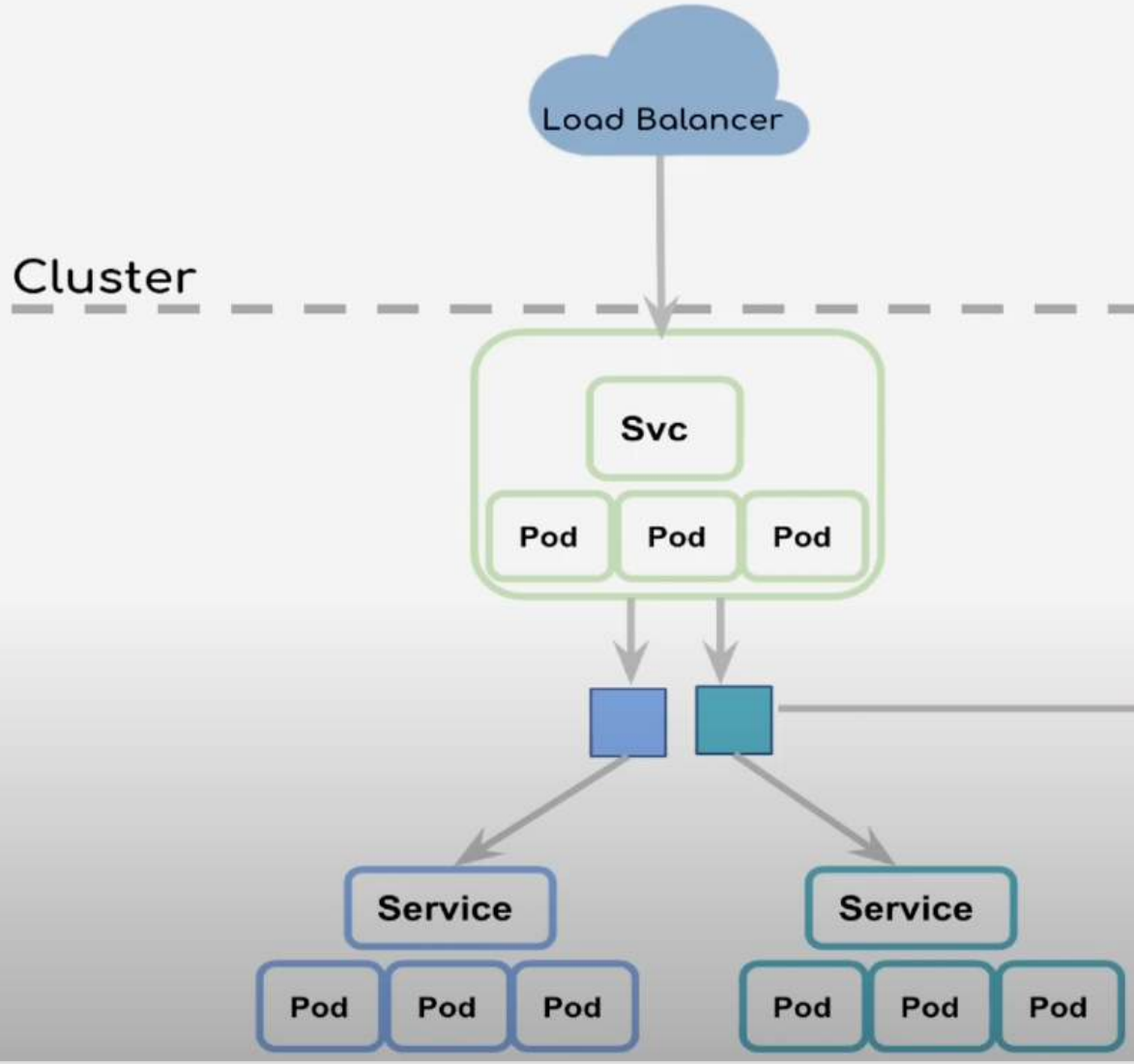

Load Balancer
Load Balancer
Cluster
Service
Service
Pod
Pod
Pod
Pod
Pod
Pod
Load Balancer
Cluster
Ingress
Service
Service
Pod
Pod
Pod
Pod
Pod
Pod


Ingress
Load Balancer
Cluster
Svc
Pod
Pod
Pod
Ingress controller
Ingress Object
Service
Pod
Pod
Pod
Service
Pod
Pod
Pod

Ingress

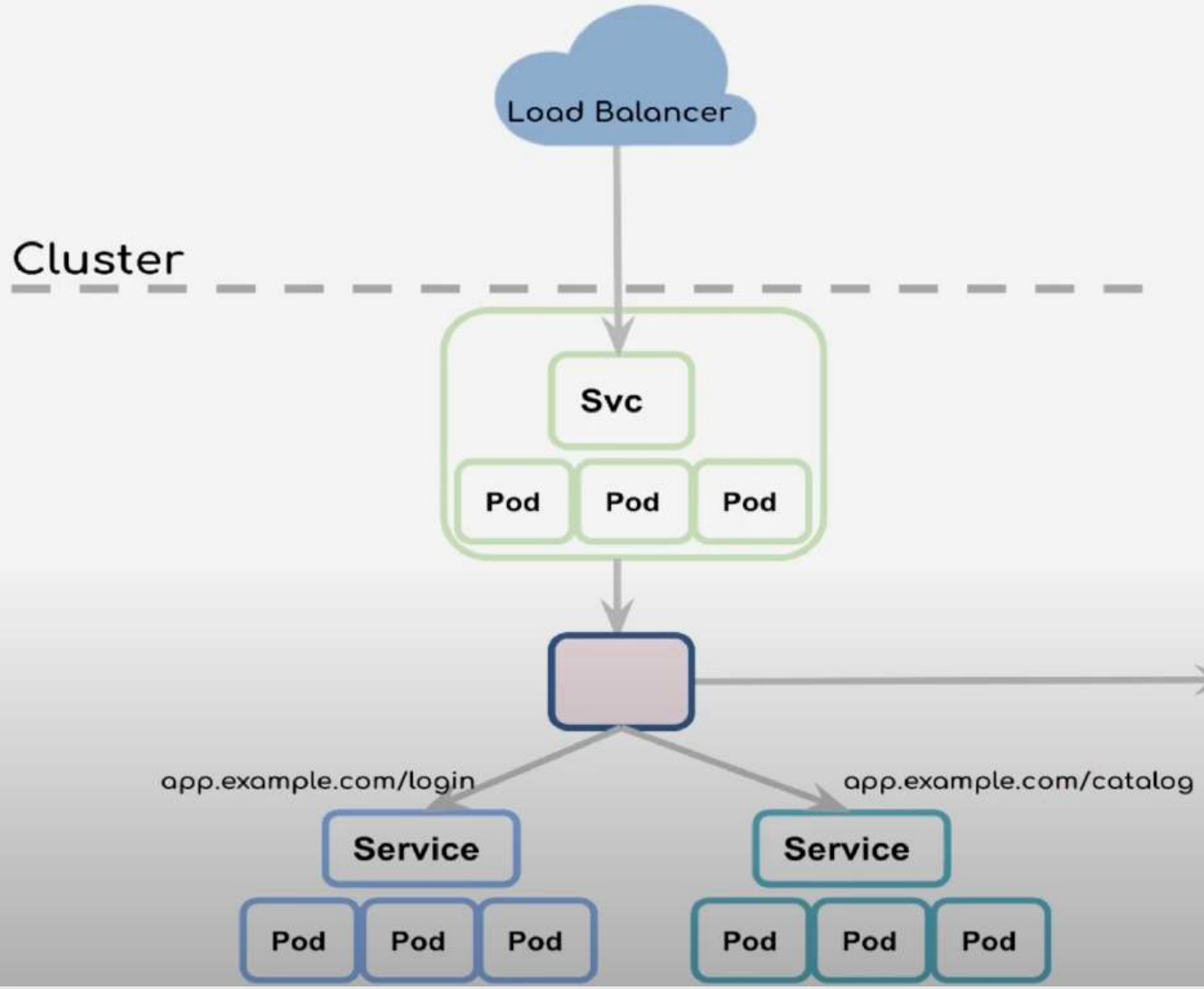

Load Balancer
Cluster
Svc
Pod
Pod
Pod
Ingress controller
Service
Pod
Pod
Pod
Service
Pod
Pod
Pod

Ingress



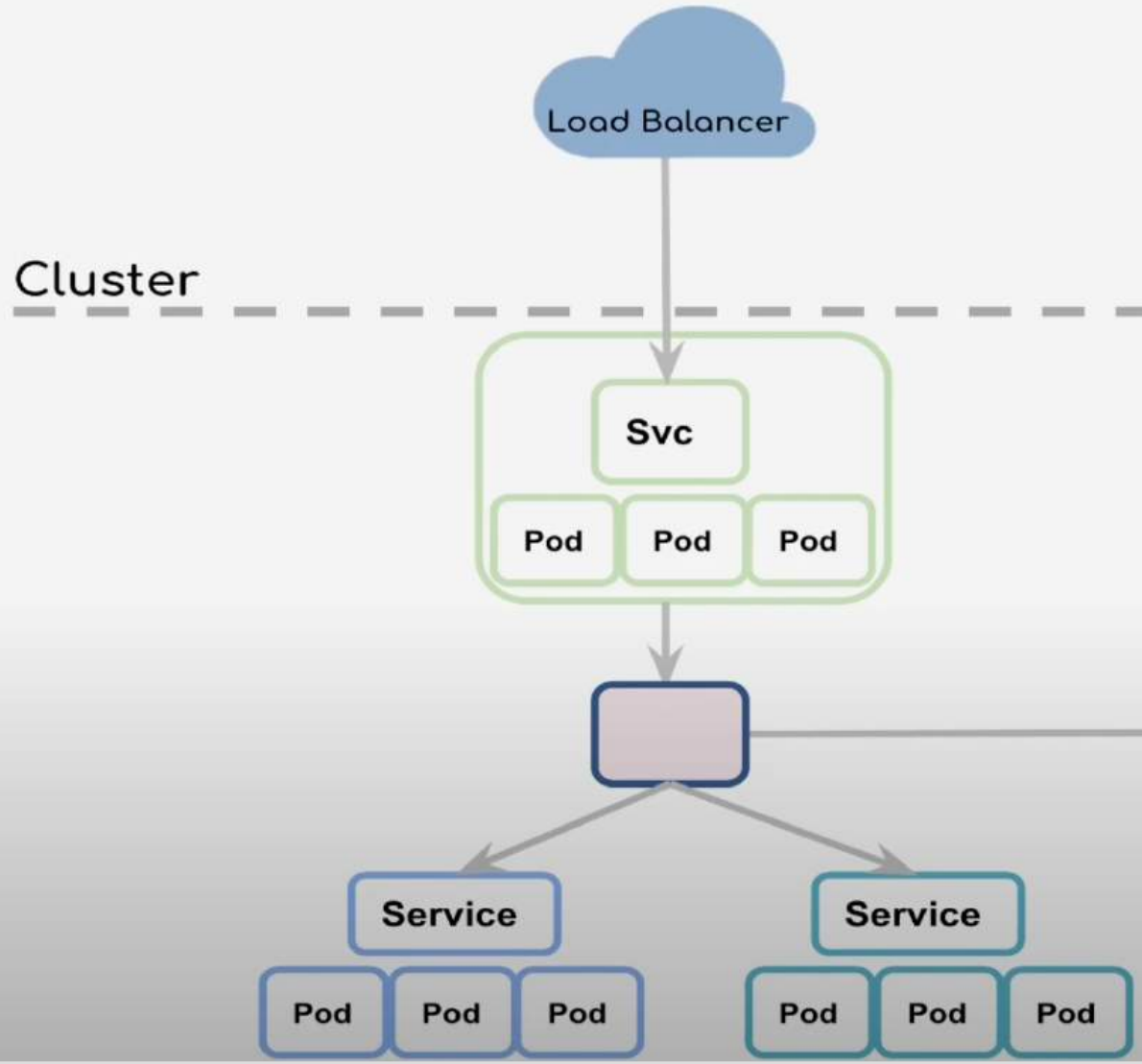

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: catalog
spec:
  ingressClassName: nginx
  rules:
  - http:
      paths:
      - path: /catalog
        pathType: Prefix
        backend:
          service:
            name: catalog
            port:
              number: 80
```

Ingress



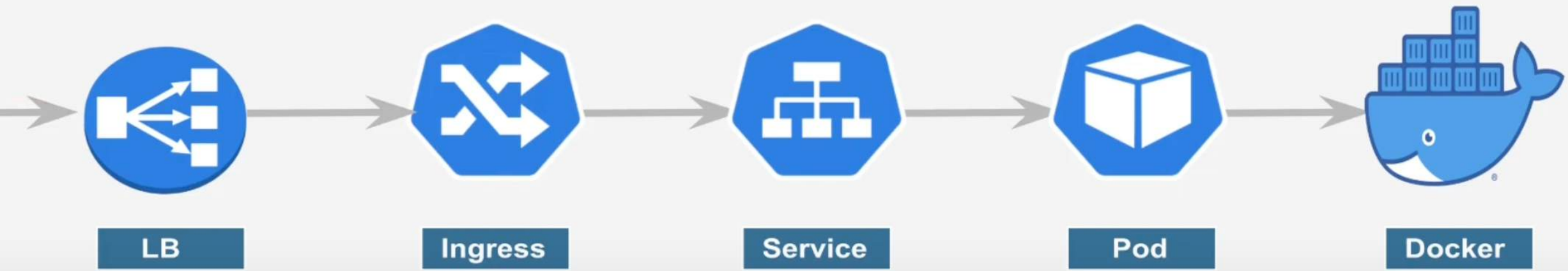

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: example
spec:
  rules:
  - host: app.example.com
    http:
      paths:
      - path: /login
        pathType: Prefix
        backend:
          service:
            name: login
            port:
              number: 4200
      - path: /catalog
        pathType: Prefix
        backend:
          service:
            name: catalog
            port:
              number: 8080
```

Ingress




```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: example
spec:
  rules:
  - host: login.example.com
    http:
      paths:
      - pathType: Prefix
        path: "/"
        backend:
          service:
            name: login
            port:
              number: 80
  - host: catalog.example.com
    http:
      paths:
      - pathType: Prefix
        path: "/"
        backend:
          service:
            name: catalog
            port:
              number: 80
```

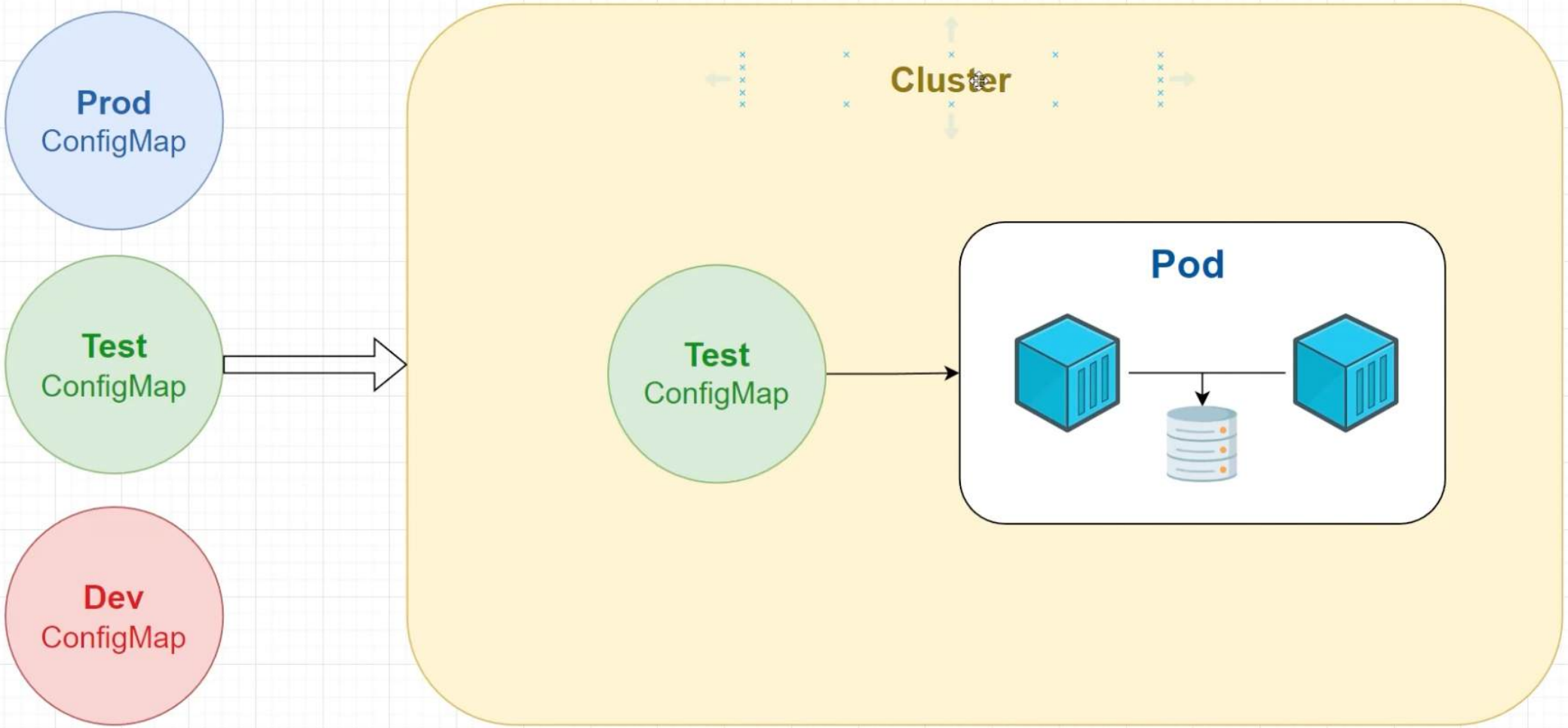
Ingress



Config Maps

A **ConfigMap** lets you **store configuration** for other **objects** to use. They accept **key-value pairs** as their values. **ConfigMaps** are designed to store **config parameters and inject** them into **running pods.**

ConfigMaps are ideal for most situations where you want to supply **environment-specific configuration values to your pods** such as IP address of your application's **database server** or the URL of a **proxy service.**


Prod
ConfigMap
Test
ConfigMap
Dev
ConfigMap
Cluster
Test
ConfigMap
Pod

Secrets

A Kubernetes **Secret** is **sensitive information** – login **usernames** and **passwords, tokens, keys**, etc – that is used within a Kubernetes environment.

The **primary purpose** of **Secrets** is to **reduce the risk** of exposing **sensitive data** while **deploying applications** on Kubernetes.

Secrets have a **size limit of 1 MB**. When it comes to **implementation**, you can either **mount Secrets as volumes or expose them as environment variables** inside the Pod manifest files.