Kubernetes Arabic Course

Ahm

Kubernetes From Scratch - Hands-on

01 - Introduction

Ahmed ElBakry

Course Contents

01 Introduction

- Course Introduction
- What is Kubernetes
- Kubernetes Features

02 Kubernetes Architecture

- Cluster Architecture
- Master Components
- Worker Components

03 Setup Lab Cluster

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

04 Basic Concepts

- Namespaces
- Resource Quota
- Labels and Selectors
- Nodes

Course Contents

05 Pods

- Pods Explained
- Pods demo
- Multi-container pod

06 Deployments

- Deployment Explained
- Deployment Demo
- Rolling Update

07 Services

- Service Explained
- Service Types
- Service Demo

08 Ingress

- Ingress Explained
- Deploy Nginx Ingress Controller
- Ingress Demo

Course Contents

09 Deploy Your First App

- Containerize The Application
- Deploy the Application
- Volumes
- Secrets - ConfigMaps

10 Deploying With Helm

- Kubernetes Package Manager (Helm)
- Deploy Your First Helm Chart
- Helm Templates

11 Monitoring

- Deploy Prometheus
- Deploy Alert Manager
- Explore Prometheus & Alert Manager

12 Summary

- Wrap Up
- What's Next

About The Instructor

AHMED ELBAKRY

- Sr Cloud Architect
- + 12 Years IT Experience
- RHCE, VCP, Google Professional Cloud Architect
- Kubernetes, Terraform, Python, GCP, AWS, etc.

/ahmed.elbakry.39

@ahmed43068401

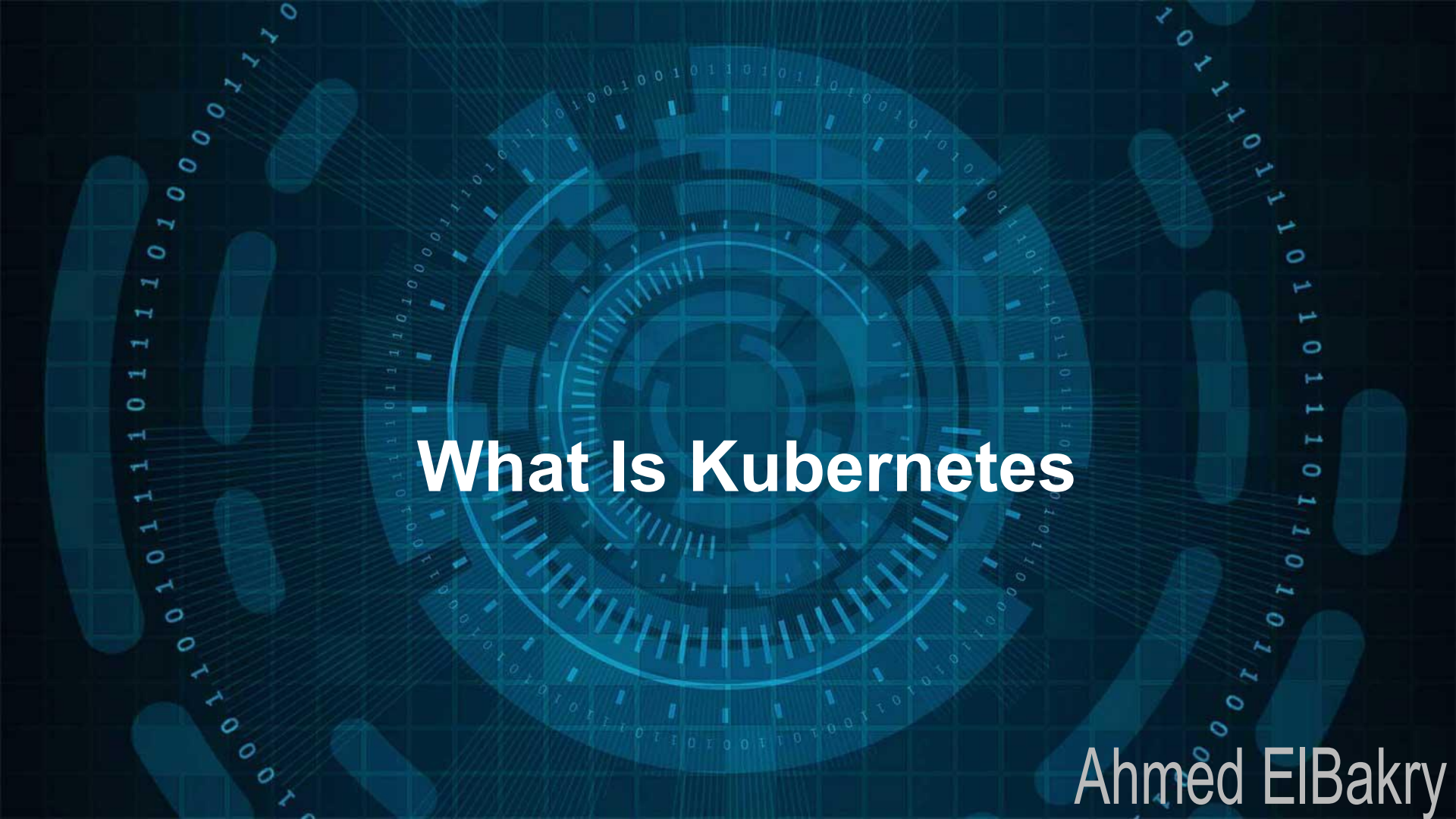
/in/ahmed-elbakry-56261134


THANK YOU
Ahmed ElBakry

Kubernetes From Scratch - Hands-on

02 - What Is Kubernetes

Ahmed ElBakry


What Is Kubernetes
Ahmed ElBakry

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

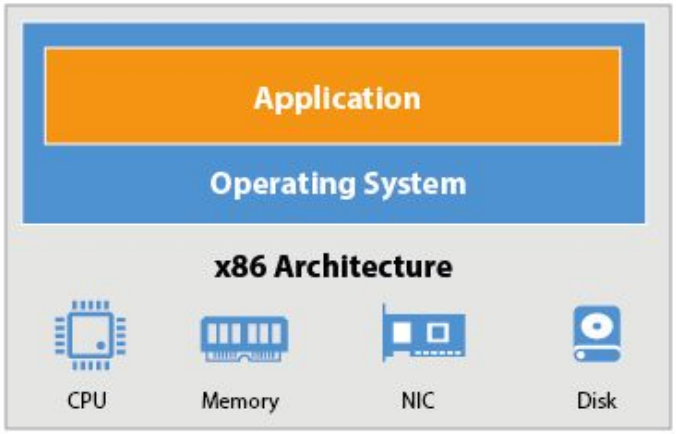
It became the industry standard for deploying containers in production

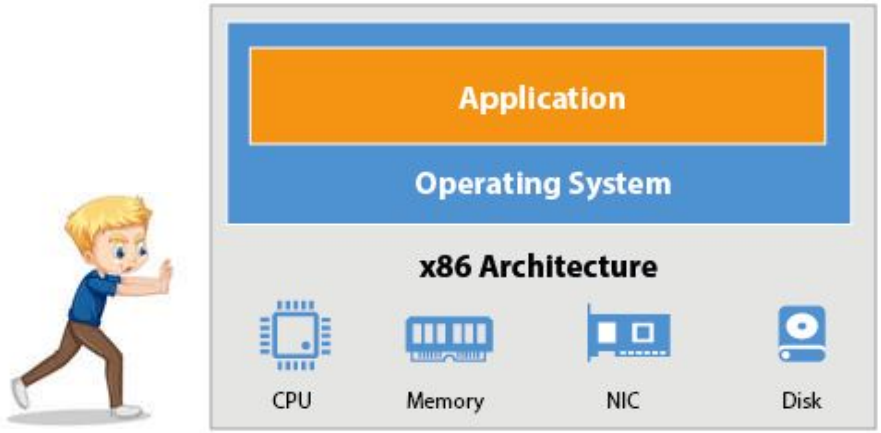
Ahmed ElBakry

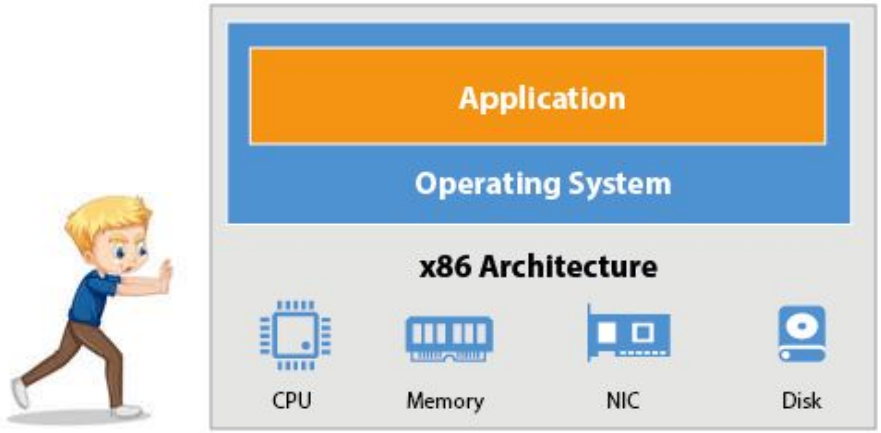
Container Orchestration

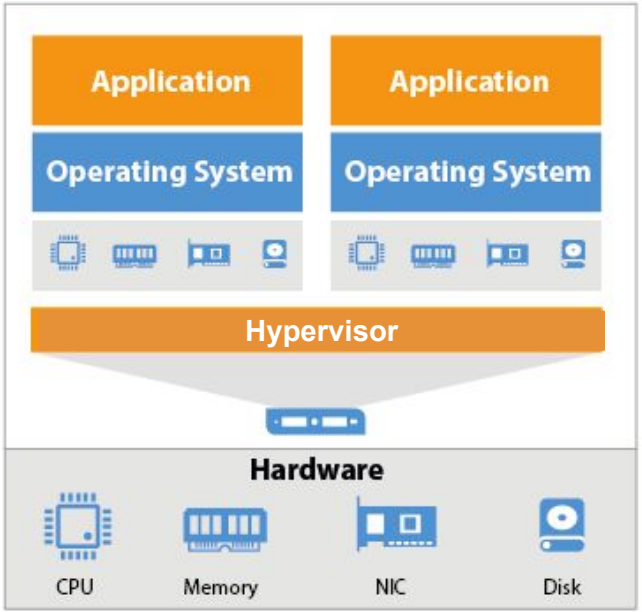
It's an open source container orchestration platform, to automate and manage containerized applications.

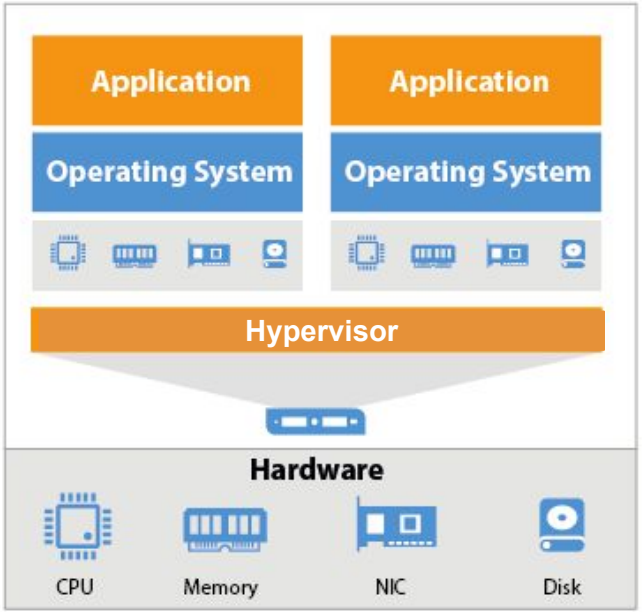
Ahmed ElBakry

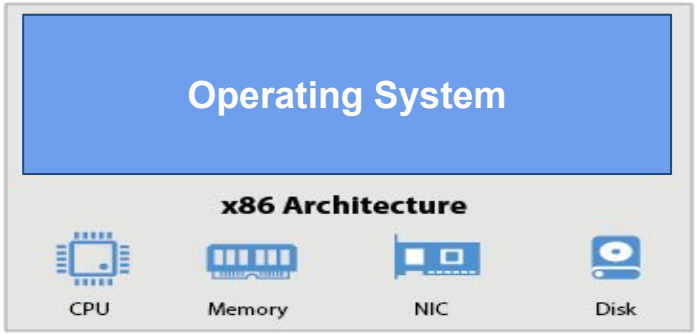


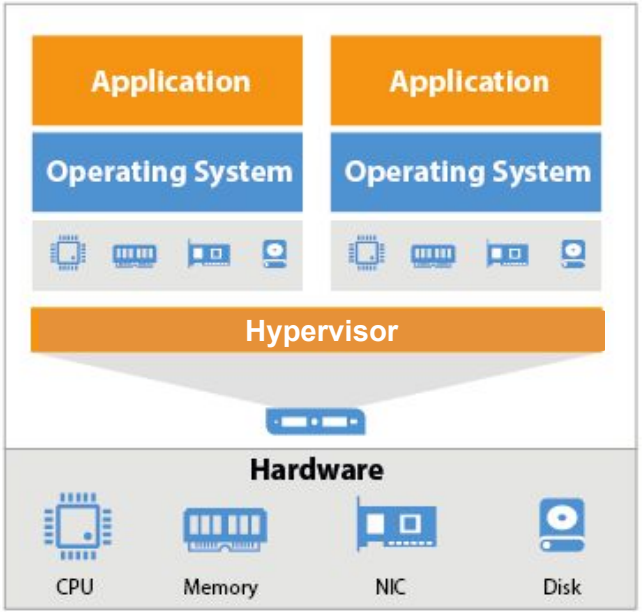


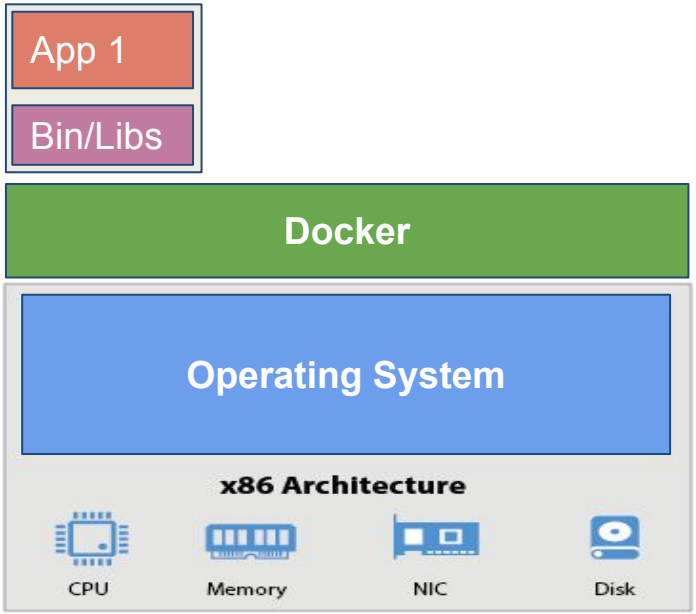


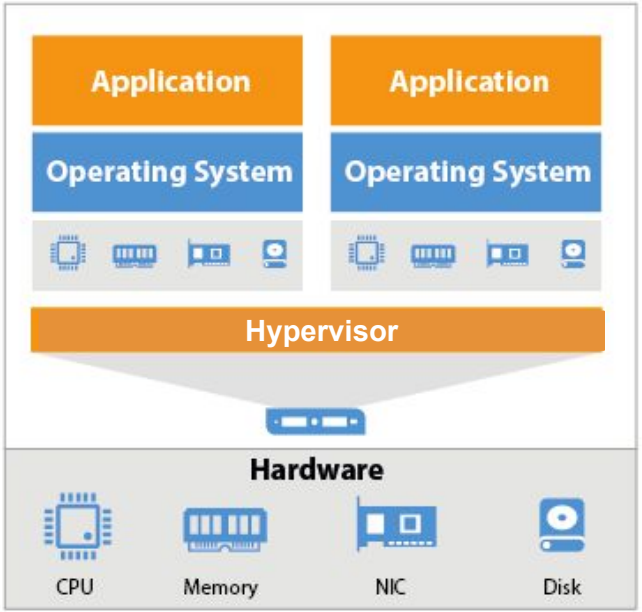


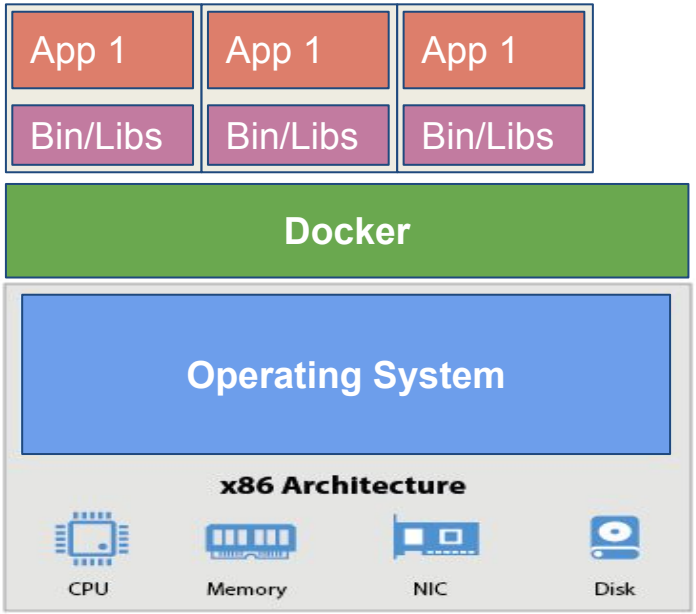


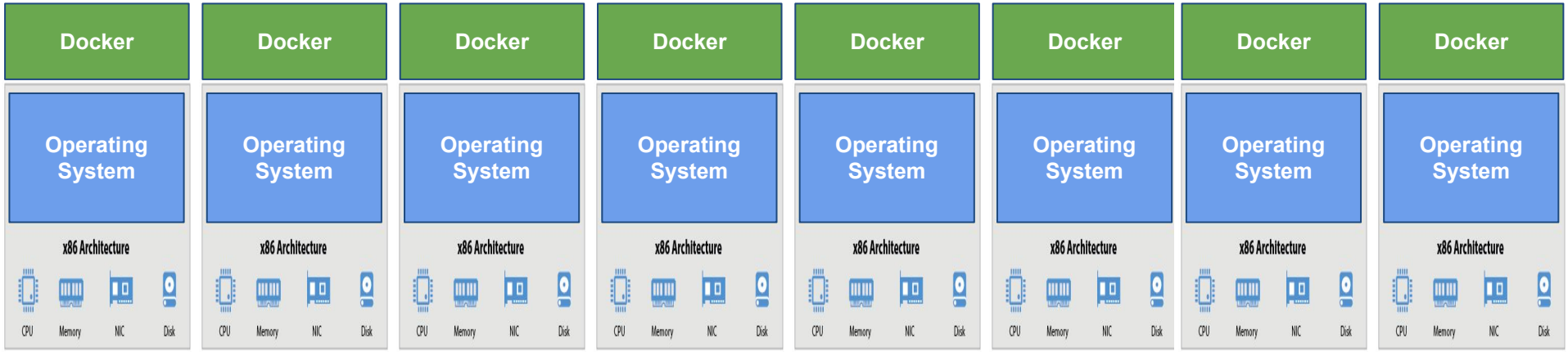


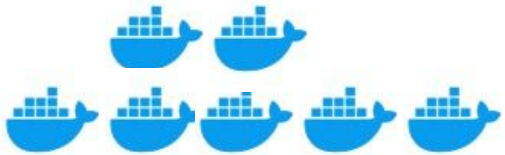


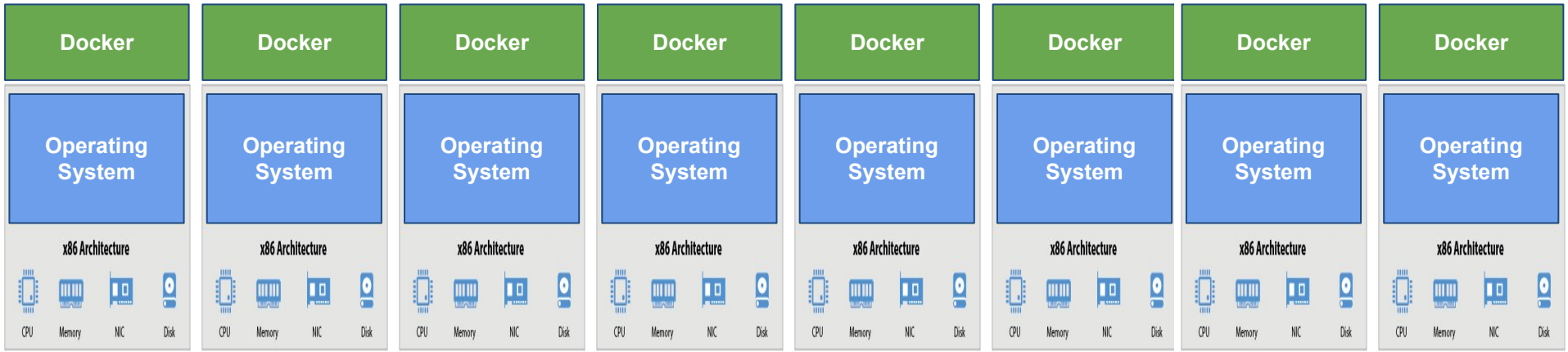


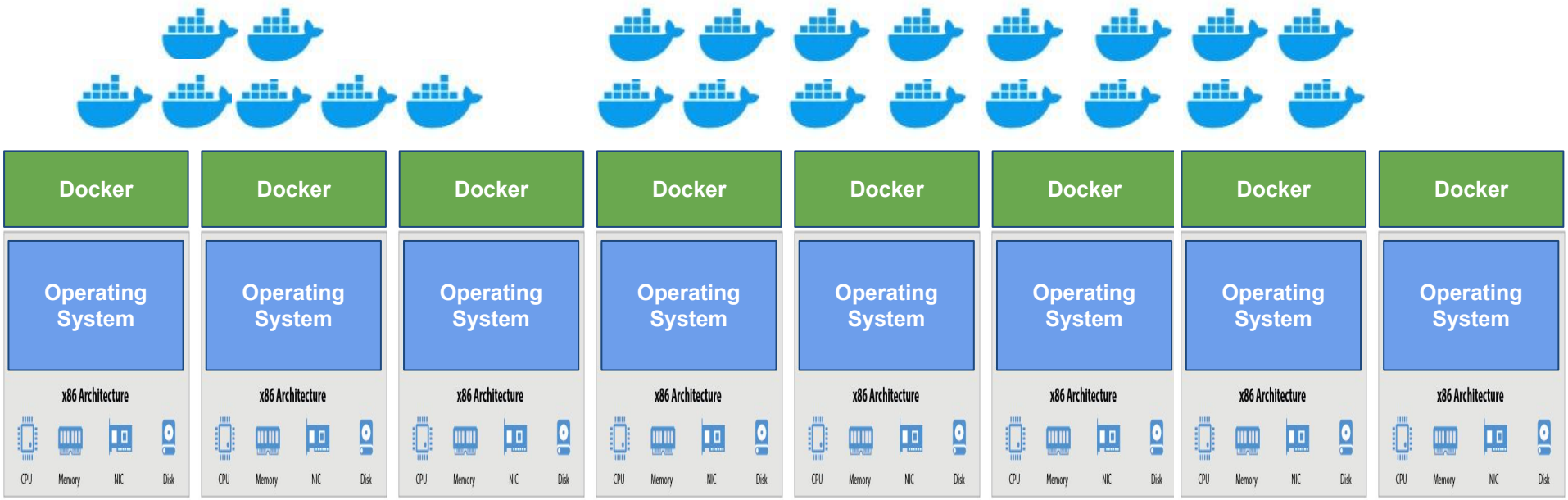


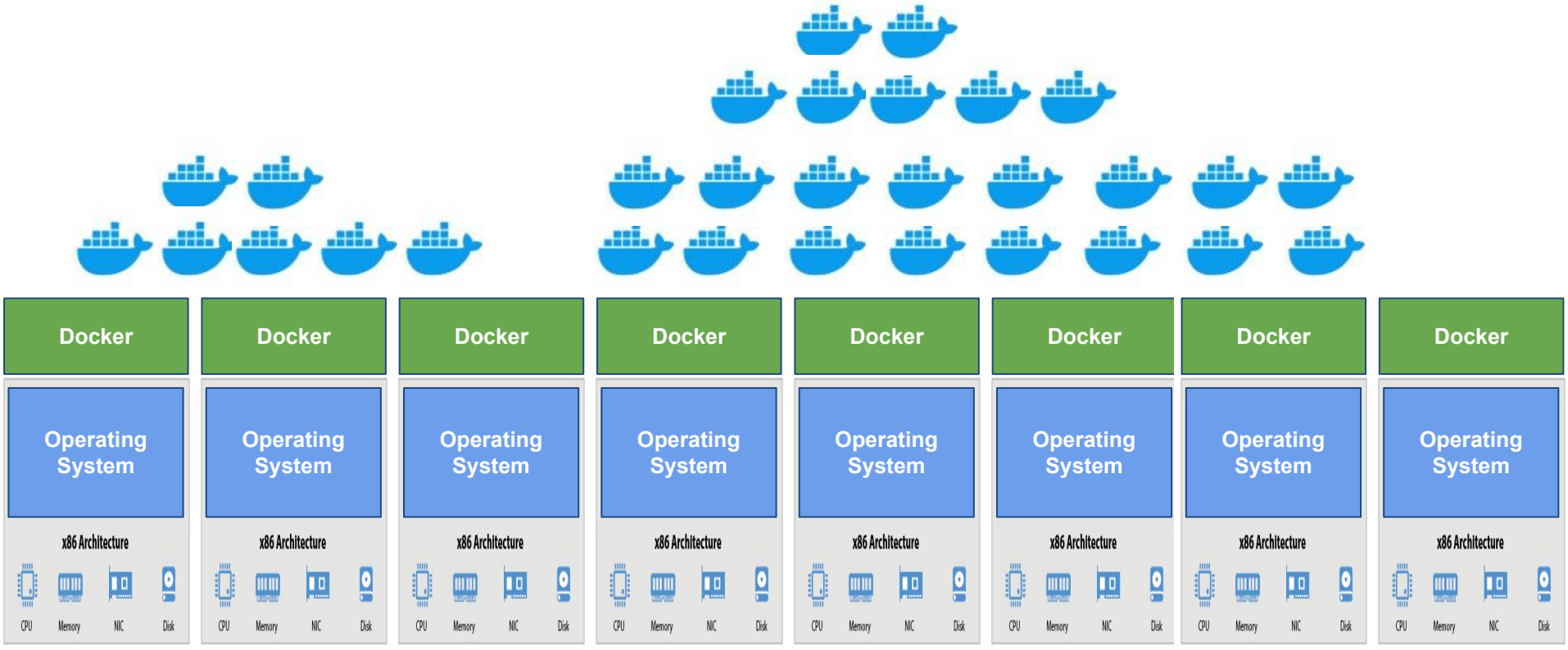


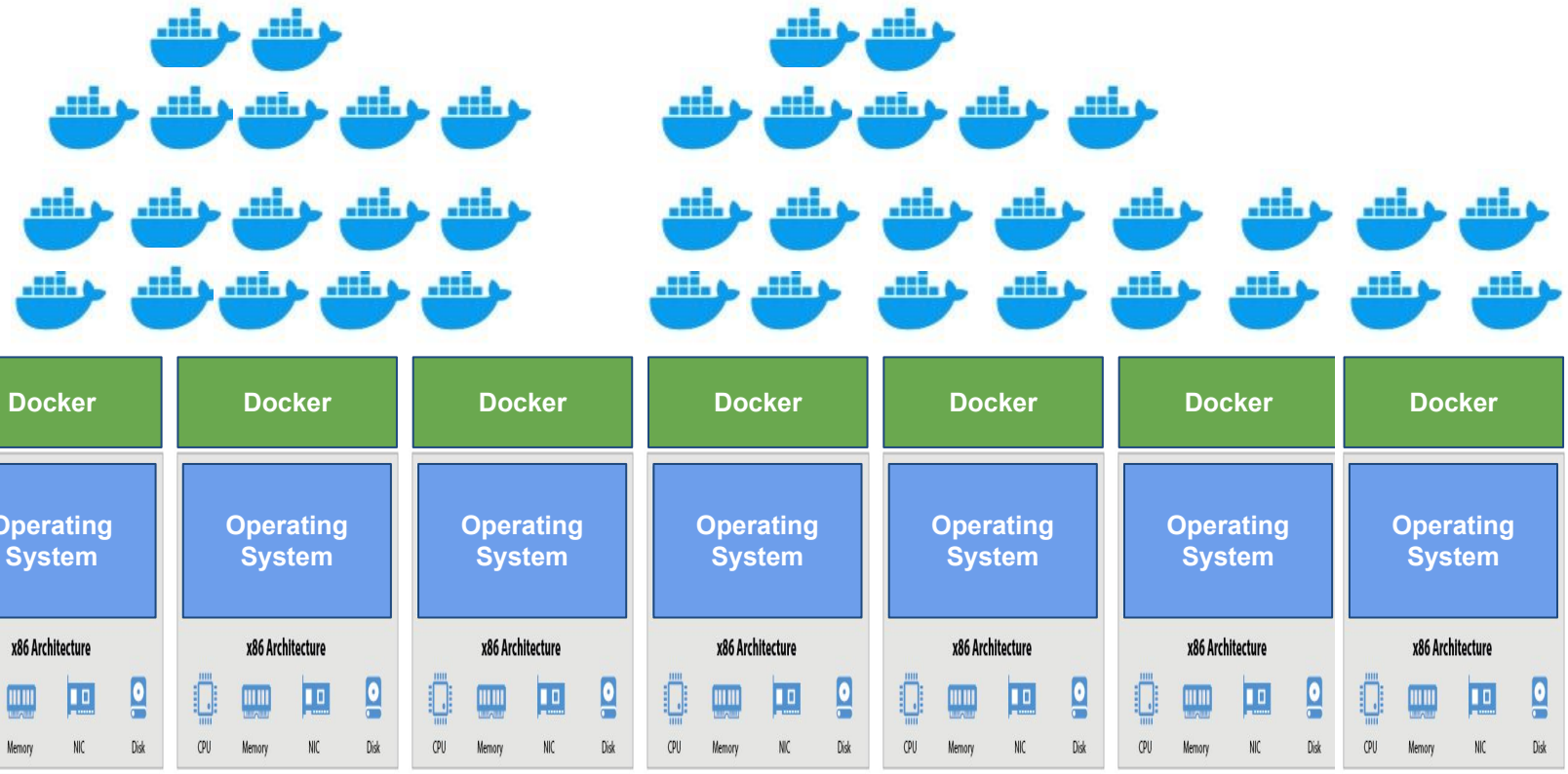

Docker
Operating System
x86 Architecture
CPU
Memory
NIC
Disk

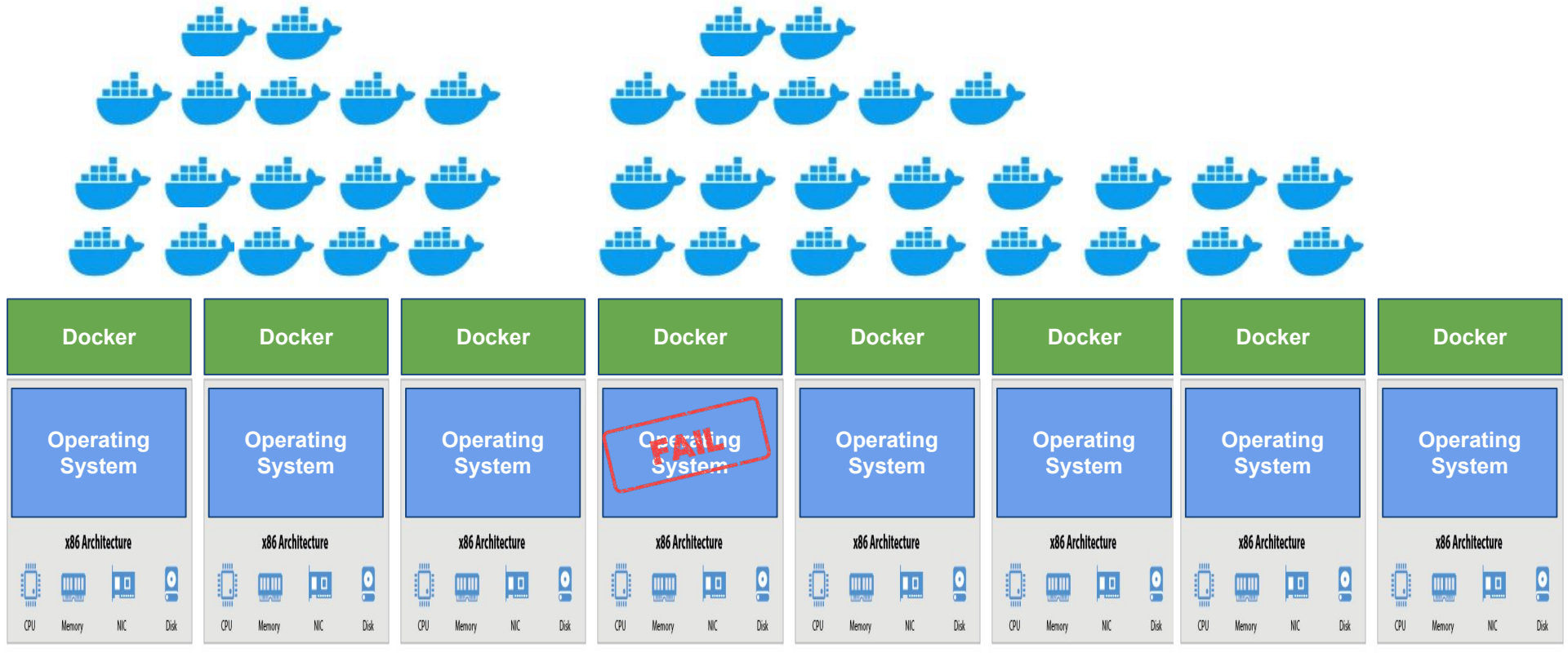


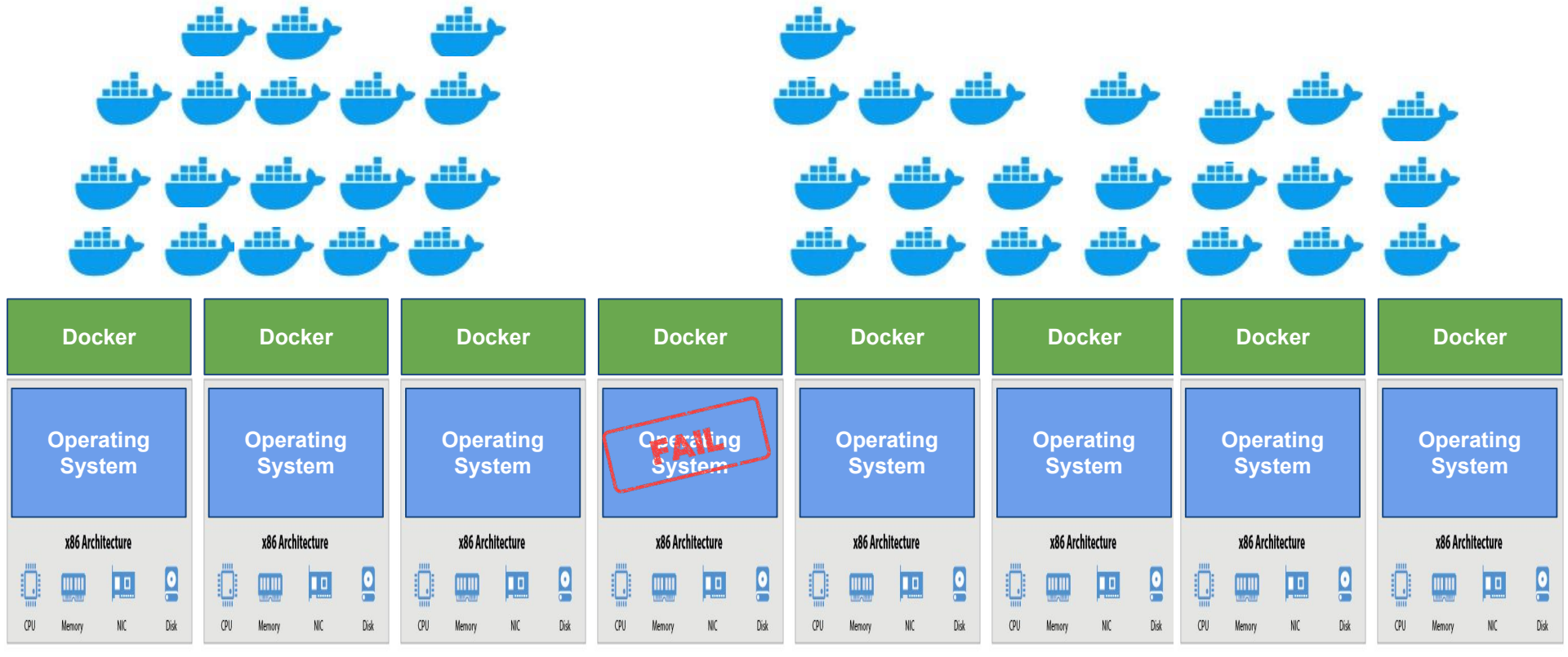

Docker
Operating System
x86 Architecture
CPU
Memory
NIC
Disk

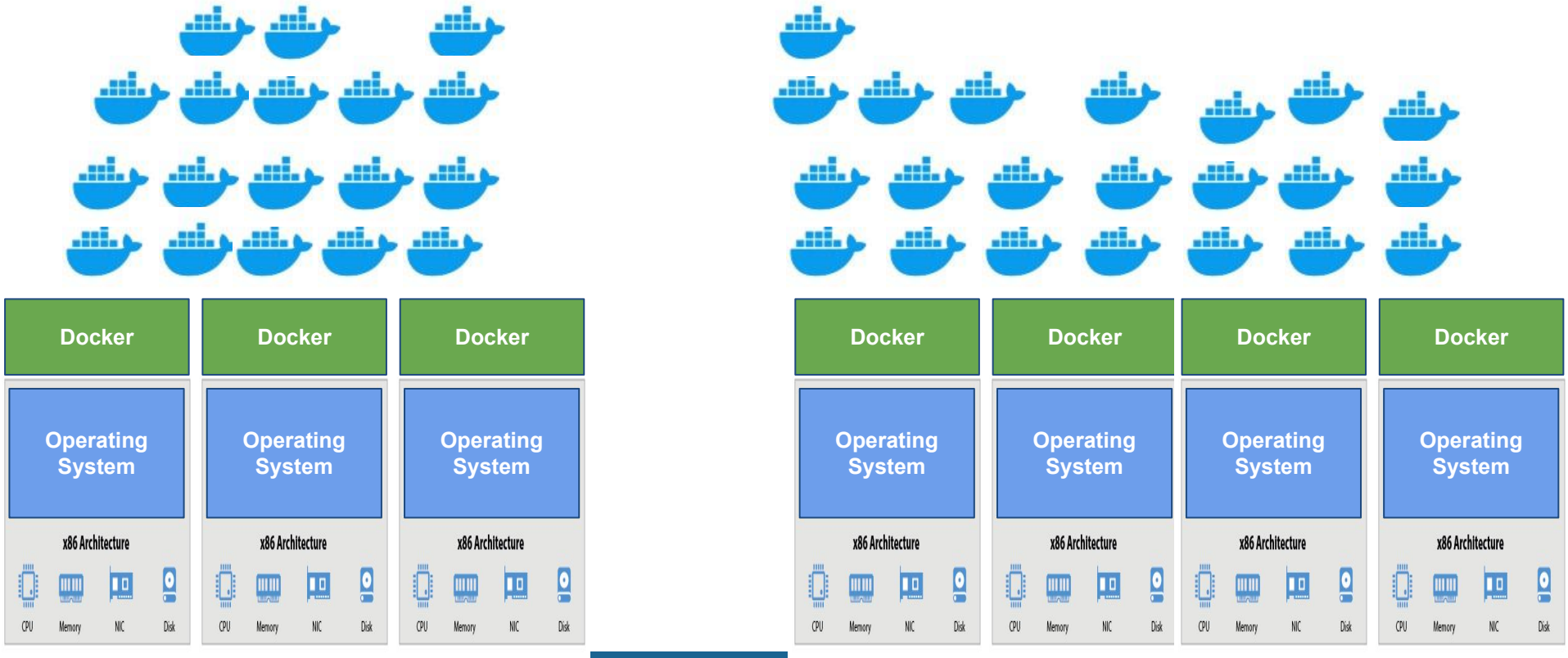

Docker
Operating System
x86 Architecture
CPU
Memory
NIC
Disk

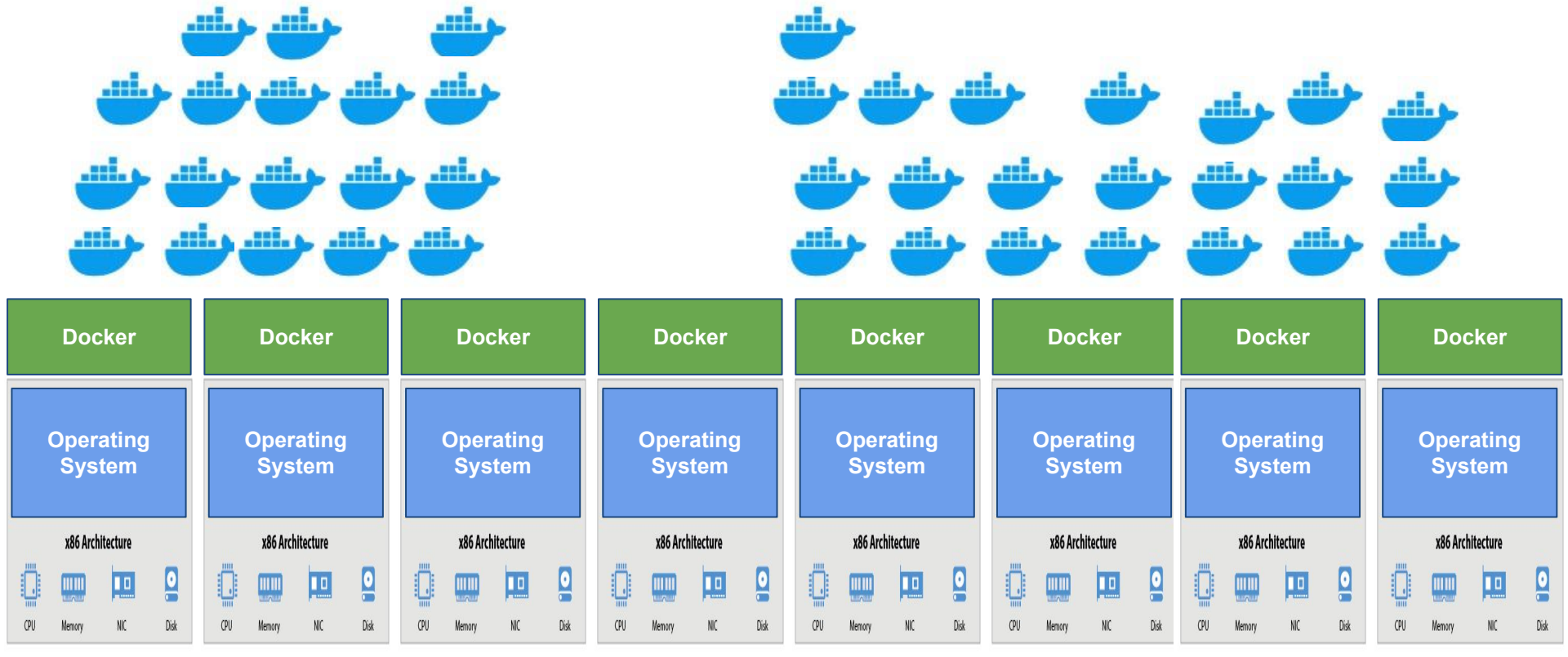

Docker
Operating System
x86 Architecture
CPU
Memory
NIC
Disk

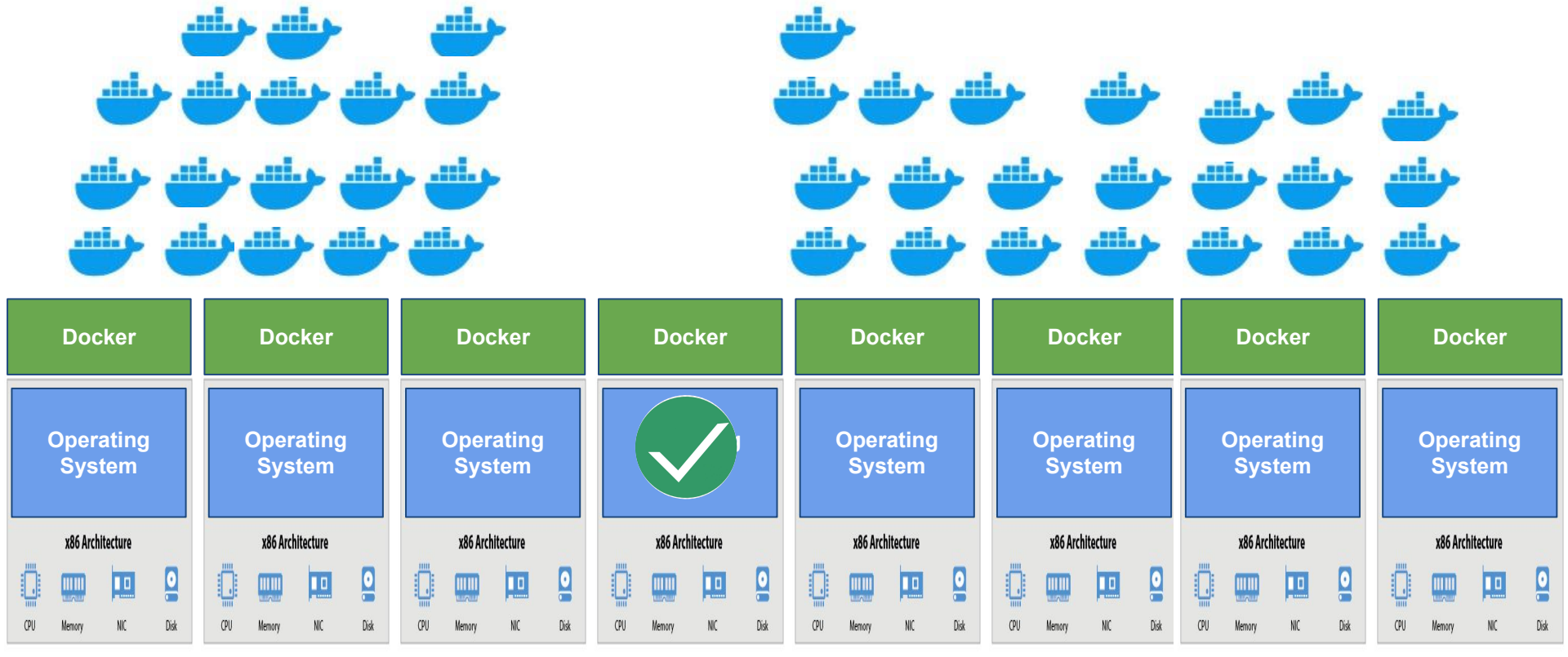

Docker
Operating System
x86 Architecture
CPU
Memory
NIC
Disk

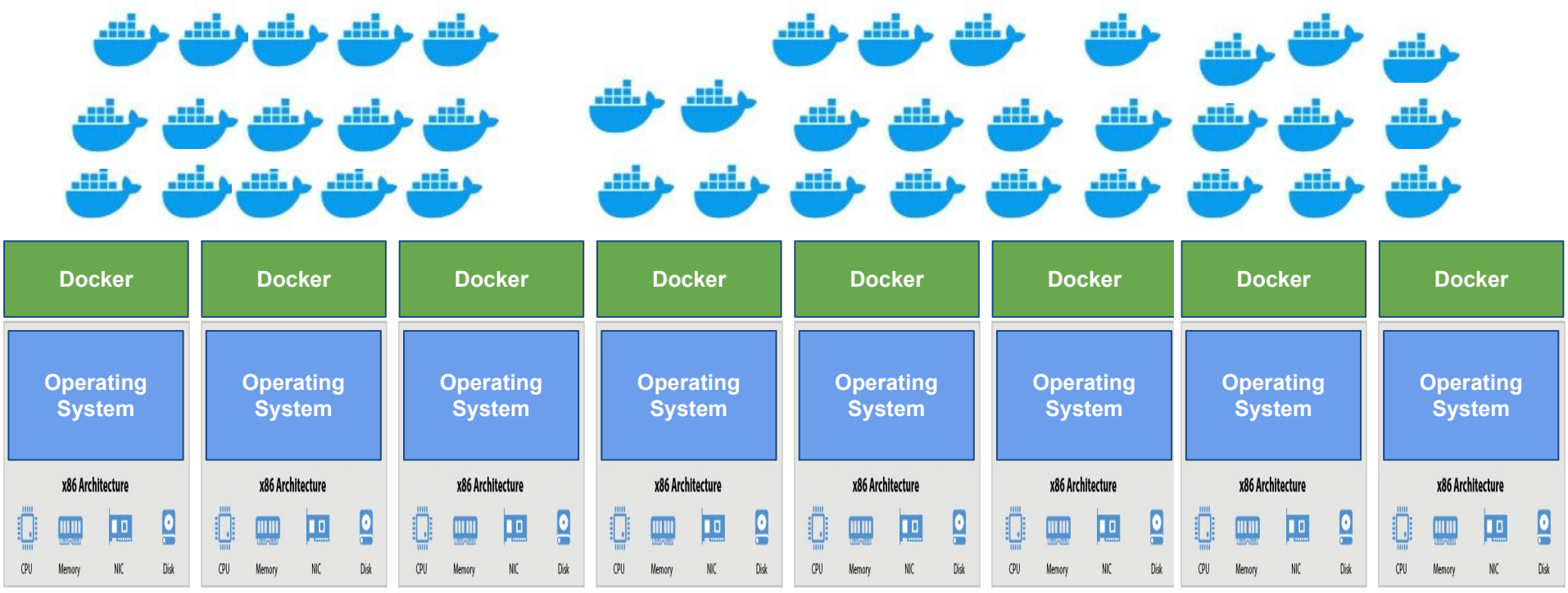

Docker
Operating System
x86 Architecture
CPU
Memory
NIC
Disk
FAIL


Docker
Docker
Docker
Docker
Docker
Docker
Docker
Docker
Operating System
Operating System
Operating System
Operating System
FAIL
Operating System
Operating System
Operating System
Operating System
x86 Architecture
CPU
Memory
NIC
Disk


Docker
Docker
Docker
Operating System
Operating System
Operating System
x86 Architecture
x86 Architecture
x86 Architecture
CPU
Memory
NIC
Disk
Docker
Docker
Docker
Docker
Operating System
Operating System
Operating System
Operating System
x86 Architecture
x86 Architecture
x86 Architecture
x86 Architecture


Docker
Operating System
x86 Architecture
CPU
Memory
NIC
Disk


Docker
Operating System
x86 Architecture
CPU
Memory
NIC
Disk


Docker
Operating System
x86 Architecture
CPU
Memory
NIC
Disk

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

Ahmed ElBakry

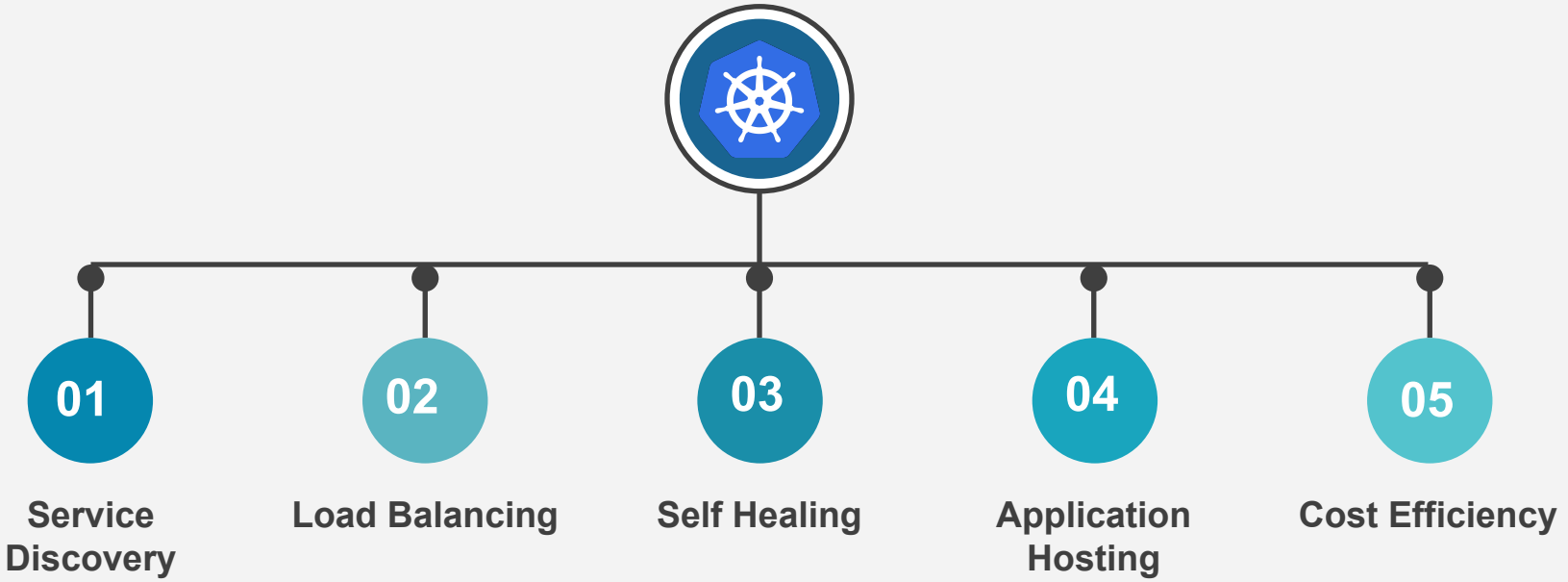

THANK YOU
Ahmed ElBakry

Kubernetes From Scratch - Hands-on

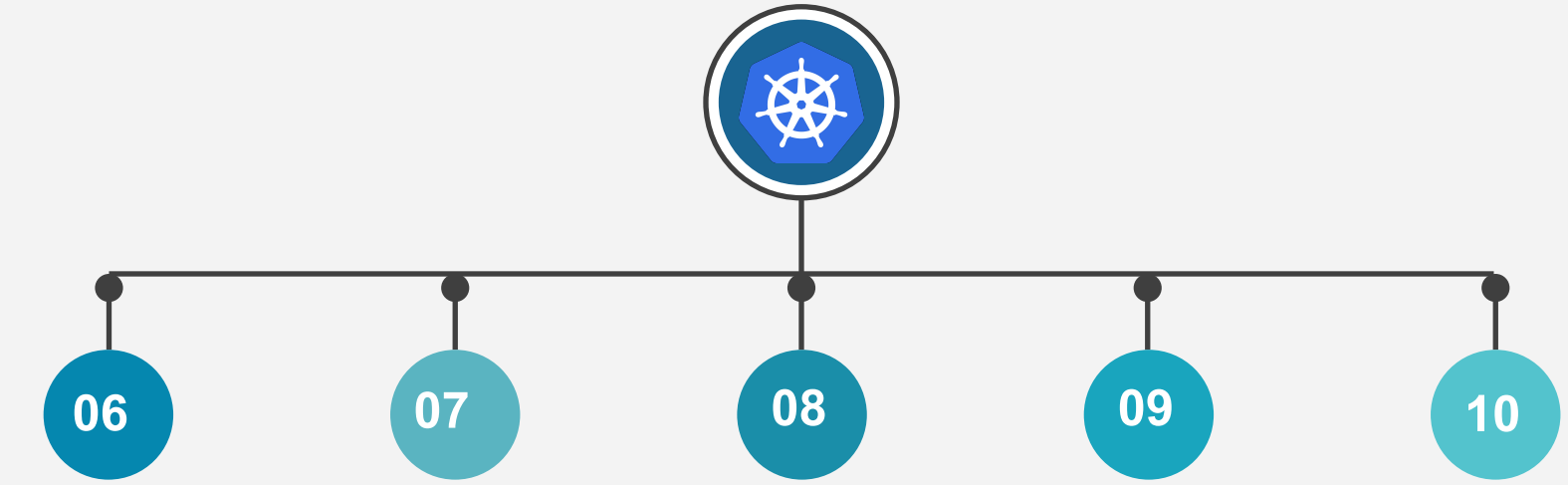
03 - Kubernetes Features

Ahmed ElBakry

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.


THANK YOU
Ahmed ElBakry

Module 2

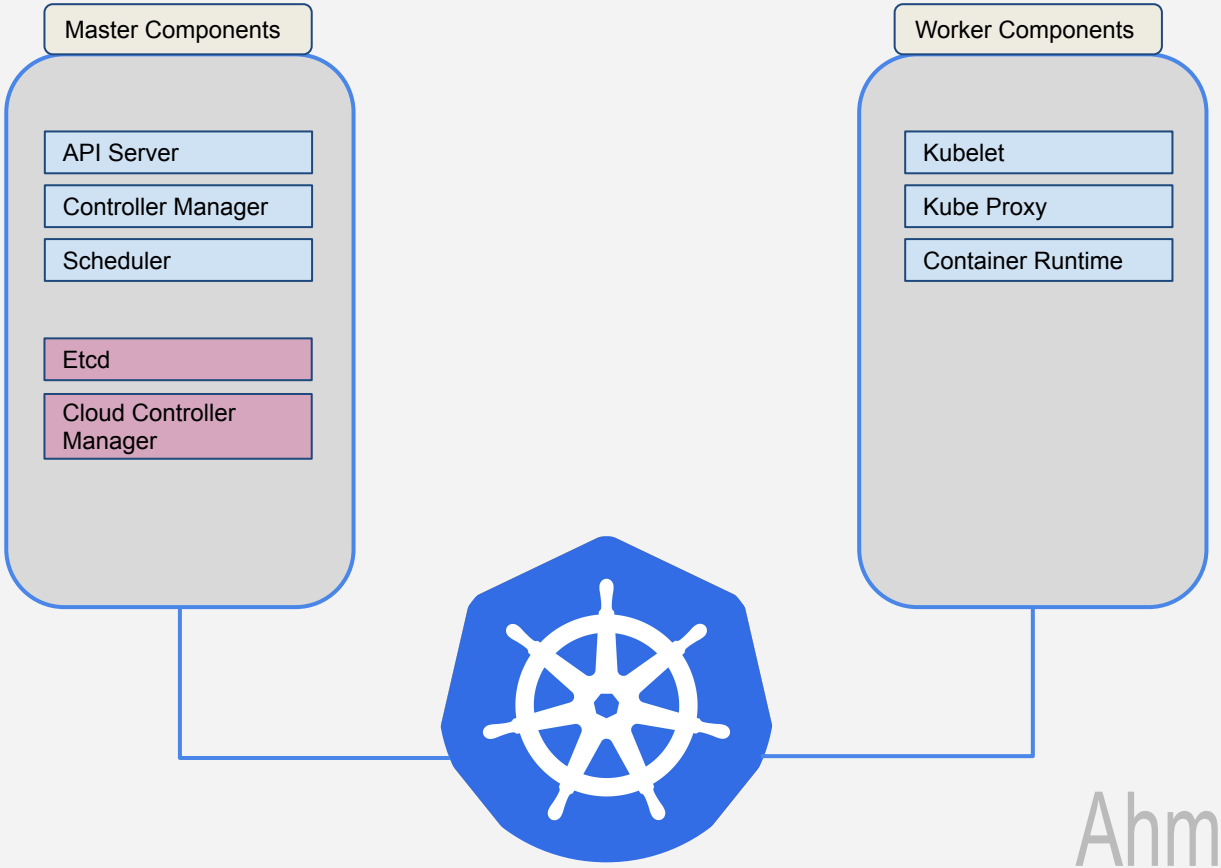
Kubernetes Architecture

Kubernetes From Scratch - Hands-on

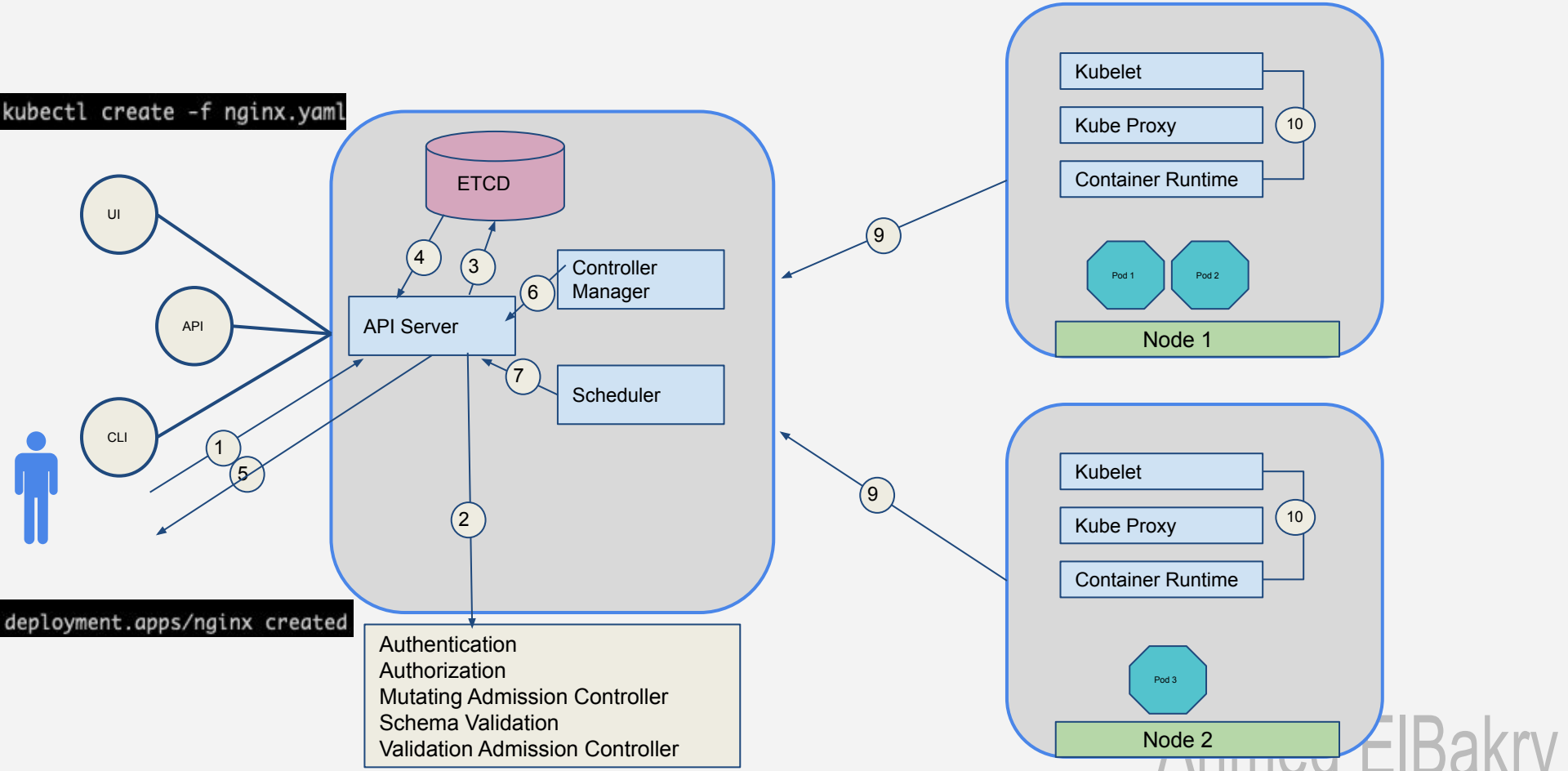
04 - Cluster Architecture

Ahmed ElBakry

Cluster Architecture

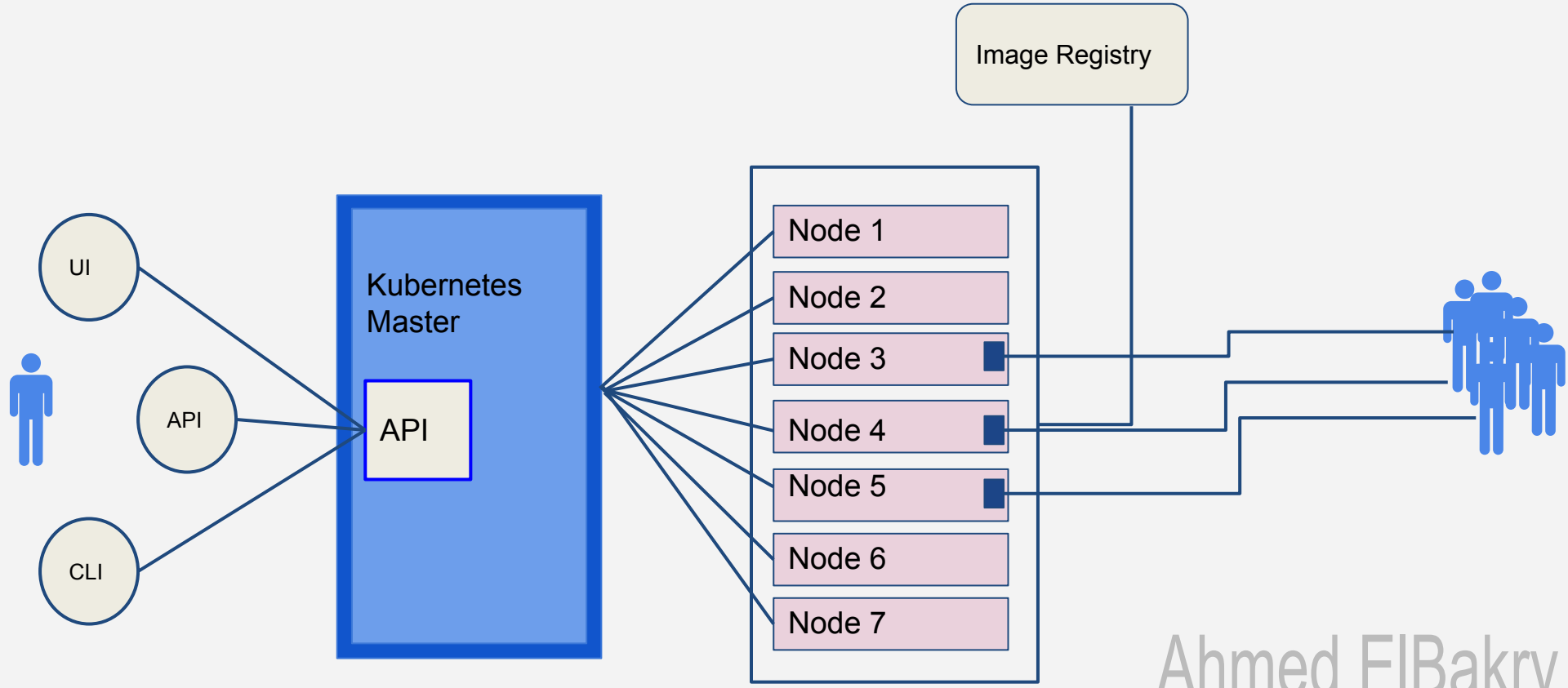


Cluster Architecture




THANK YOU
Ahmed ElBakry

Cluster Architecture

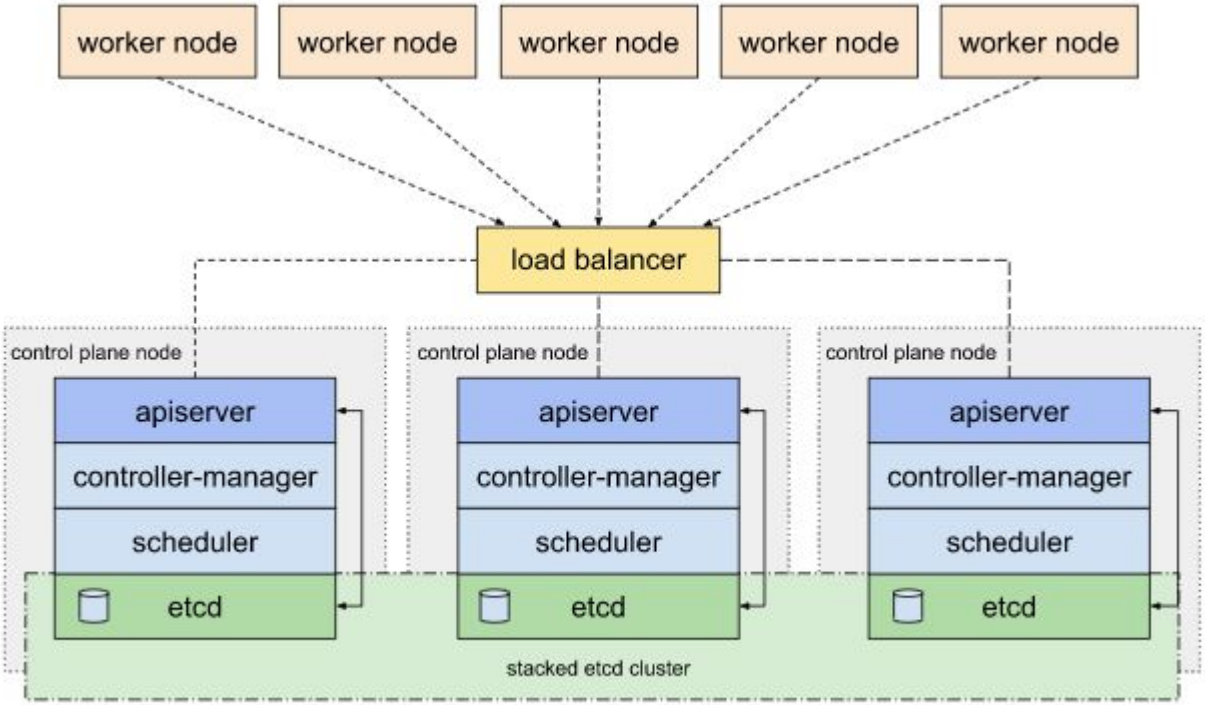


Kubernetes From Scratch - Hands-on

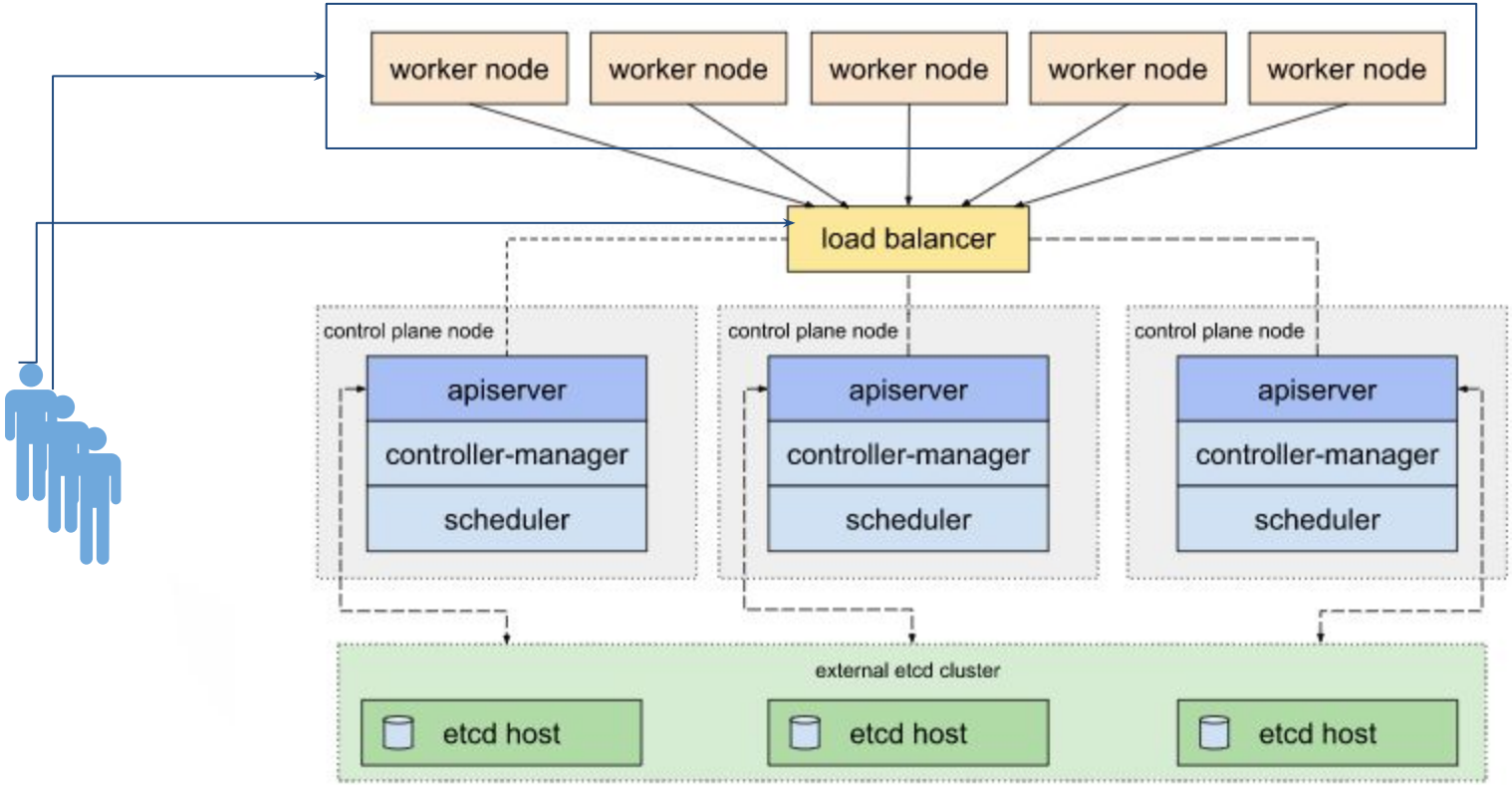
05 - Master Components

Ahmed ElBakry

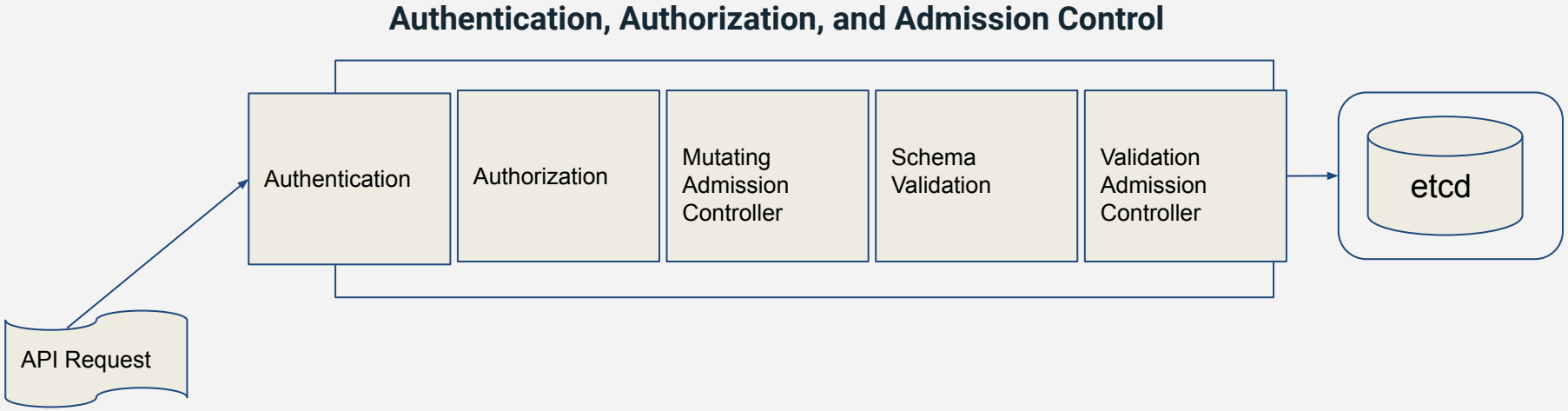
Master Components



Master Components



Master Components



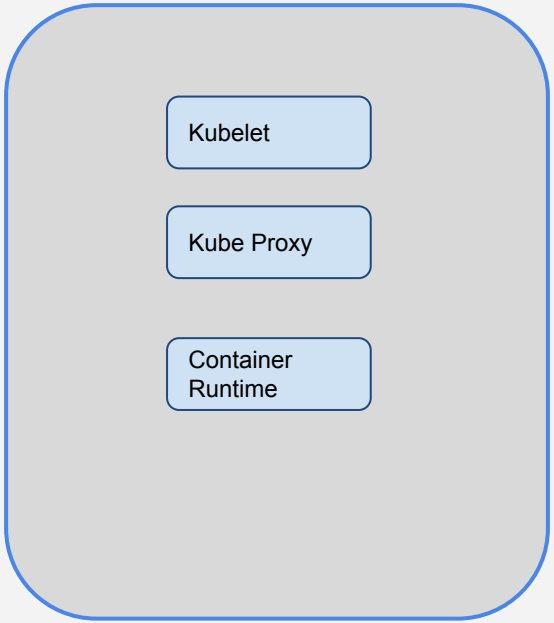

THANK YOU
Ahmed ElBakry

Kubernetes From Scratch - Hands-on

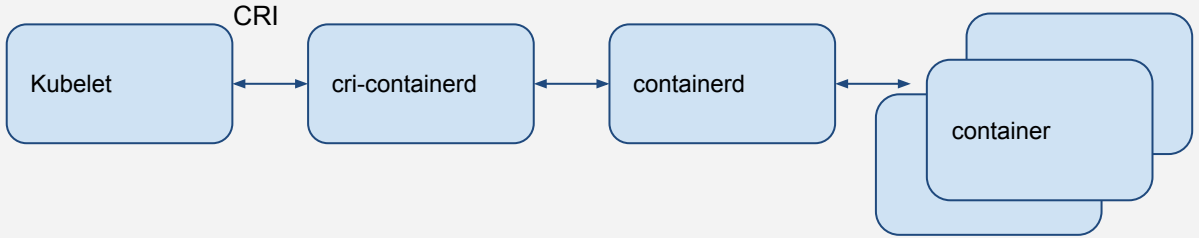
06 - Worker Components

Ahmed ElBakry

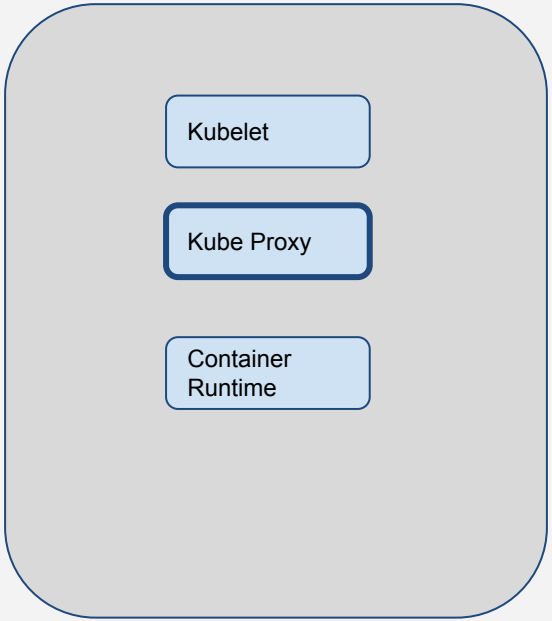
Worker Components



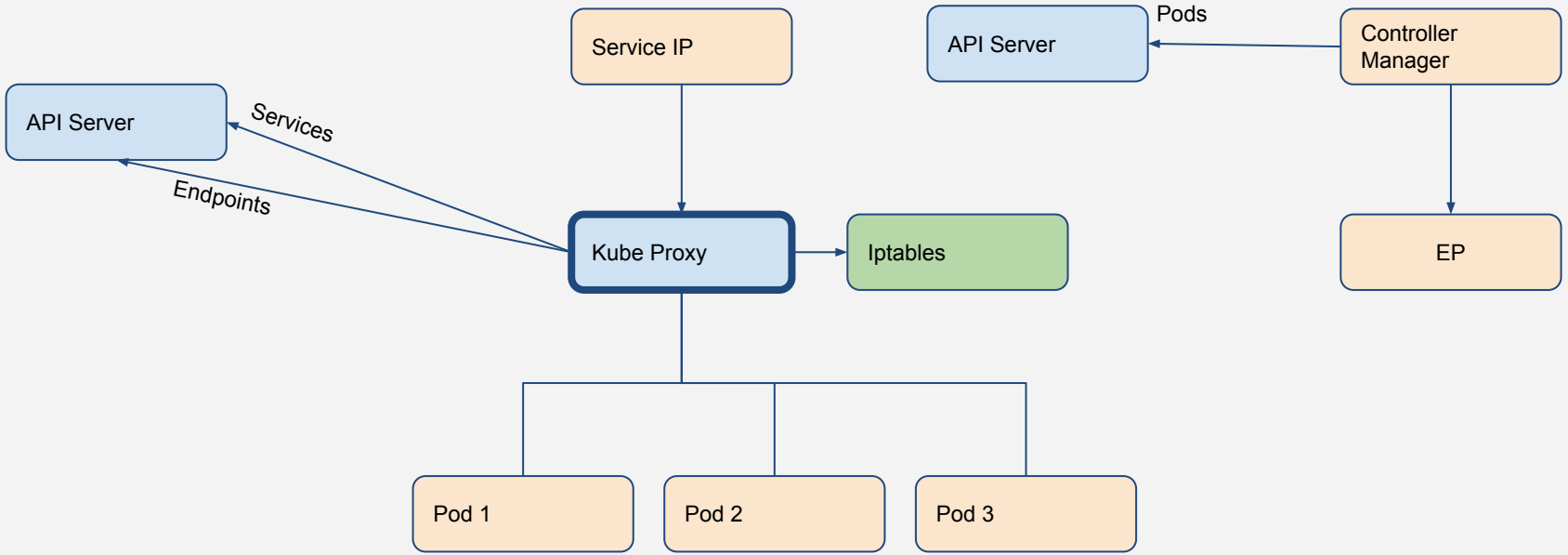
Worker Components



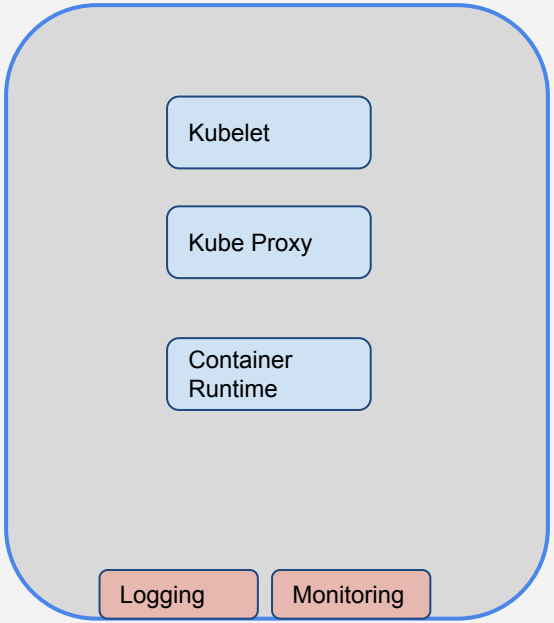
Worker Components



Worker Components



Worker Components



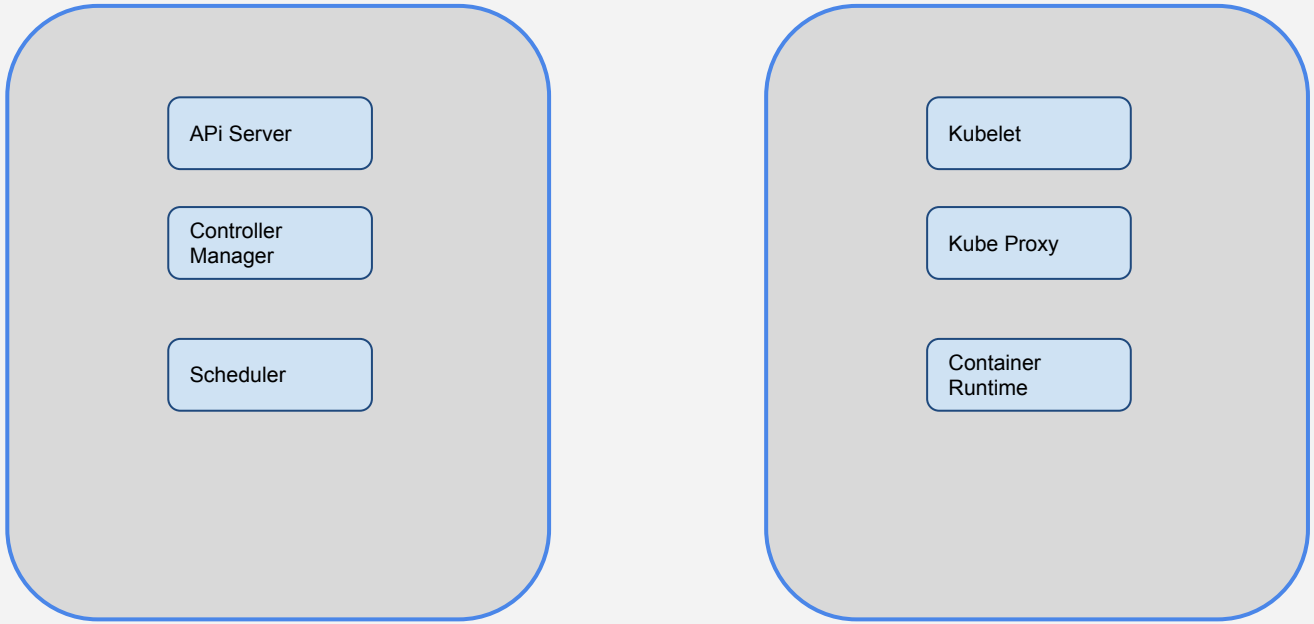

THANK YOU
Ahmed ElBakry

Kubernetes From Scratch - Hands-on

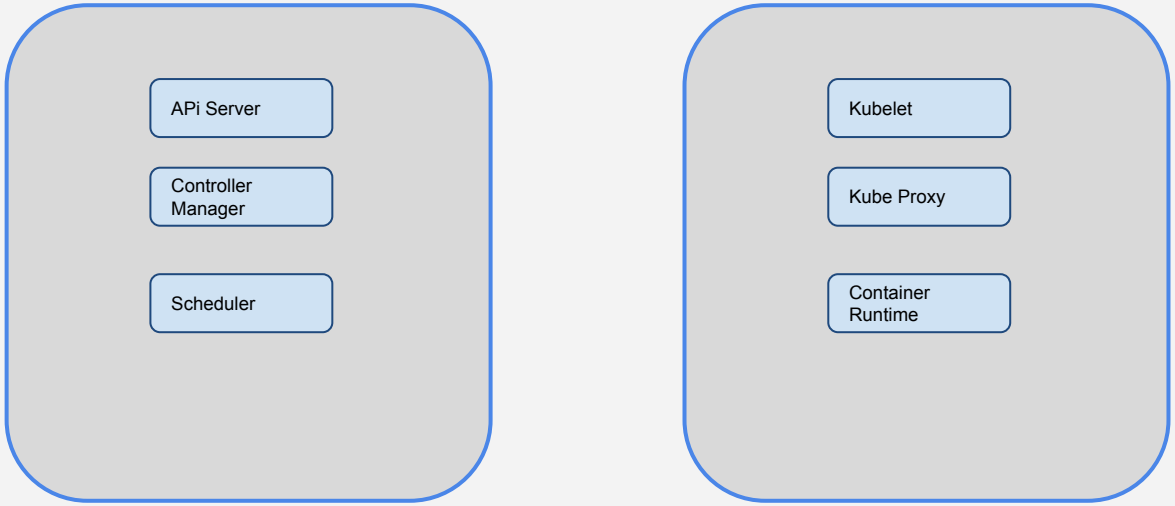
07 - Cluster Architecture Demo

Ahmed ElBakry

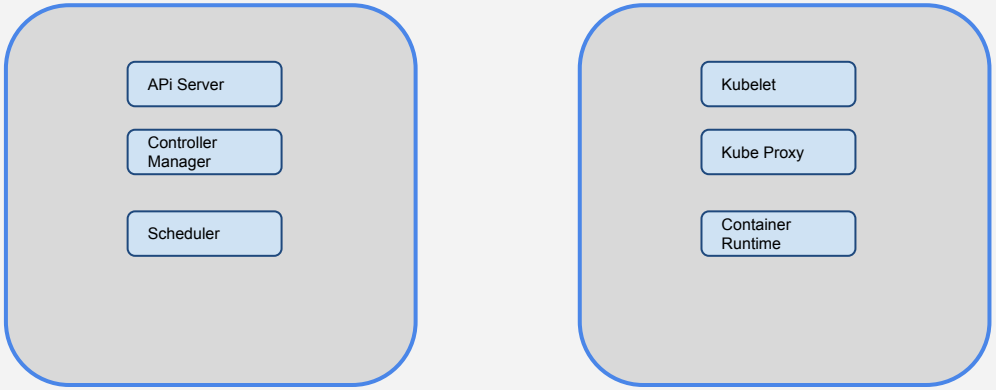
Kubernetes Architecture Demo



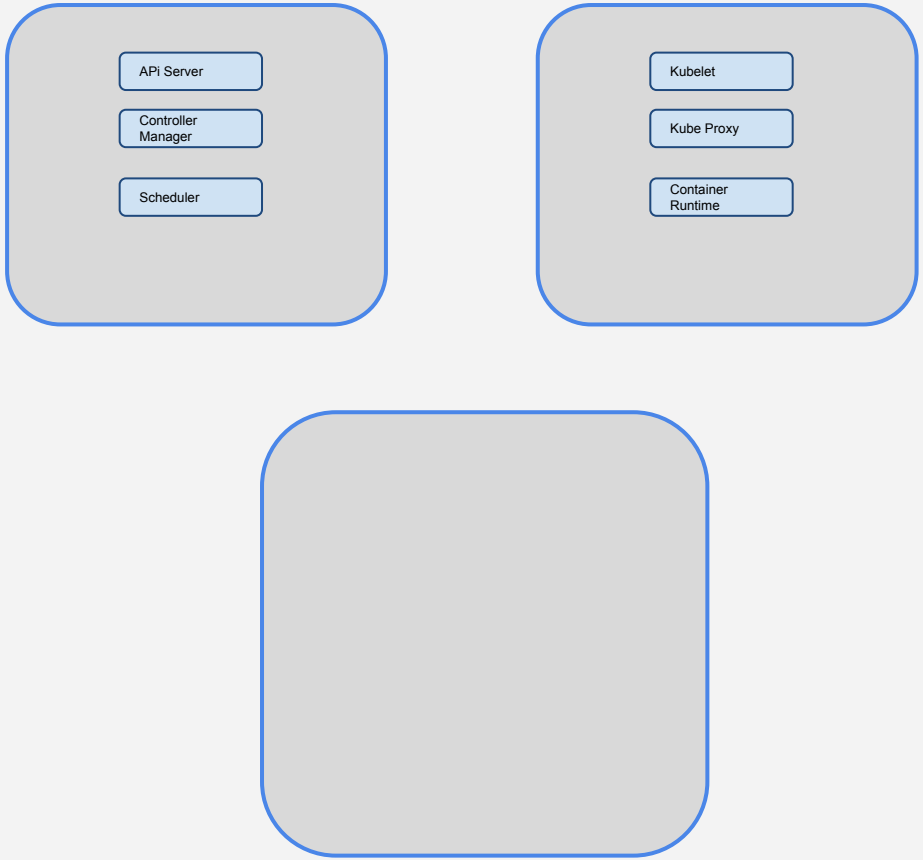
Kubernetes Architecture Demo



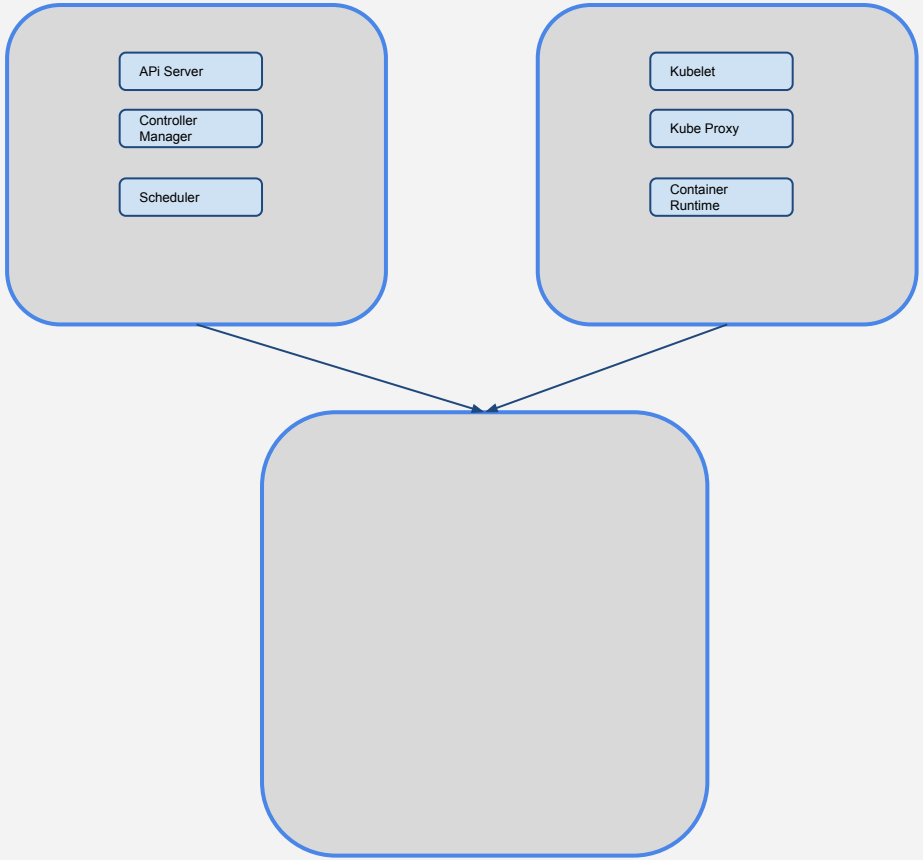
Kubernetes Architecture Demo



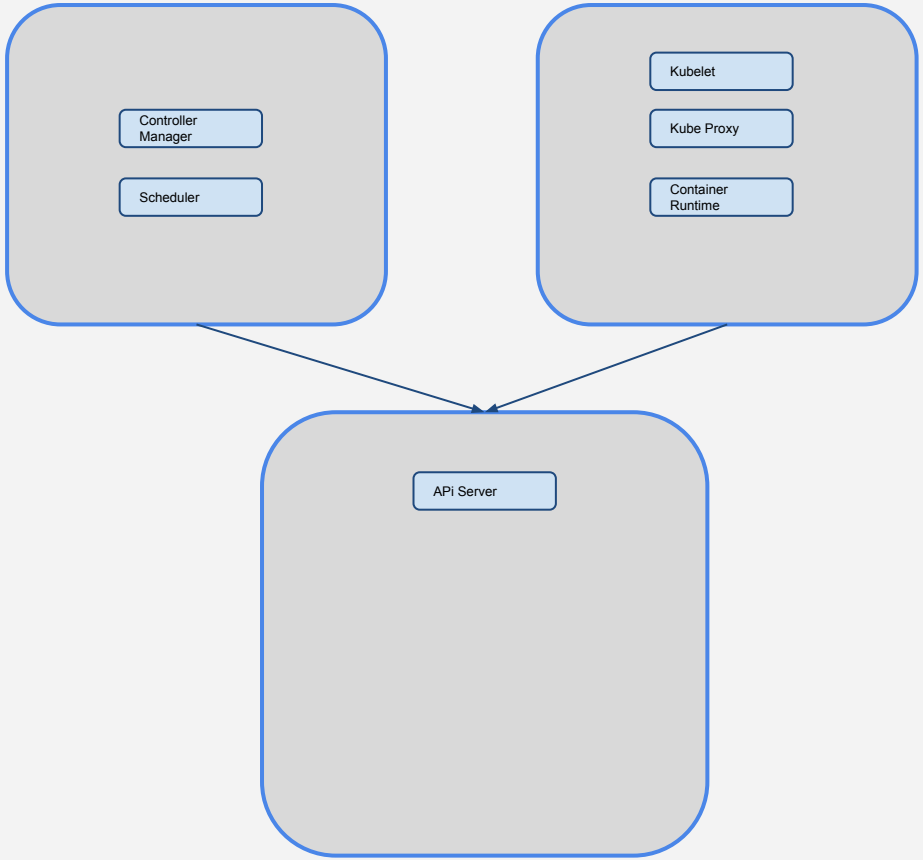
Kubernetes Architecture Demo



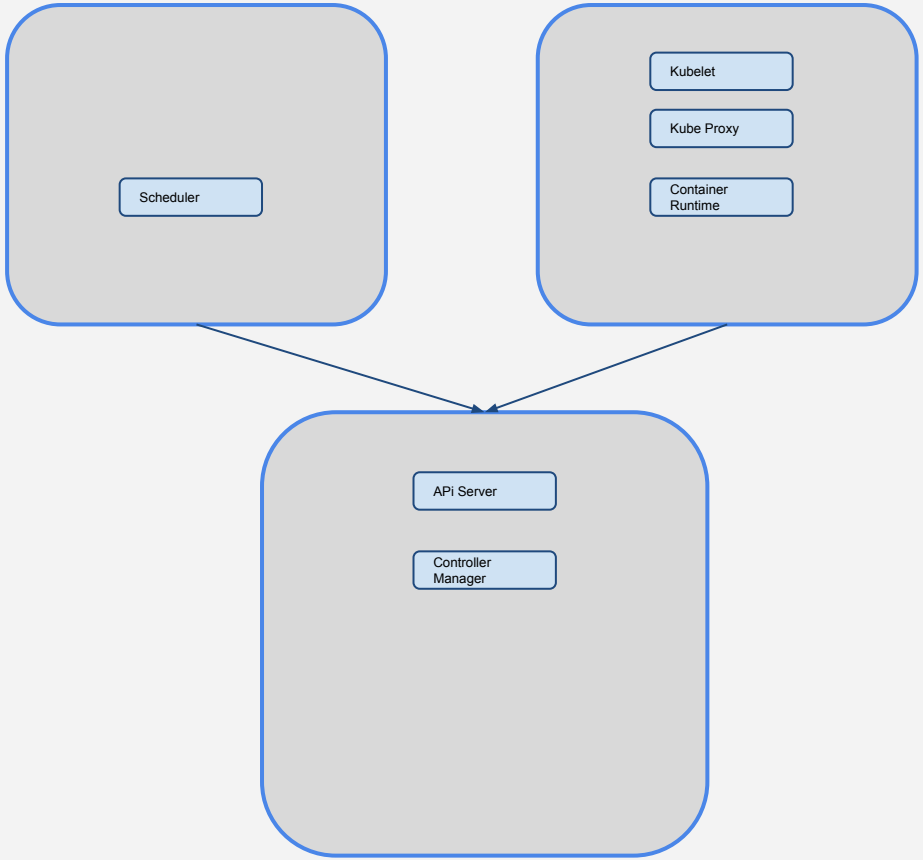
Kubernetes Architecture Demo



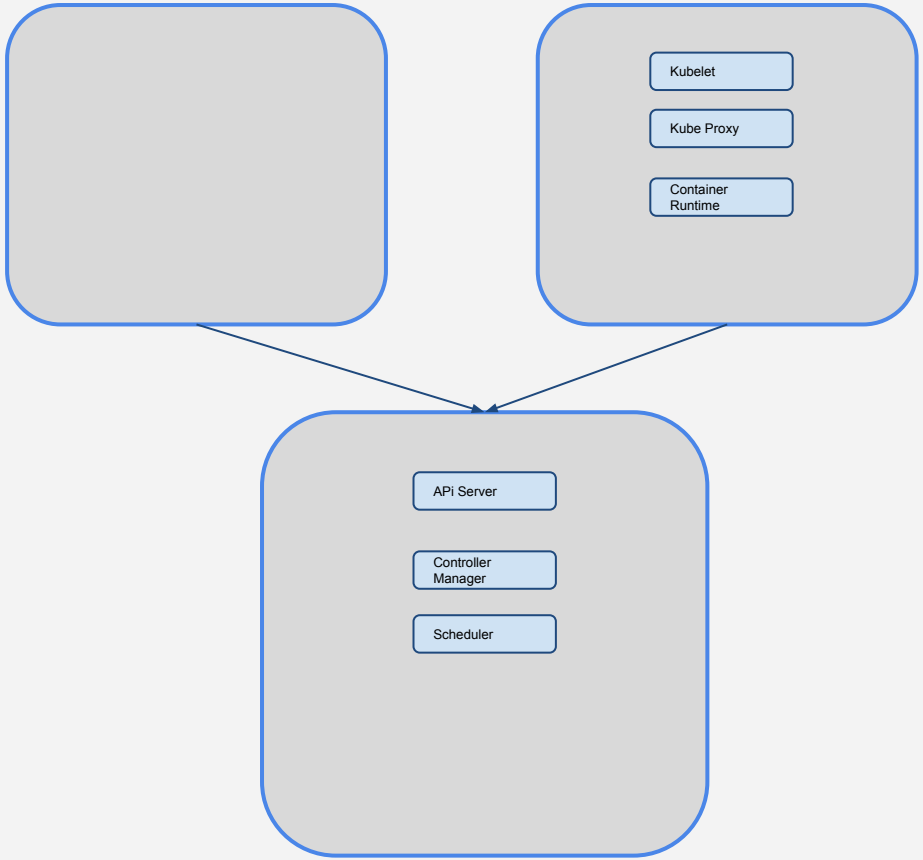
Kubernetes Architecture Demo



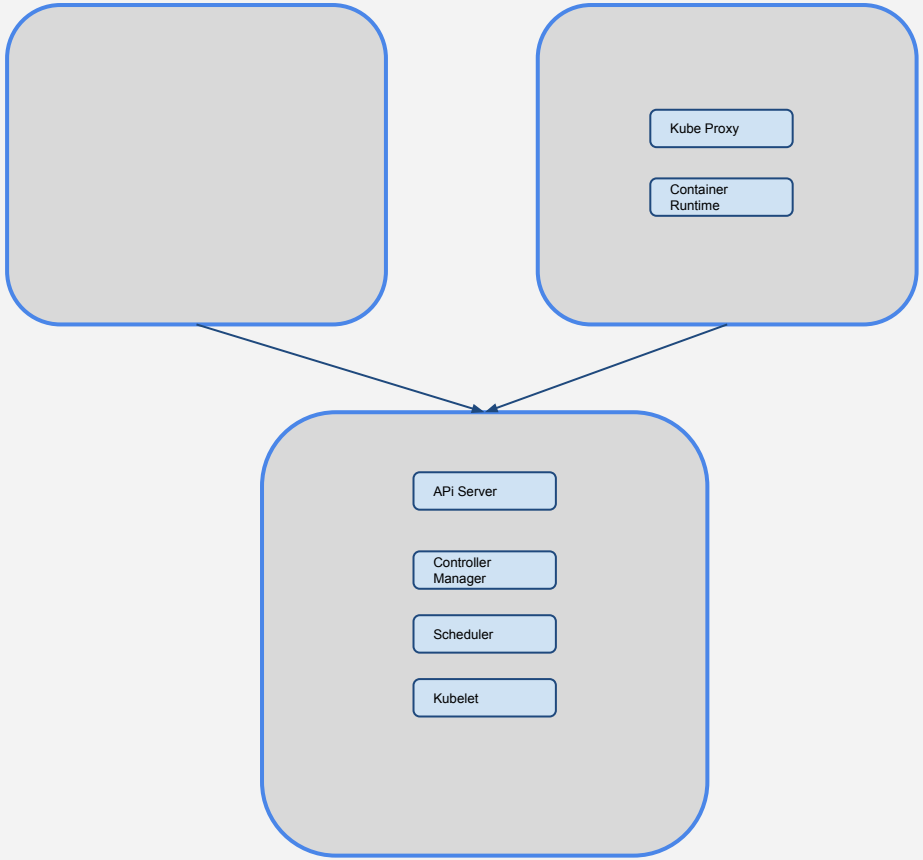
Kubernetes Architecture Demo



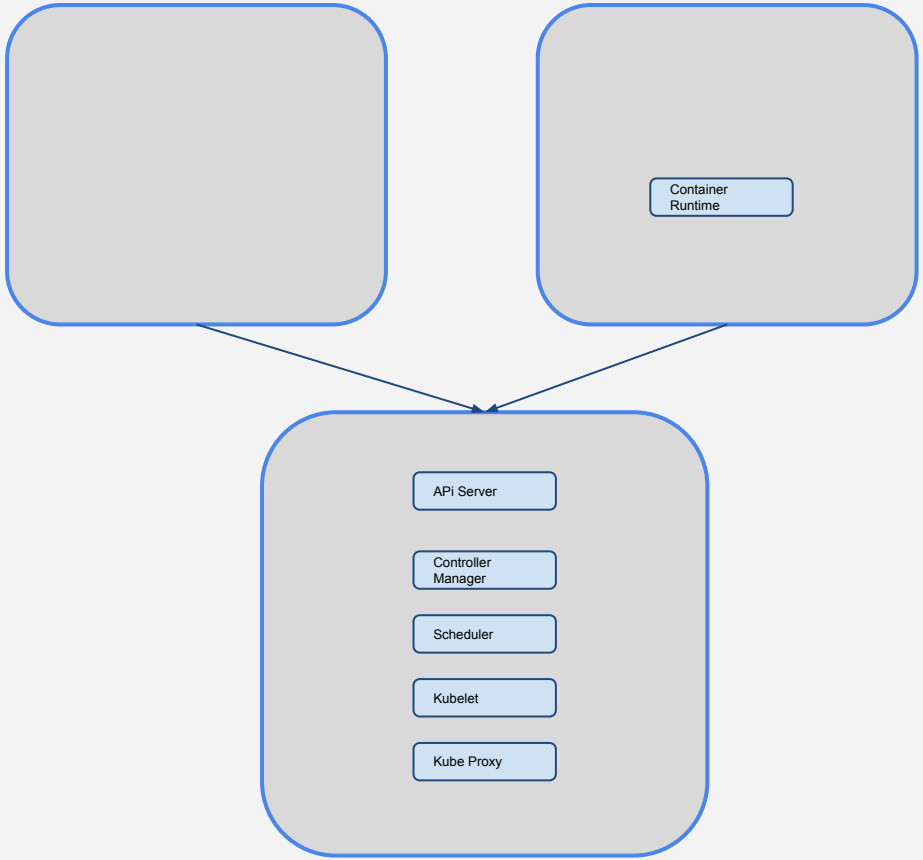
Kubernetes Architecture Demo



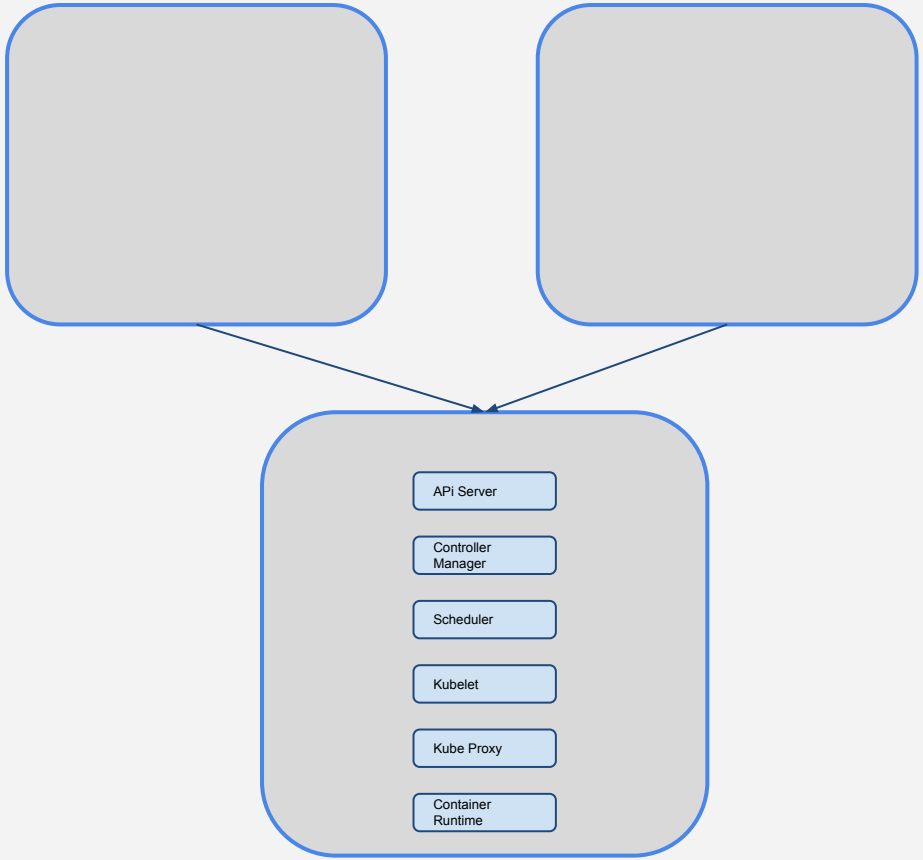
Kubernetes Architecture Demo



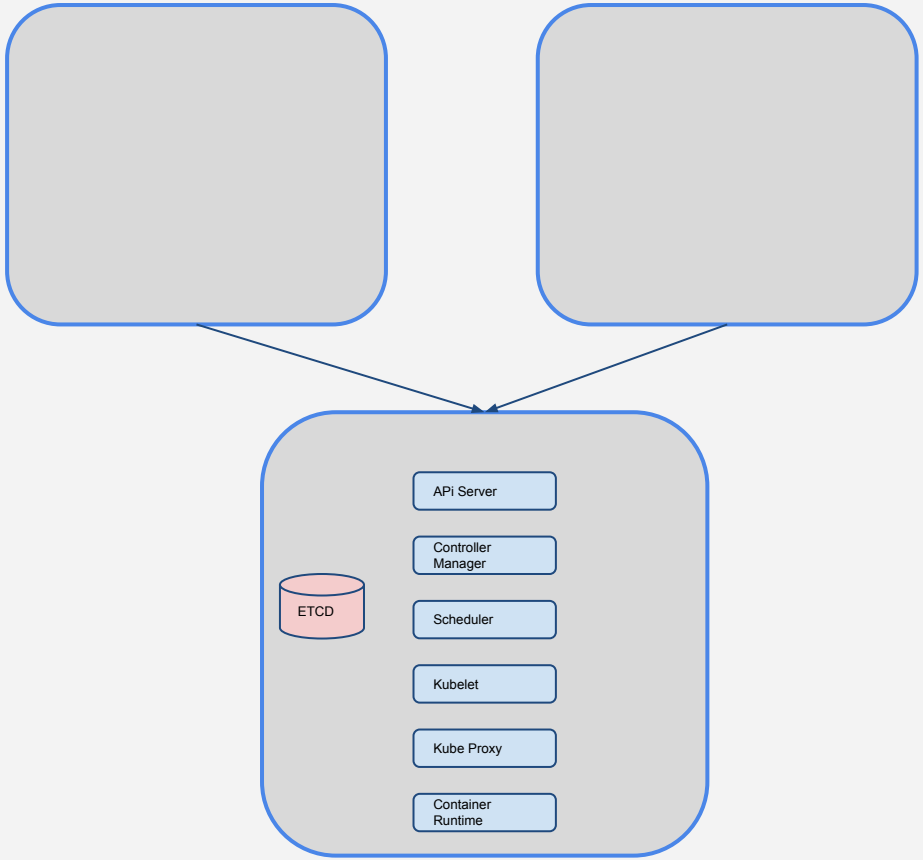
Kubernetes Architecture Demo



Kubernetes Architecture Demo



Kubernetes Architecture Demo



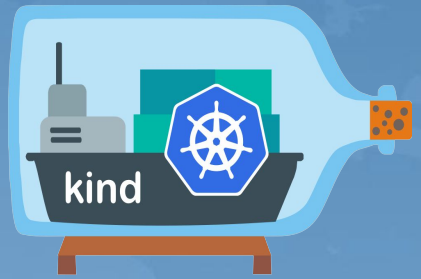
Module 3

Setup Lab Cluster

Setup Lab Cluster

In this module

Setup Single Node Cluster Using Minikube



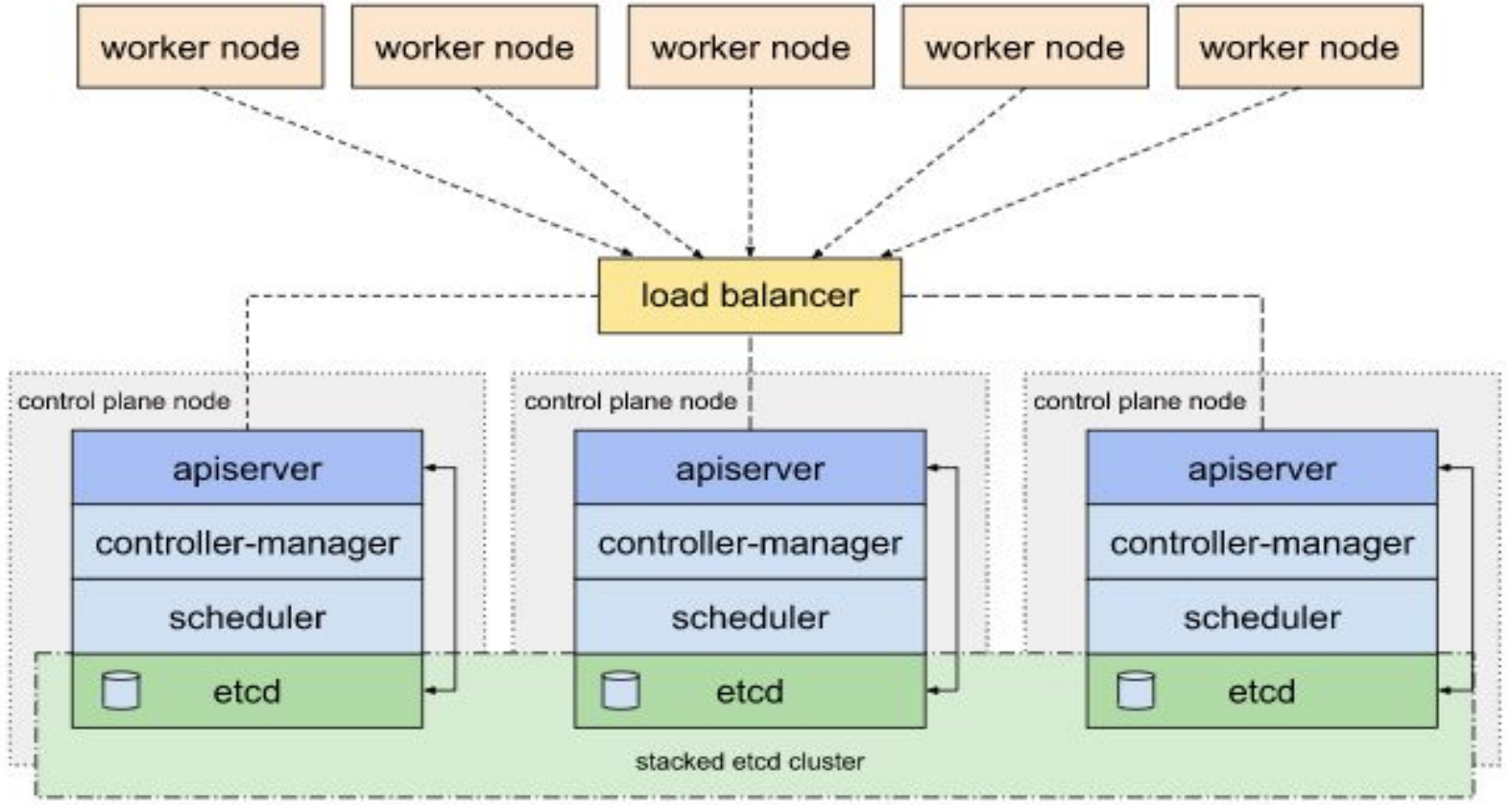
Setup Multi Node Cluster Using Kind

Setup Internal/ Private Docker Registry

Setup Single Node Cluster

In this video

- What is Minikube
- Install Hypervisor (VirtualBox)
- Install Minikube
- Setup Kubectl


worker node
worker node
worker node
worker node
worker node
load balancer
control plane node
control plane node
control plane node
apiserver
controller-manager
scheduler
etcd
apiserver
controller-manager
scheduler
etcd
apiserver
controller-manager
scheduler
etcd
stacked etcd cluster

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

Minikube

Master processes
- api-server
- controller-manager
- scheduler
- etcd

Worker processes
- kube-proxy
- kubelet
- container runtime


DEMO
Ahmed ElBakry


THANK YOU
Ahmed ElBakry

Kubernetes From Scratch - Hands-on

09 - Setup Multi Node Cluster (Kind)

Ahmed ElBakry

Setup Multi Node Cluster

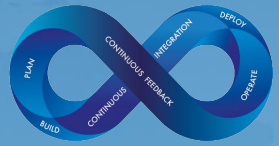
In this video

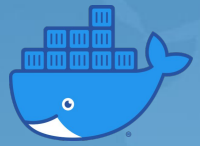
- ❑ What is Kind
- ❑ Install Docker
- ❑ Install Kind
- ❑ Explore The Cluster
- ❑ Check The kubeconfig File

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

DevOps-Kubernetes-Docker بالعربي



Ahmed ElBakry


DEMO
Ahmed ElBakry

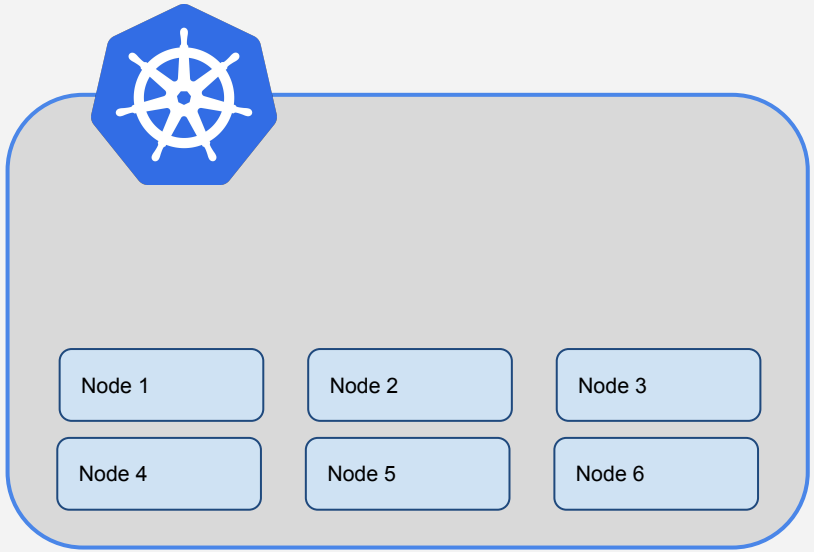

THANK YOU
Ahmed ElBakry

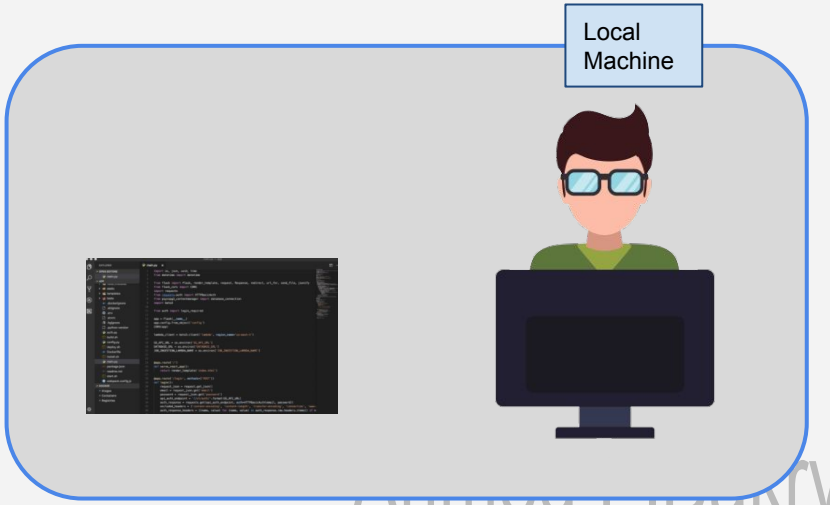
Kubernetes From Scratch - Hands-on

10 - Setup Internal Docker Registry

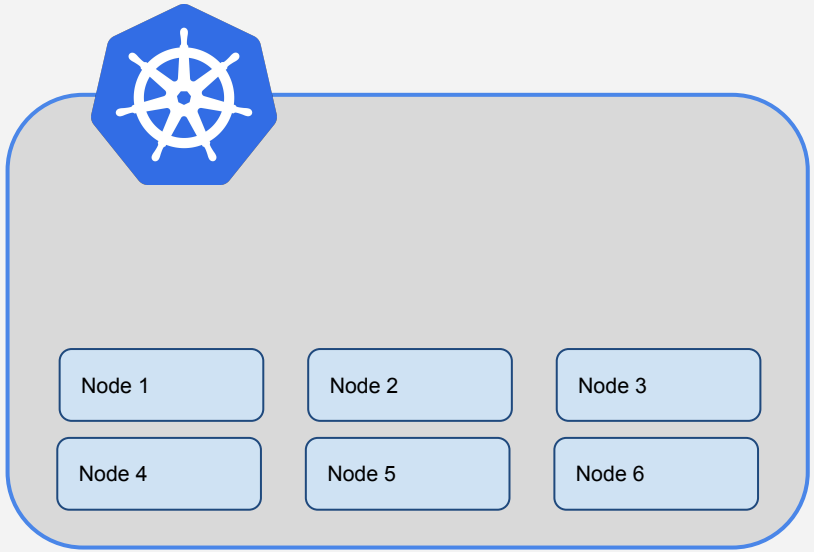
Ahmed ElBakry

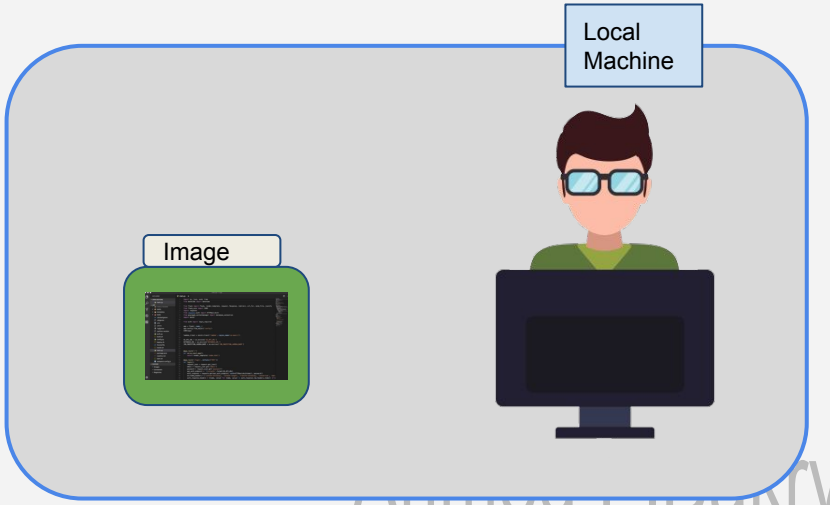
Setup Internal Docker Registry



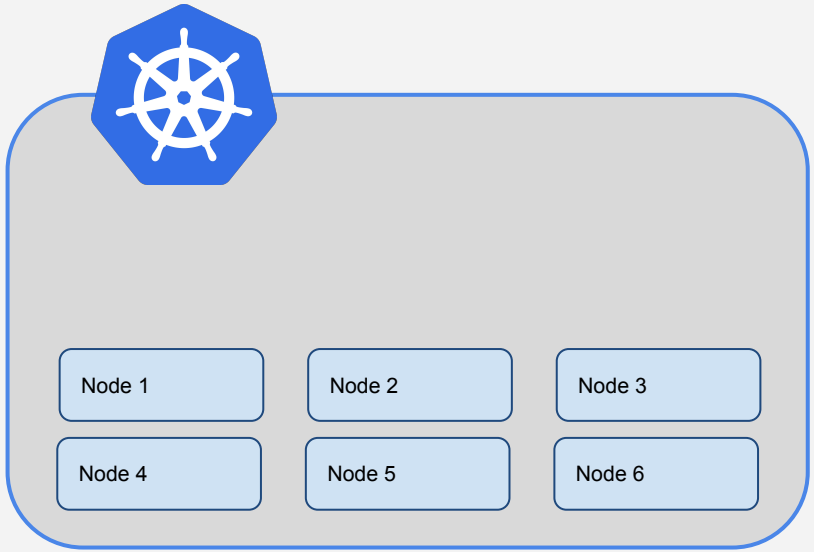


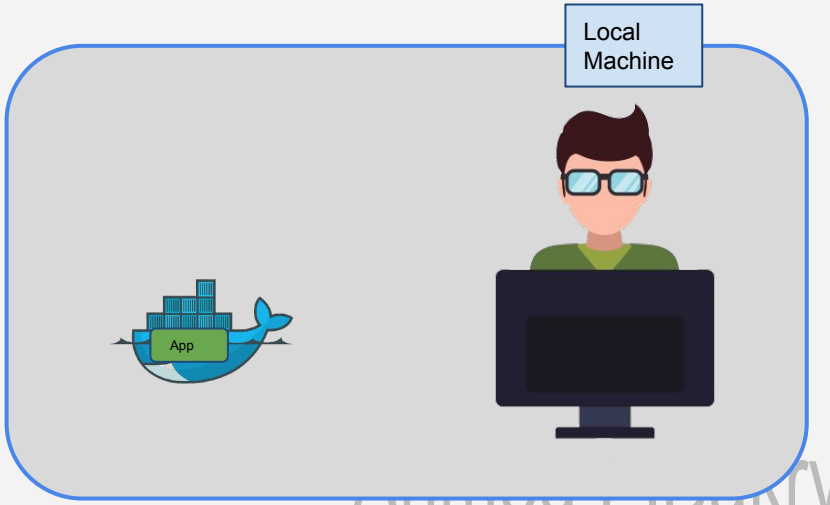
Setup Internal Docker Registry



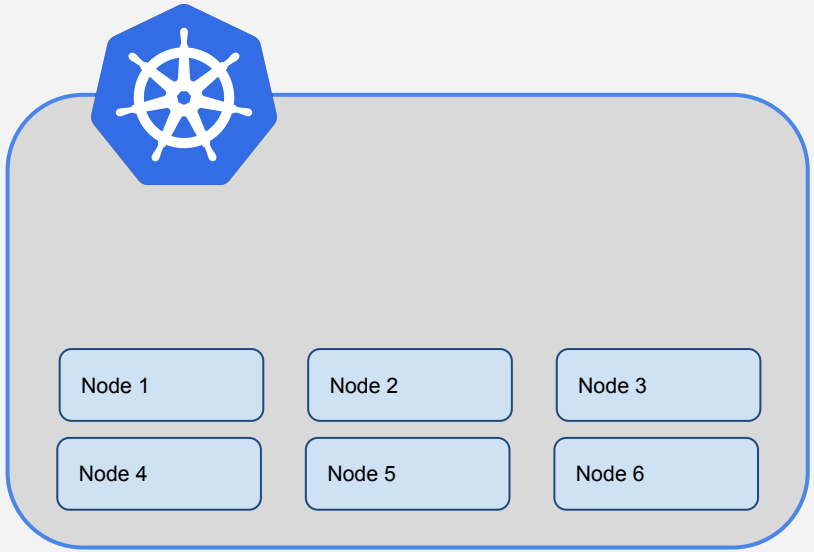


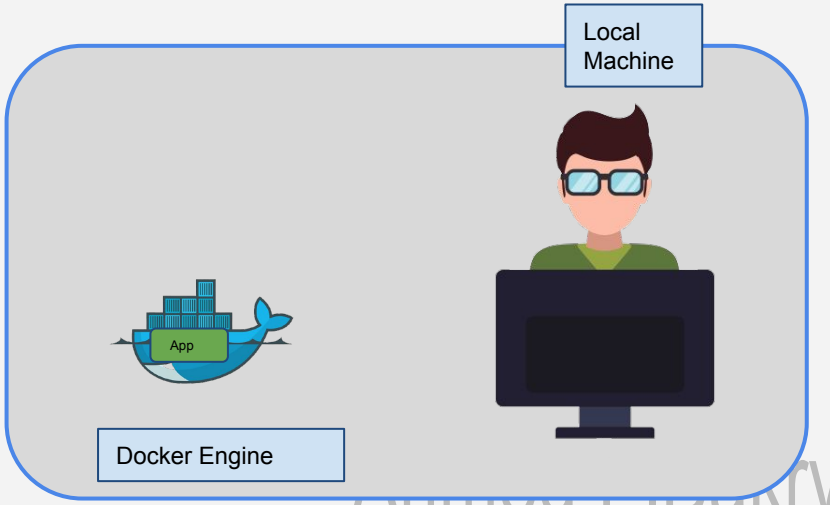
Setup Internal Docker Registry





Setup Internal Docker Registry





Setup Internal Docker Registry

Node 1

Node 2

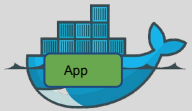
Node 3

Node 4

Node 5

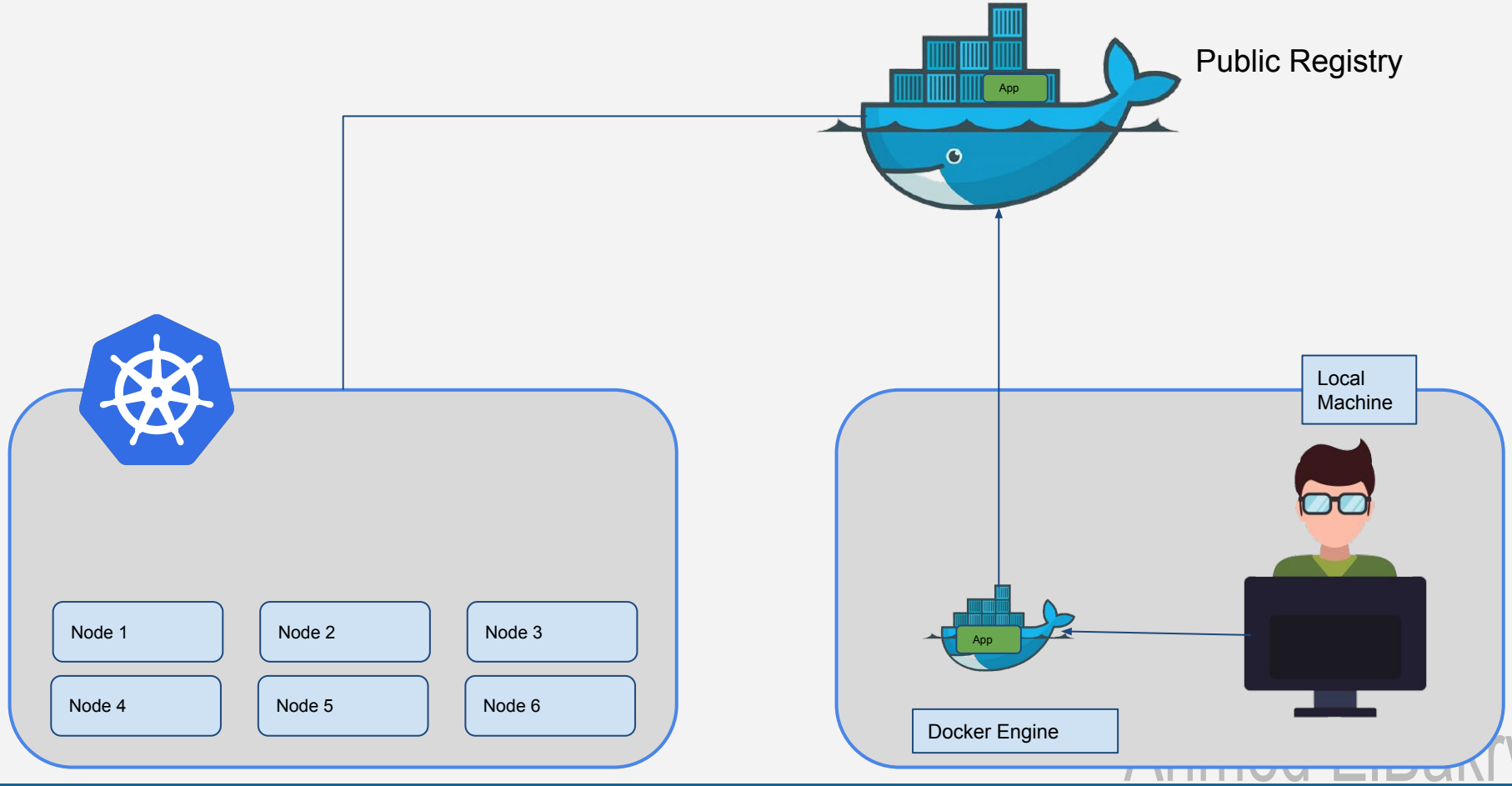
Node 6

Local Machine

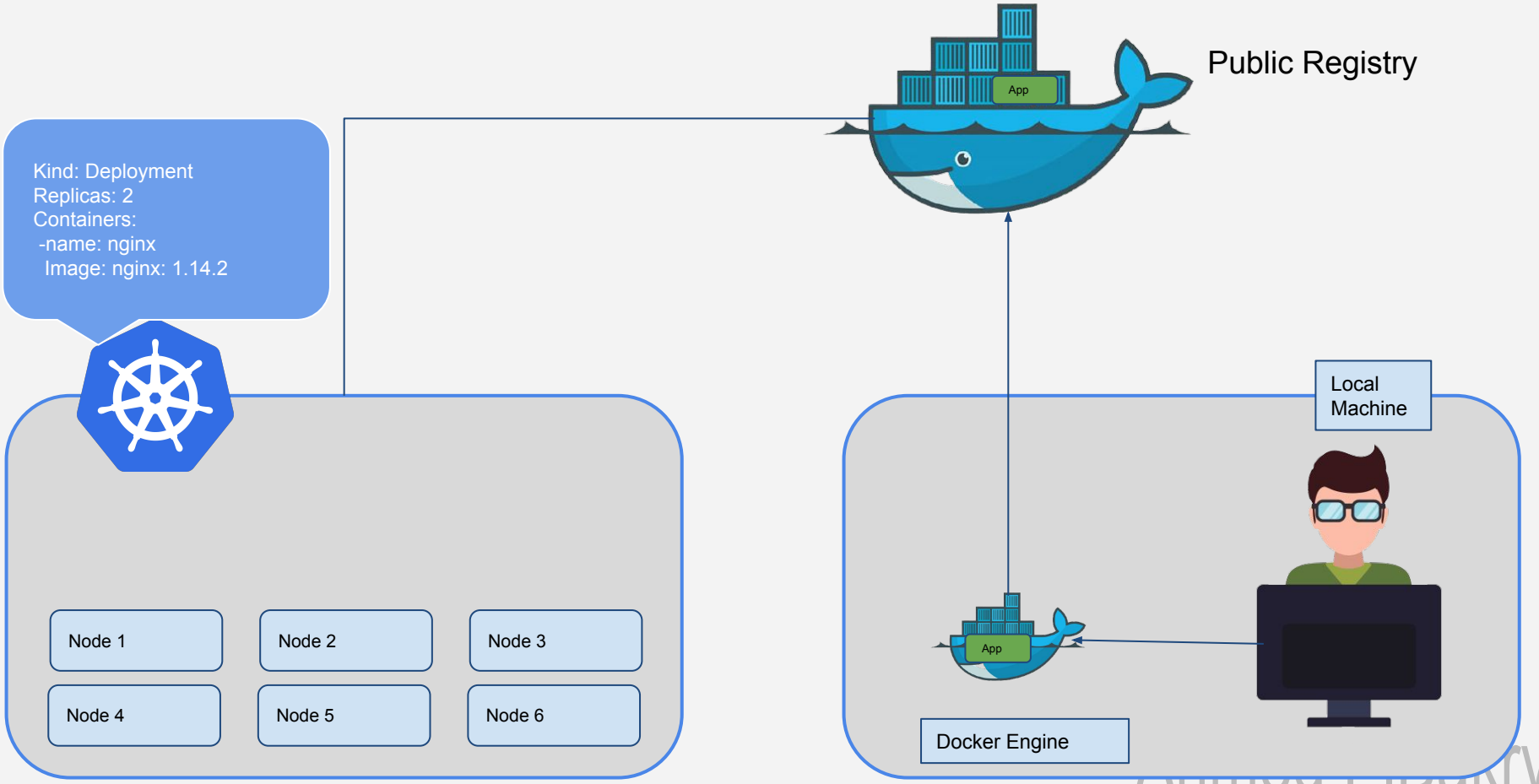


Docker Engine

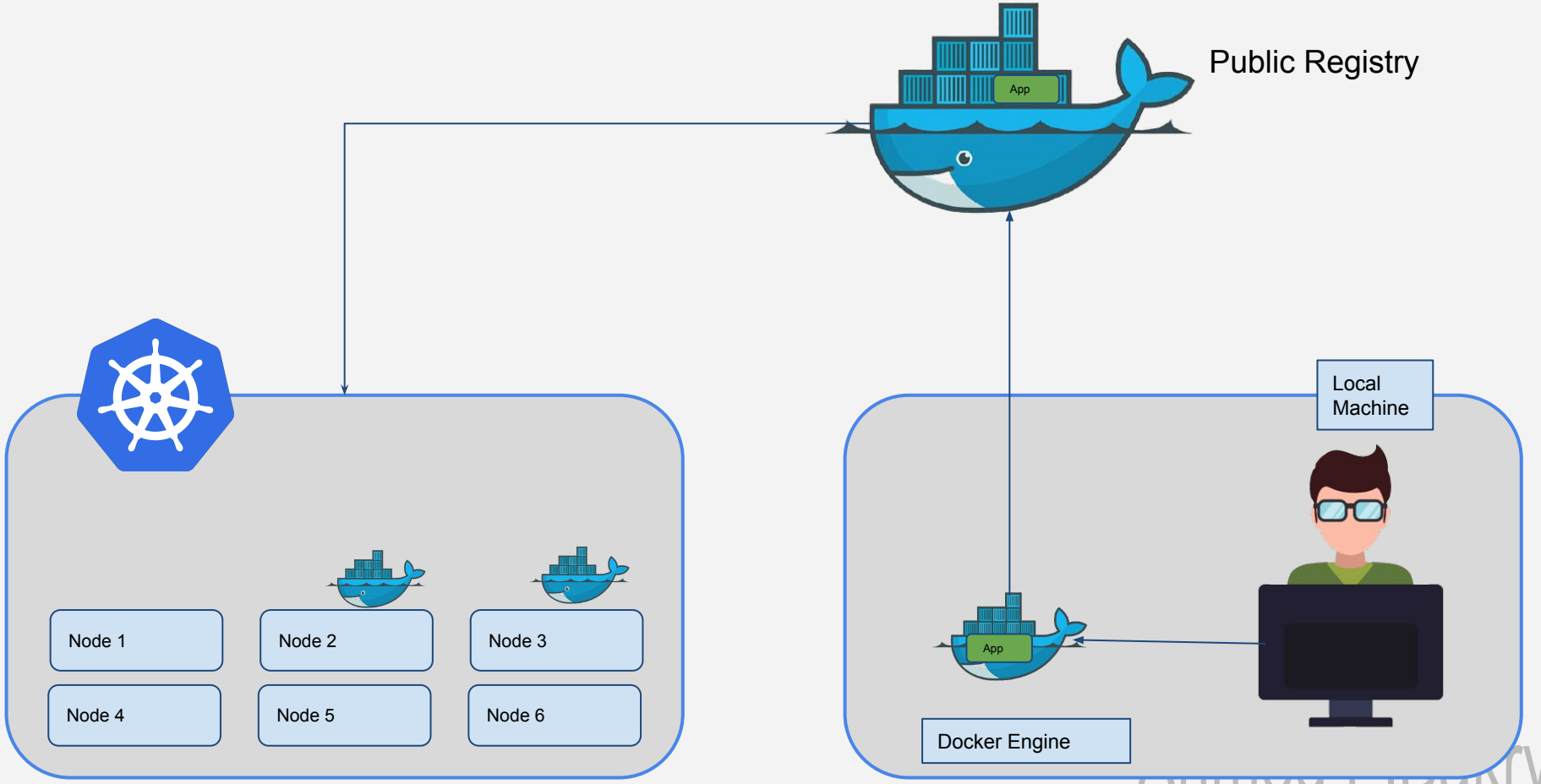
Setup Internal Docker Registry



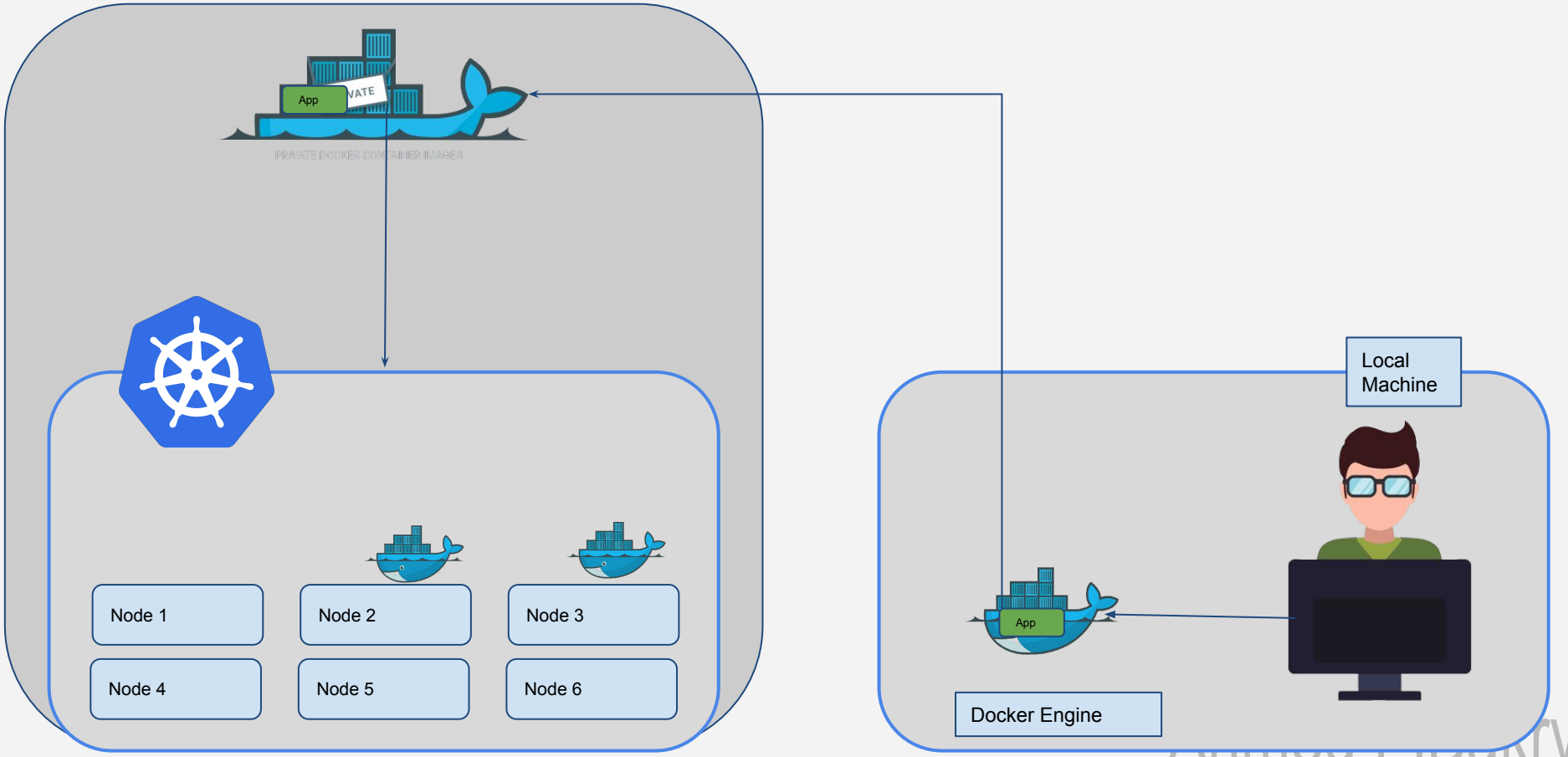
Setup Internal Docker Registry



Setup Internal Docker Registry



Setup Internal Docker Registry

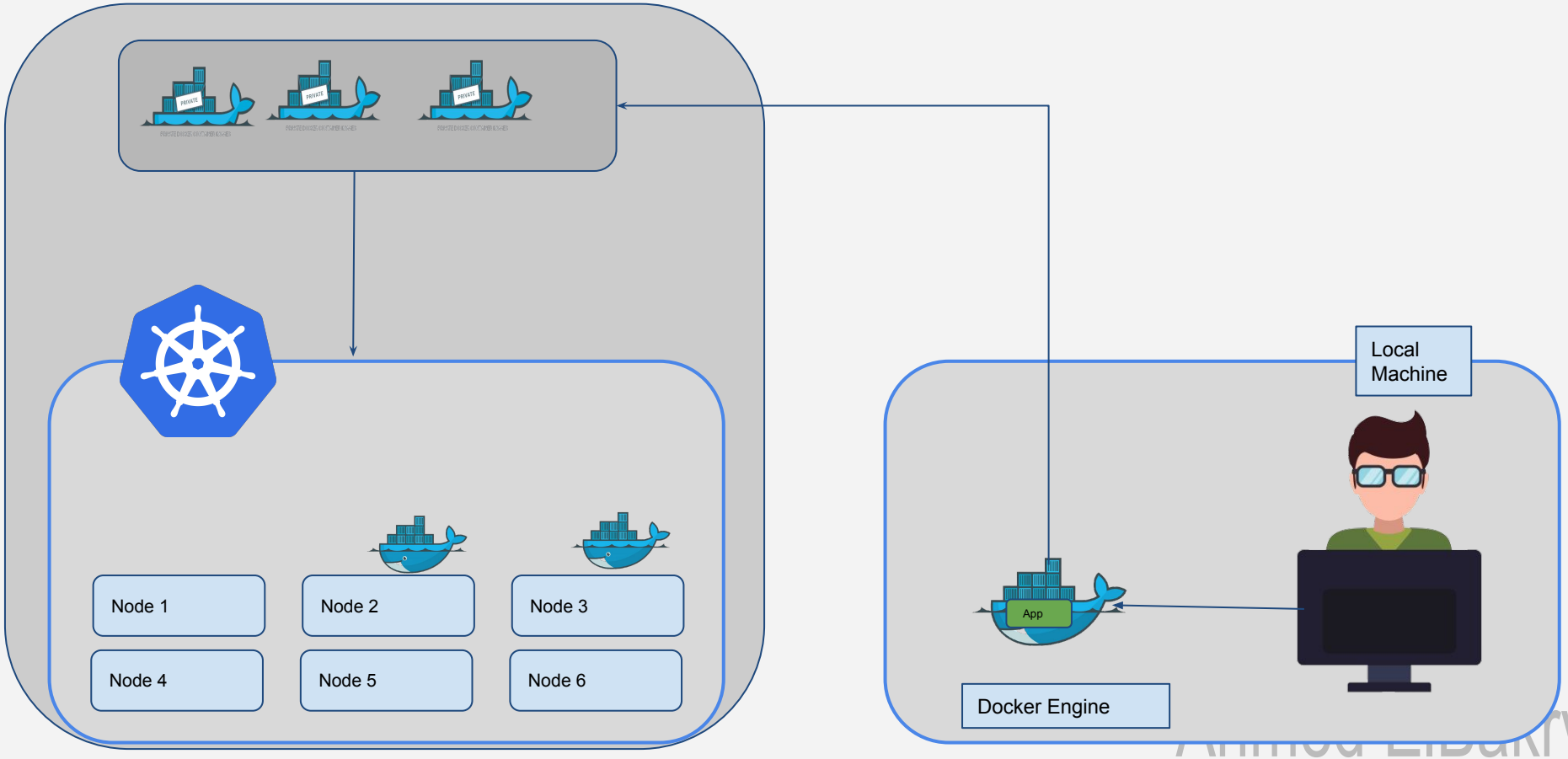


Setup Internal Docker Registry

Demo

- ❏ Run Local Application
- ❏ Containerize The Application
- ❏ Push The Application to Public Docker Registry
- ❏ Deploy The Application to Kubernetes
- ❏ Deploy Internal/Private Docker Registry
- ❏ Push The Application to Internal Registry
- ❏ Deploy The Application to Kubernetes

Setup Internal Docker Registry

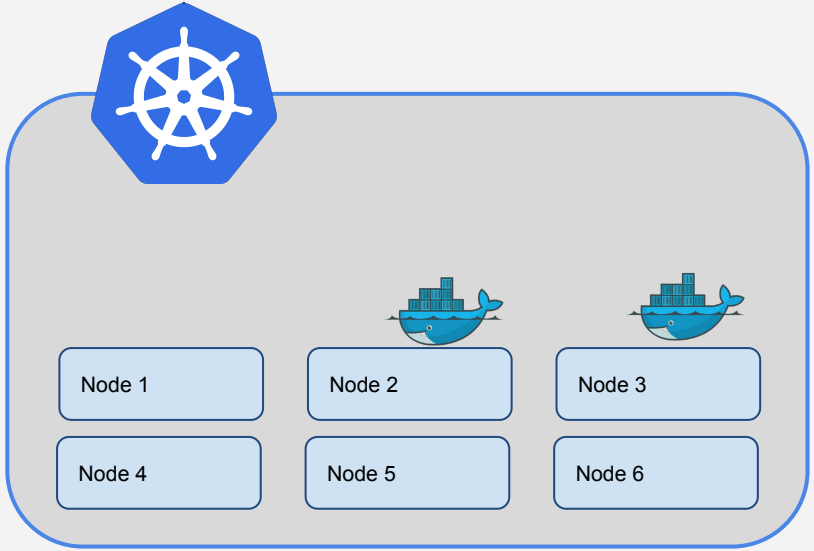


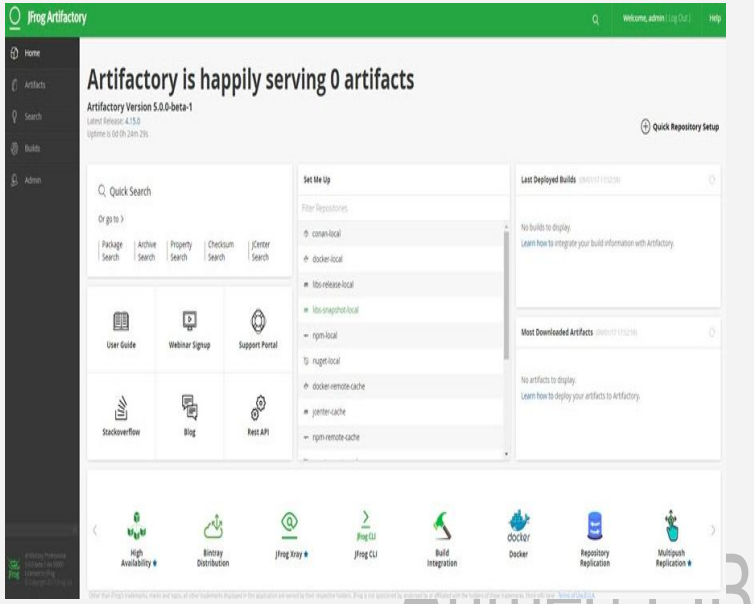
Setup Internal Docker Registry

Artifacts Server

Setup Internal Docker Registry

Artifacts Server






THANK YOU
Ahmed ElBakry

Kubernetes From Scratch - Hands-on

12 - Namespaces

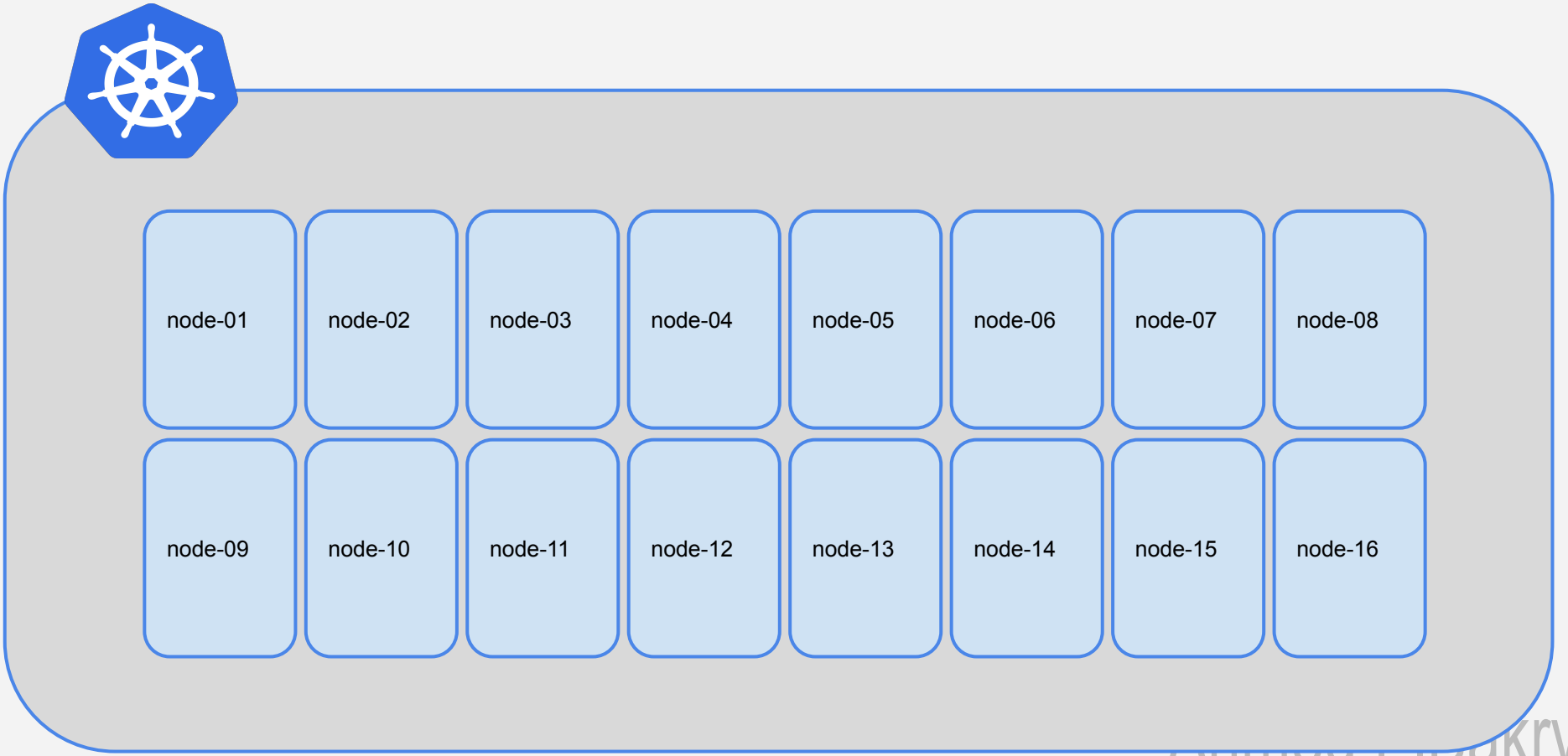
Ahmed ElBakry

Module 4

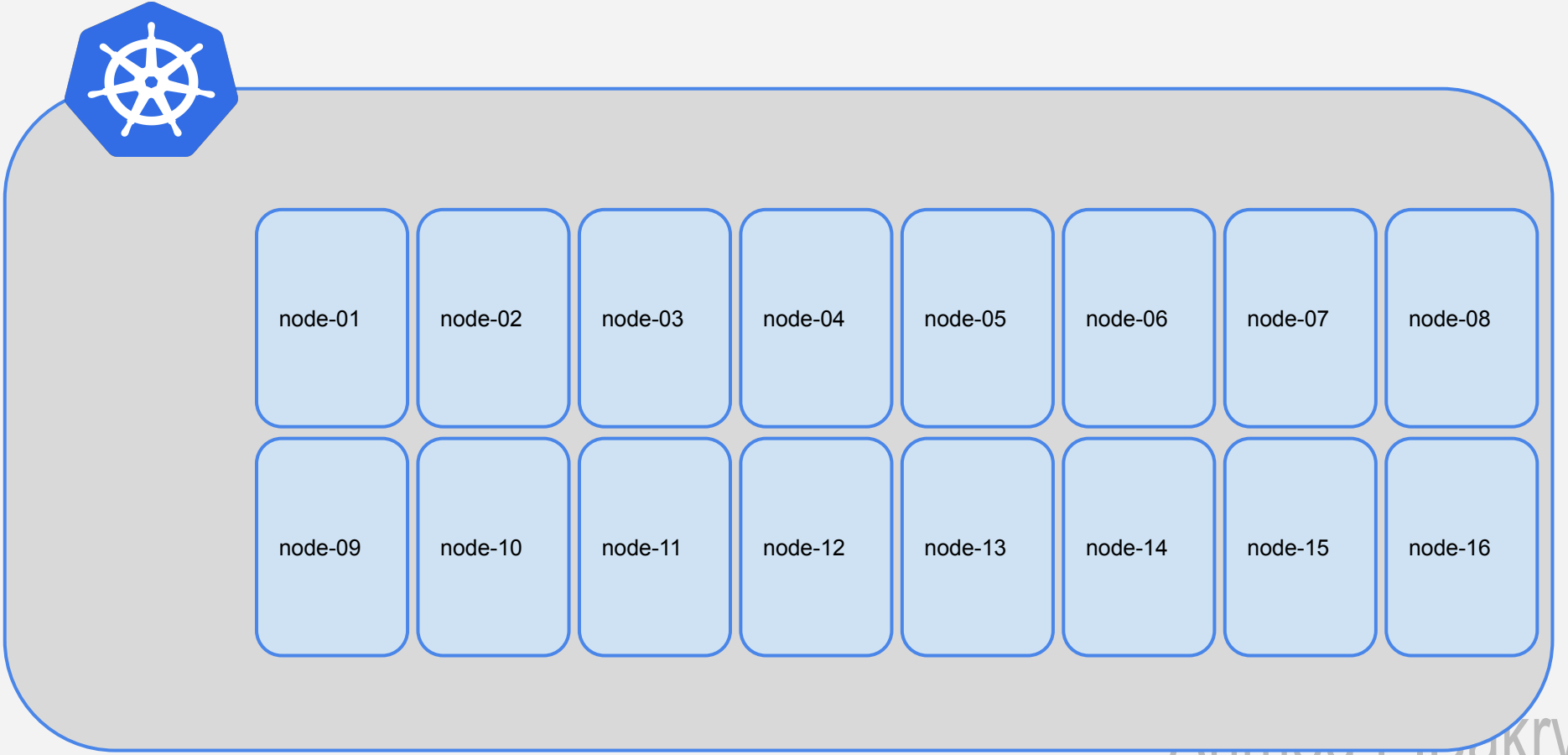
Basic Concepts

Namespaces

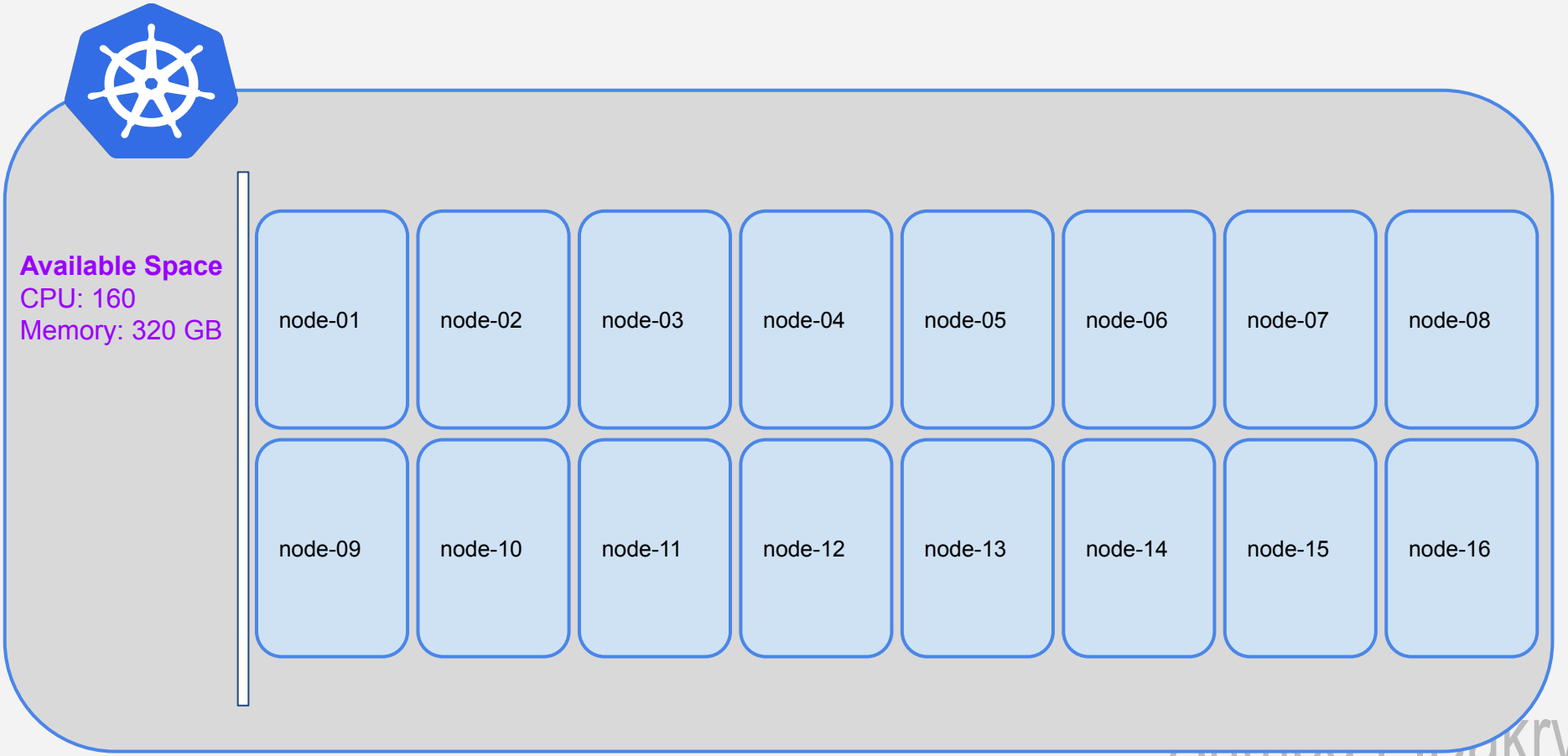
Namespaces



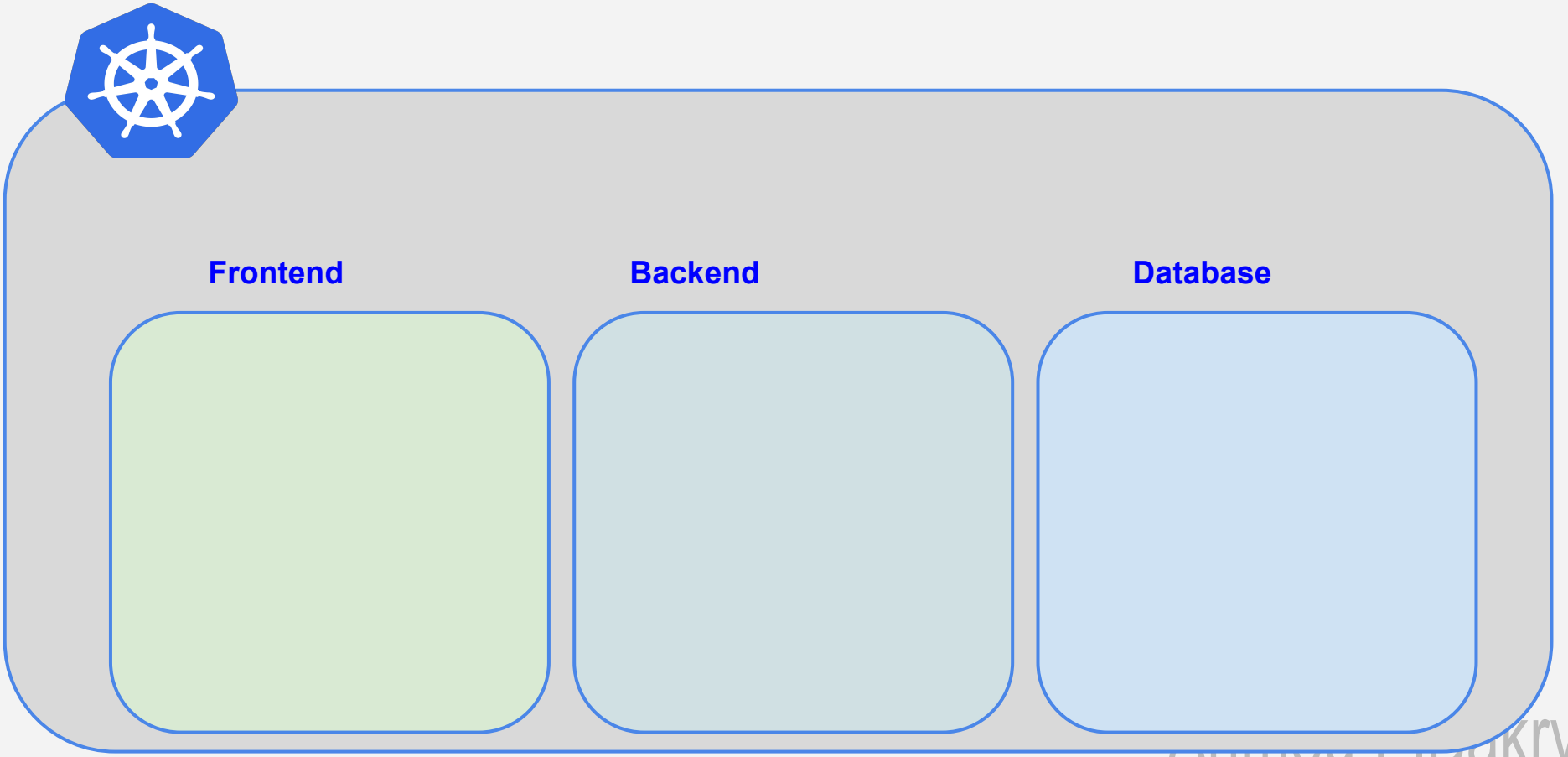
Namespaces



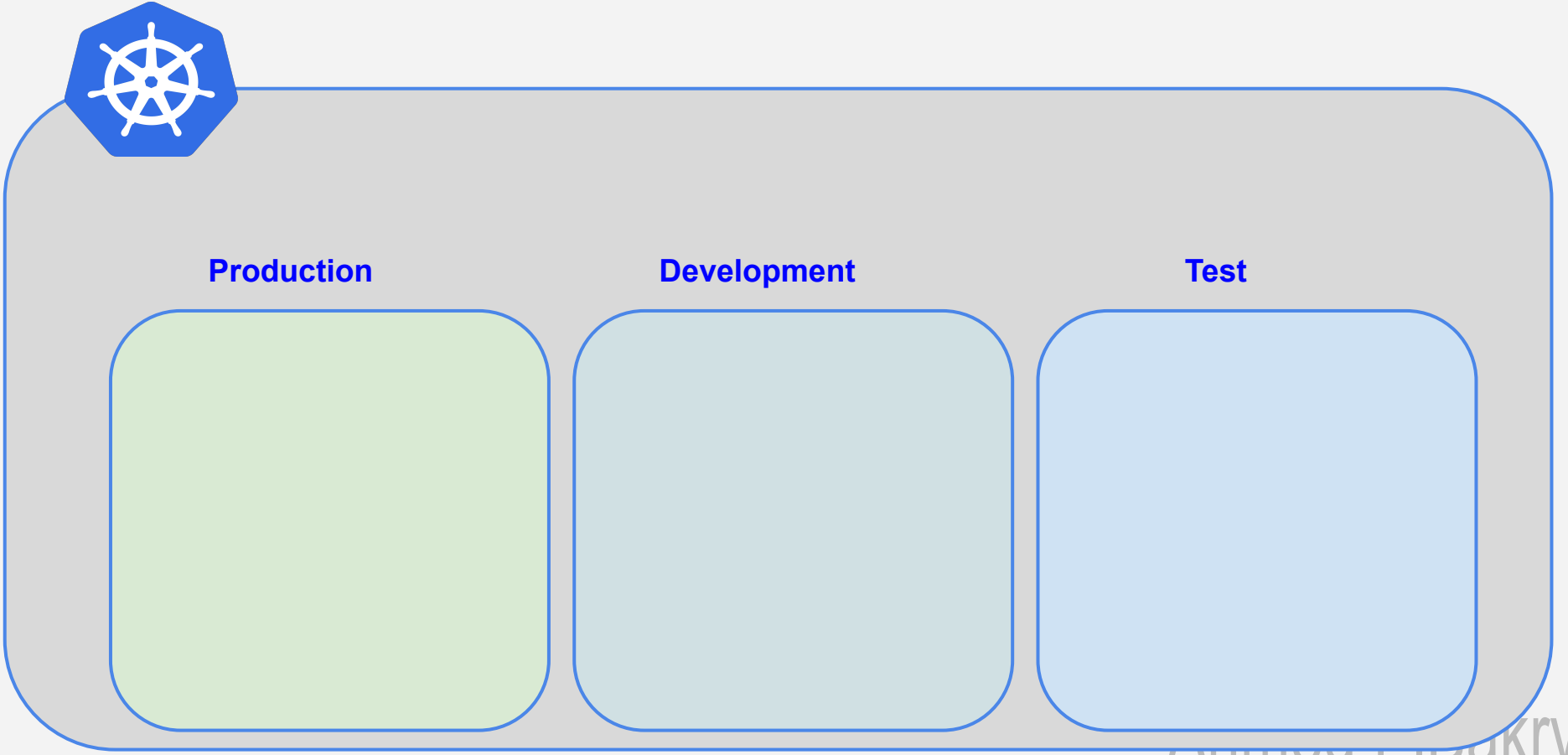
Namespaces



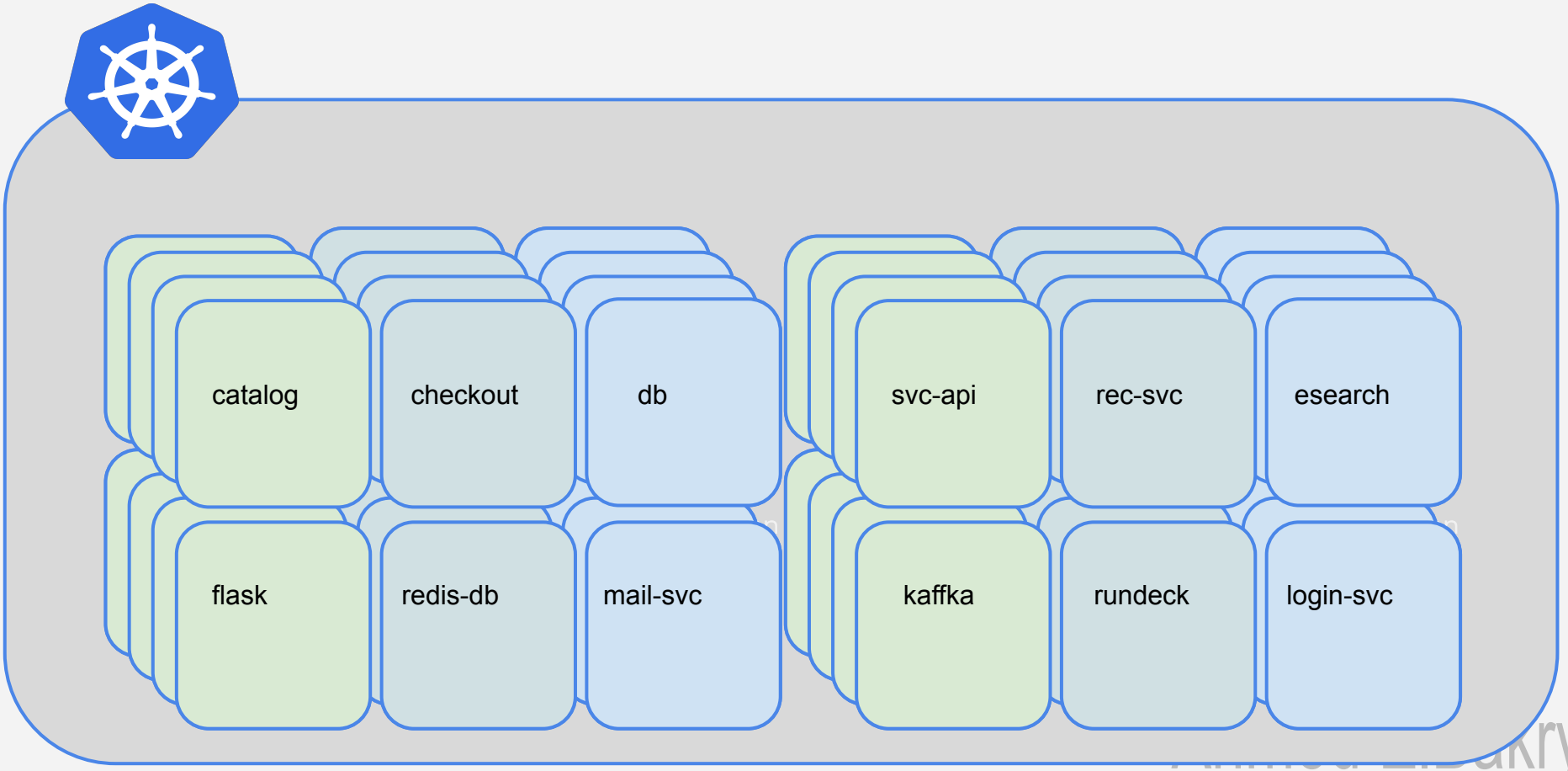
Namespaces



Namespaces



Namespaces


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

Namespaces

Demo

- ❏ List namespaces
- ❏ Create namespaces
- ❏ Switch namespaces.
- ❏ Create objects in different namespaces
- ❏ Namespaced resource vs non-namespaced resource
- ❏ Access services in different namespaces.
- ❏ Limit and isolate access with network policy


THANK YOU
Ahmed ElBakry

Kubernetes From Scratch - Hands-on

13 - Resources, Resource Quota, and LimitRanges

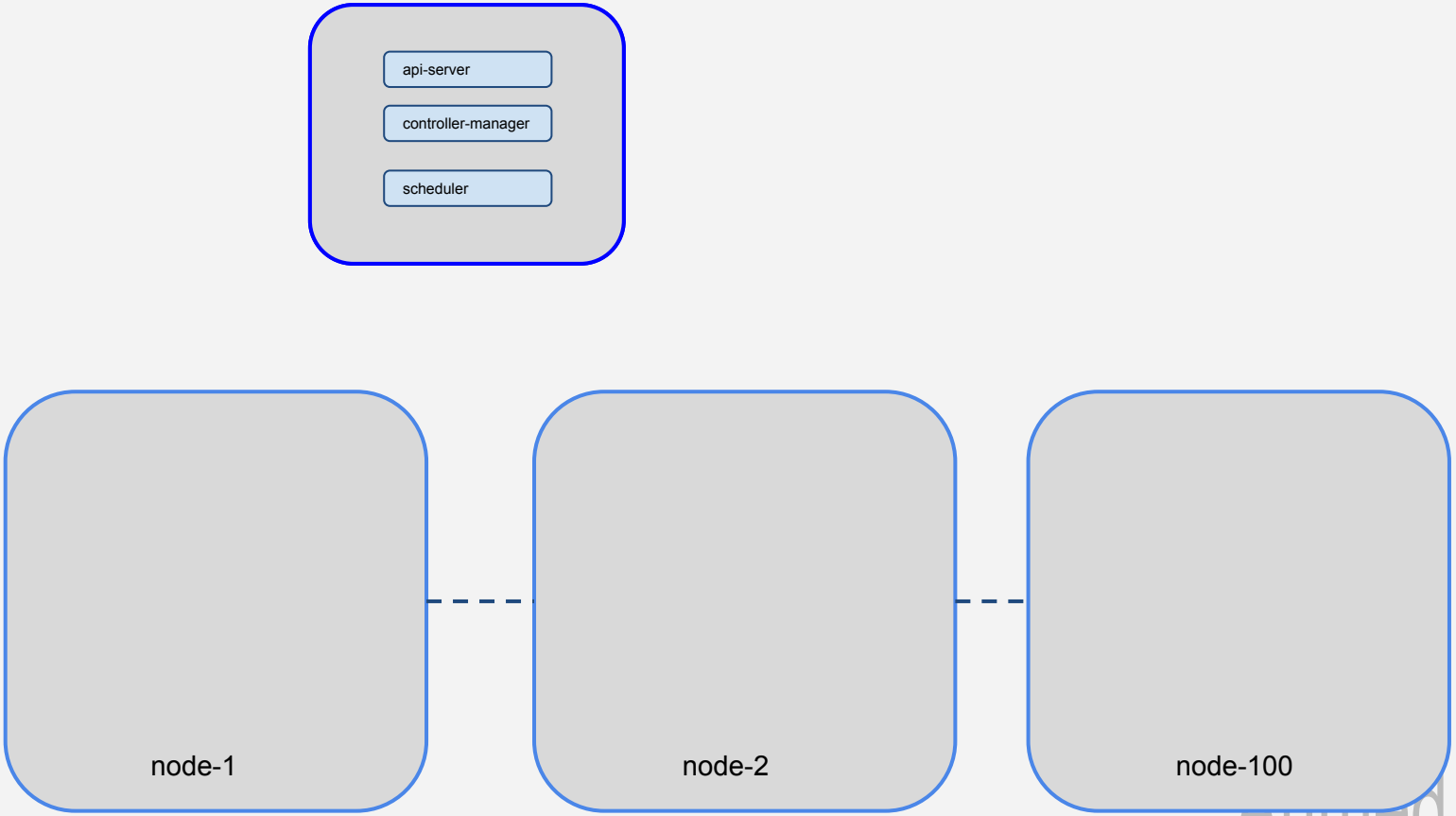
Ahmed ElBakry

Resource Quota

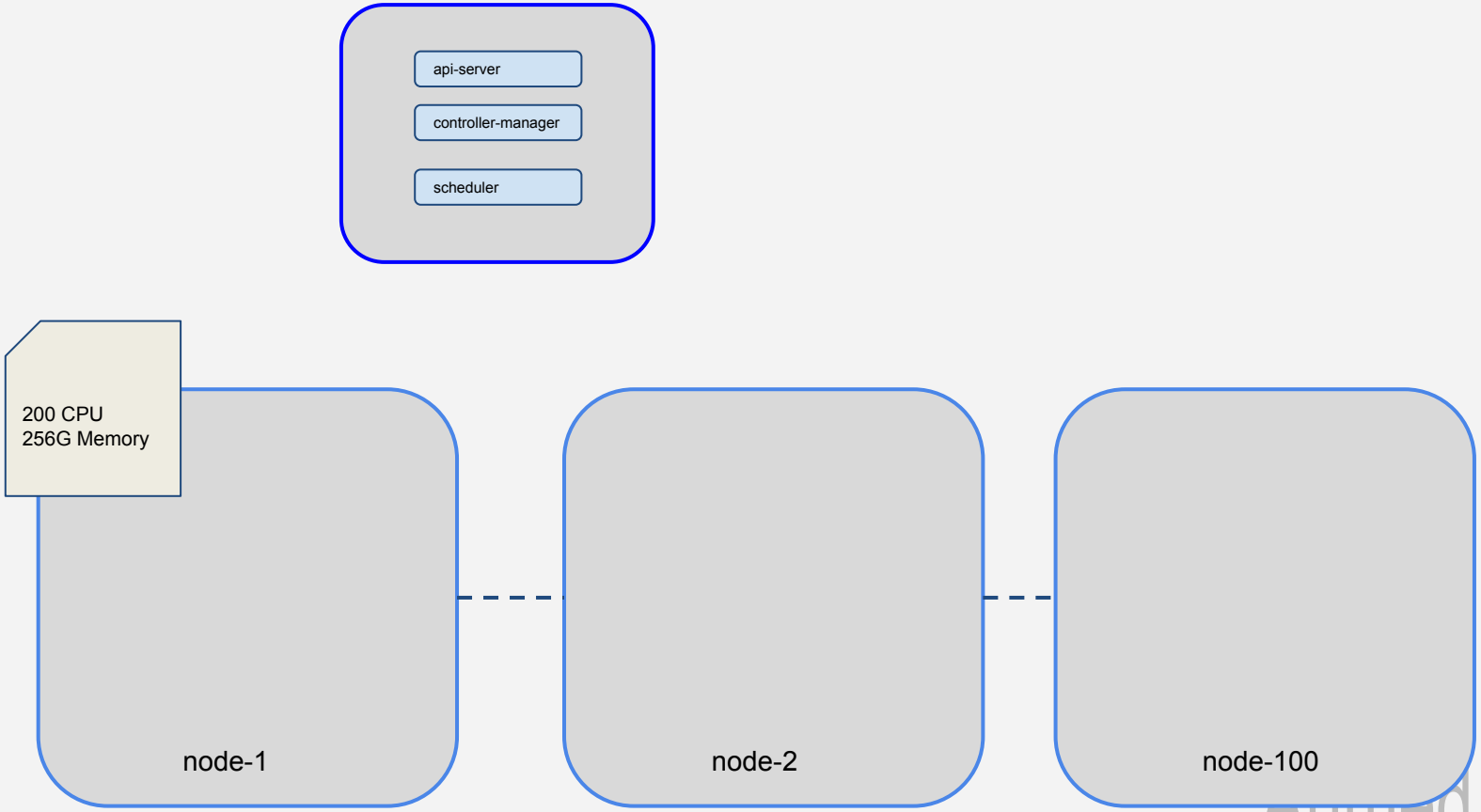
In this video

- ❑ Resources
- ❑ Limit Range
- ❑ Resource Quota
- ❑ Demo

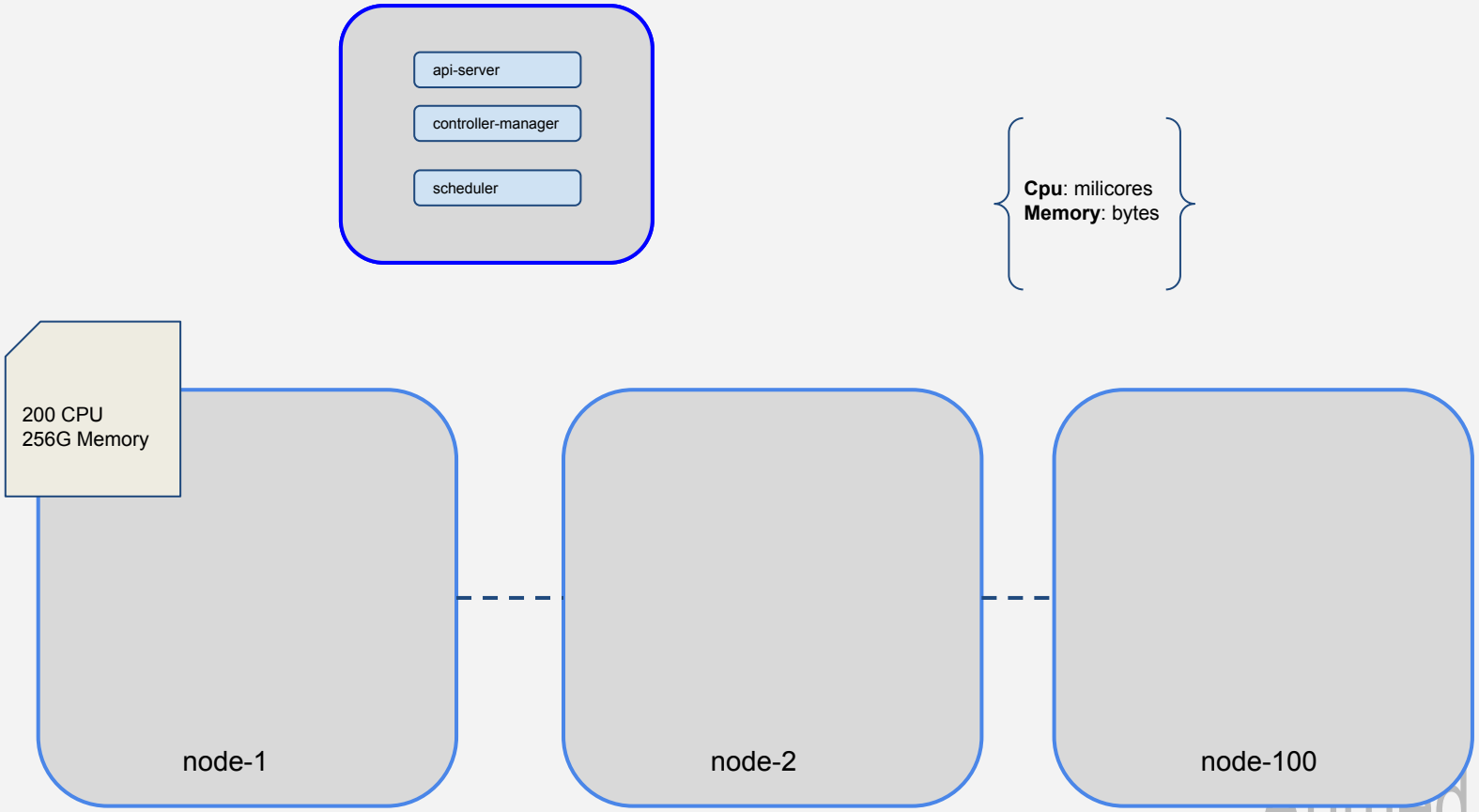
Resources



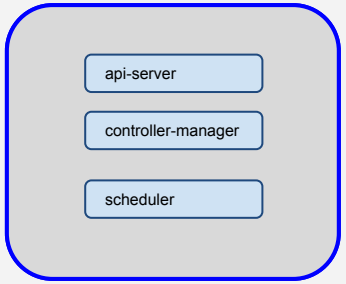
Resources



Resources

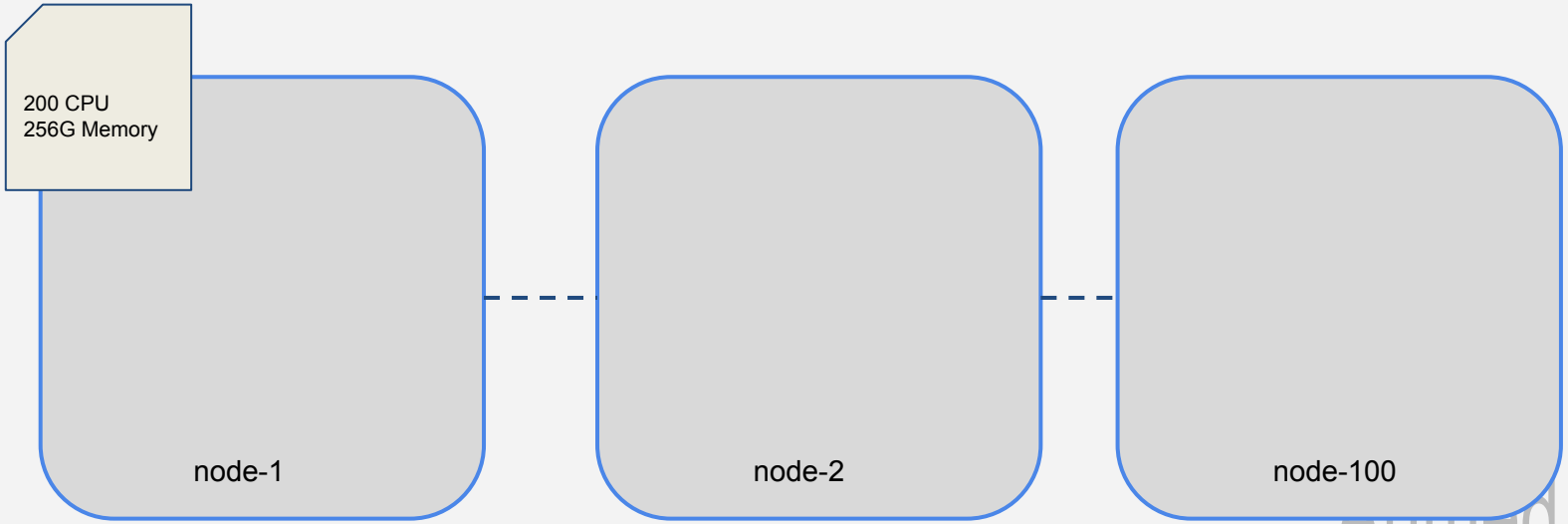


Resources

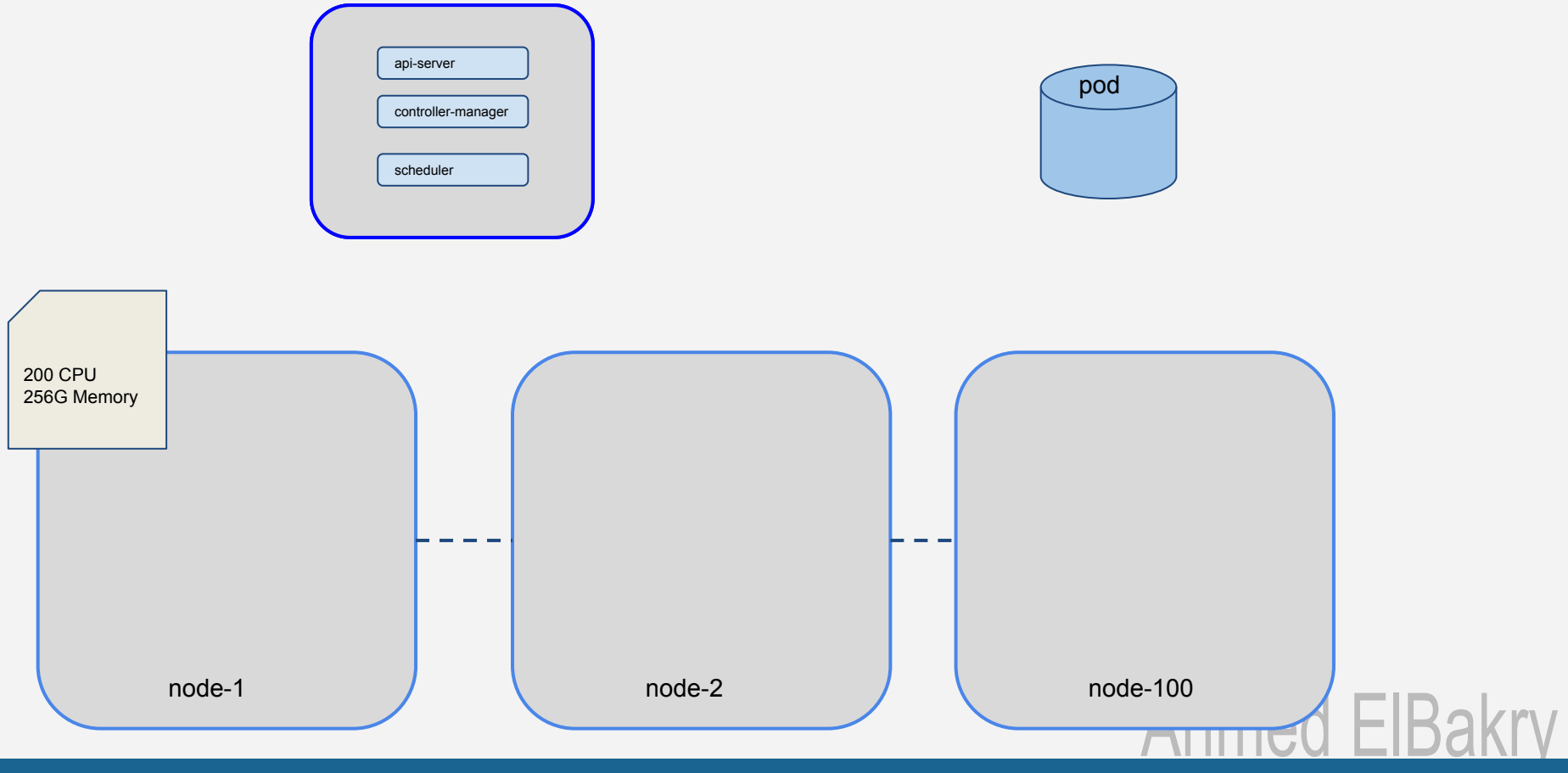


Cpu: milicores
Memory: bytes

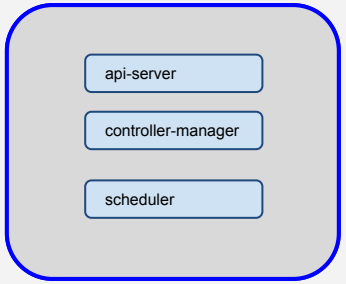
1, 100m, 200m, 2000m
50M, 2Gib

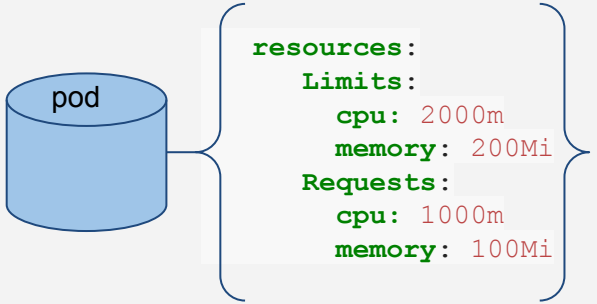


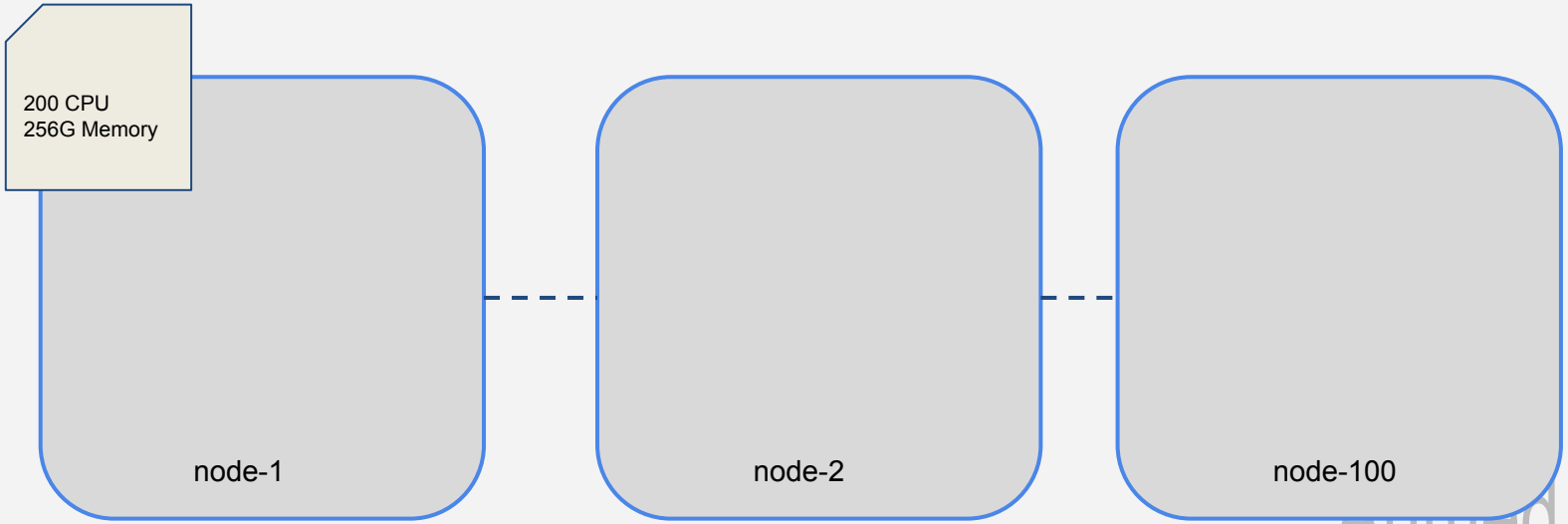
Resources



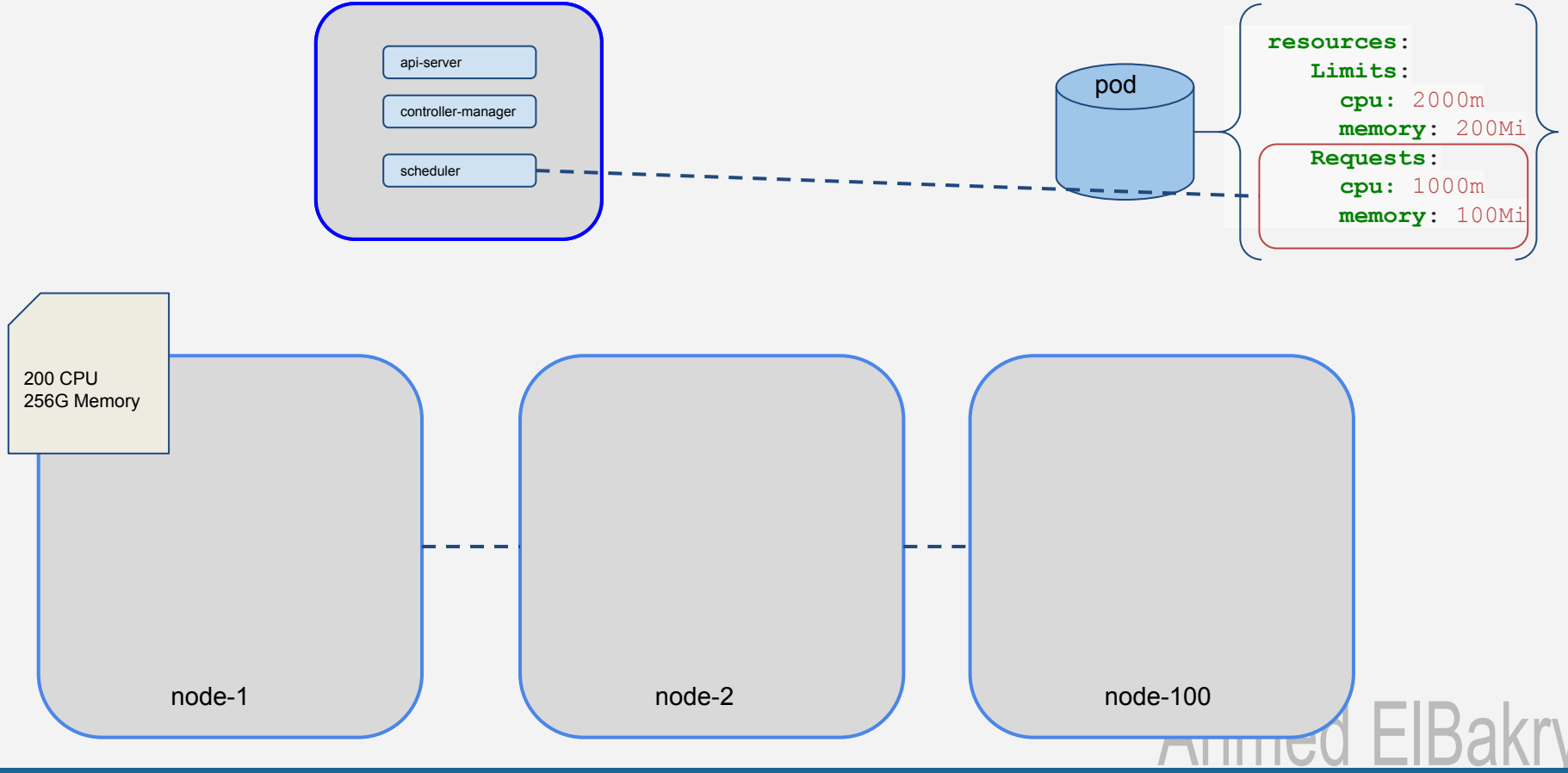
Resources







Resources

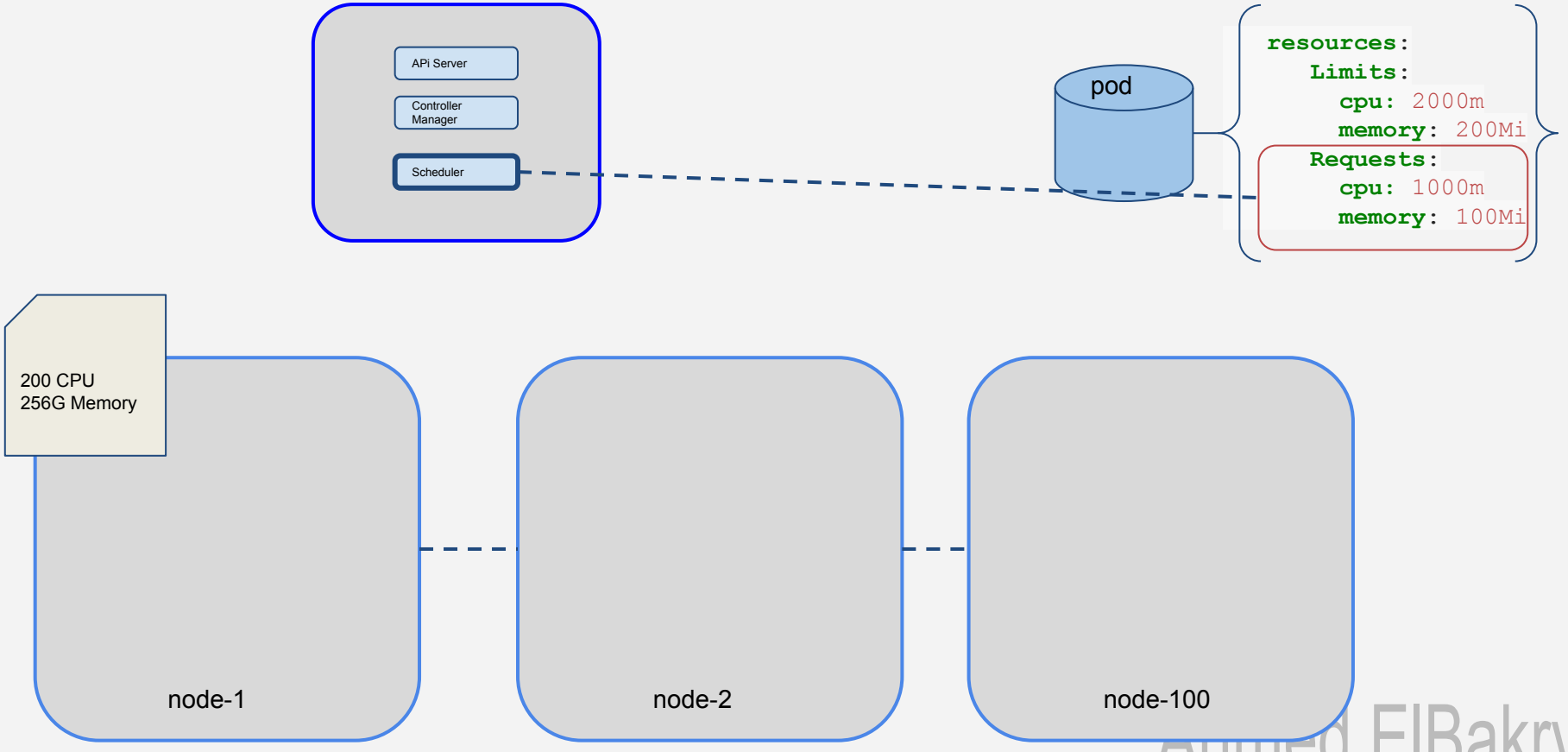


Quality of Service for Pods

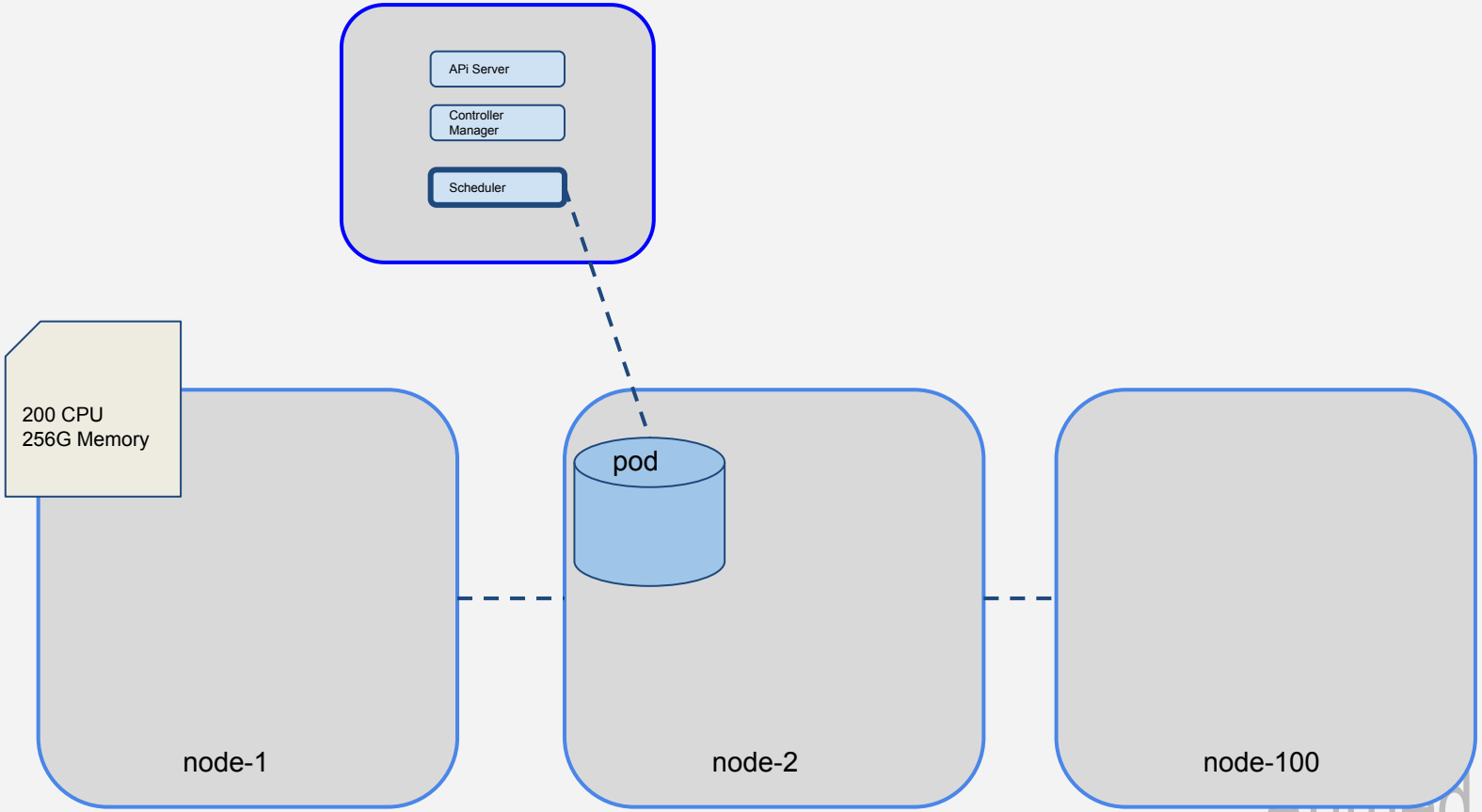
Guaranteed	Burstable	BestEffort

`apiVersion: v1`
`kind: Pod`
`metadata:`
`  name: nginx`
`  namespace: nginx`
`spec:`
`  containers:`
`  - name: nginx`
`  image: nginx`
`  resources:`
`  limits:`
`  memory: "500Mi"`
`  cpu: "1000m"`
`  requests:`
`  memory: "500Mi"`
`  cpu: "1000m"`
	`apiVersion: v1`
	`kind: Pod`
	`metadata:`
	`  name: nginx`
	`  namespace: nginx`
	`spec:`
	`  containers:`
	`  - name: nginx`
	`  image: nginx`
	`  resources:`
	`  limits:`
	`  memory: "500Mi"`
	`  cpu: "1000m"`
	`  requests:`
	`  memory: "200Mi"`
	`  cpu: "500m"`
		`apiVersion: v1`
		`kind: Pod`
		`metadata:`
		`  name: nginx`
		`  namespace: nginx`
		`spec:`
		`  containers:`
		`  - name: nginx`
		`  image: nginx`

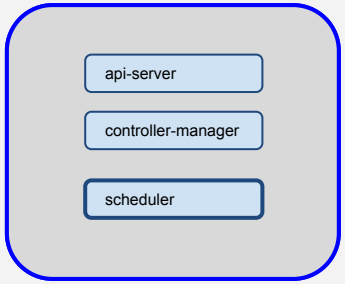
Resources

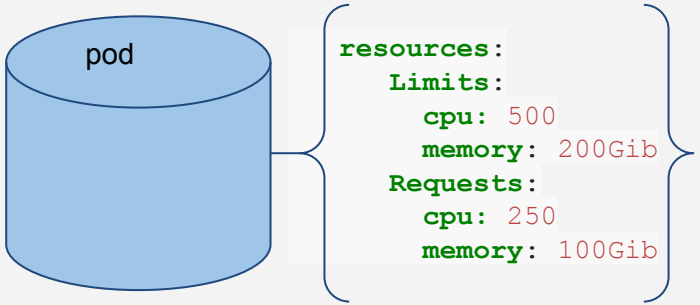


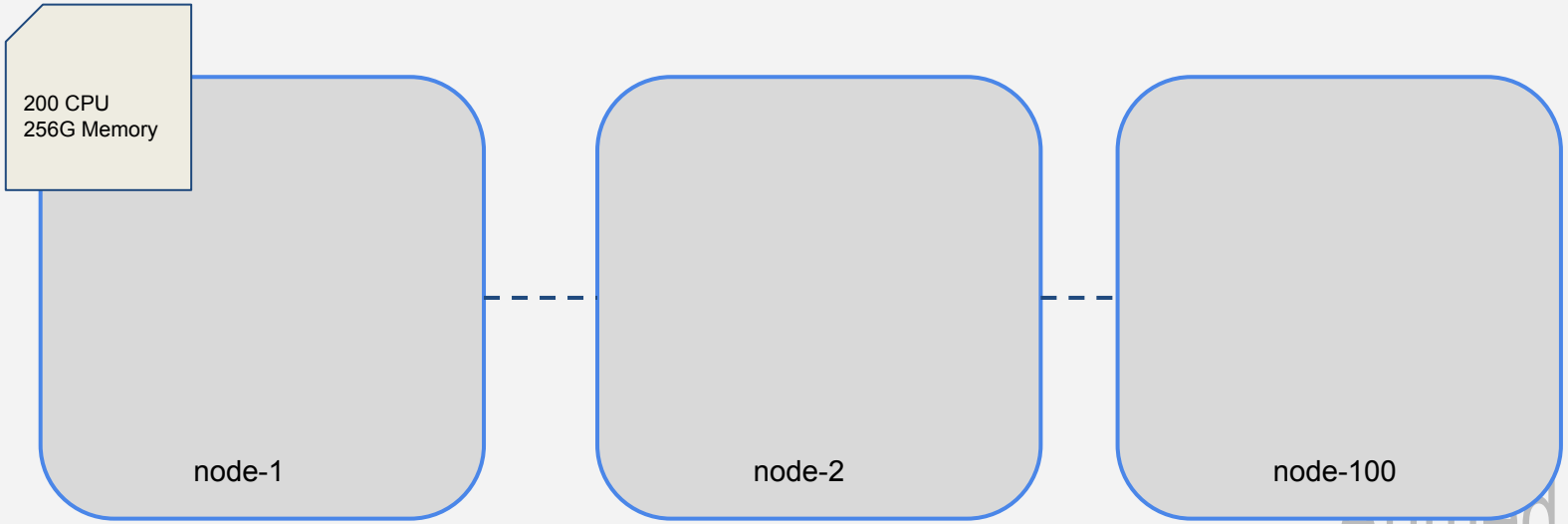
Resources



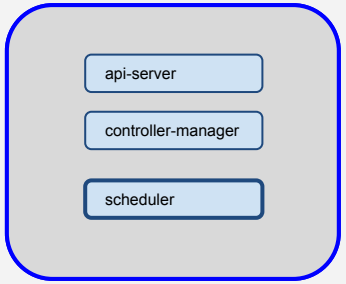
Resources

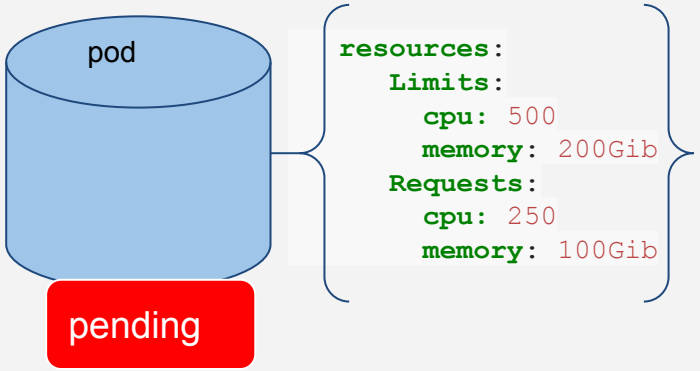


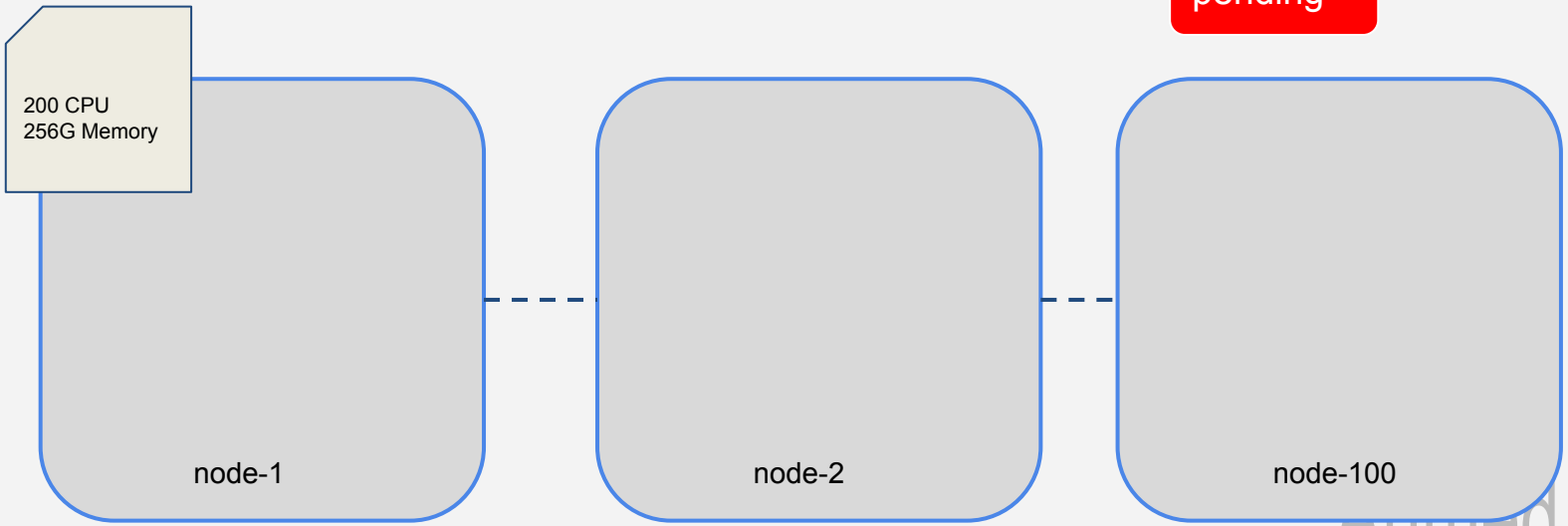




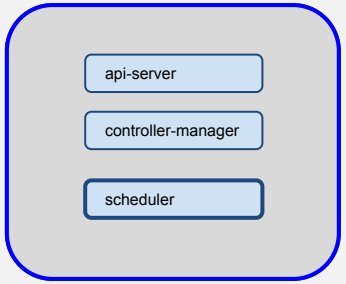
Resources

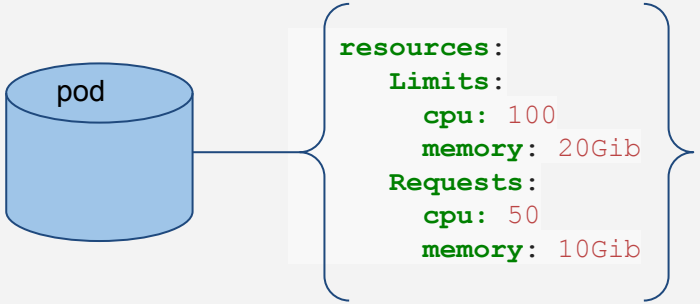


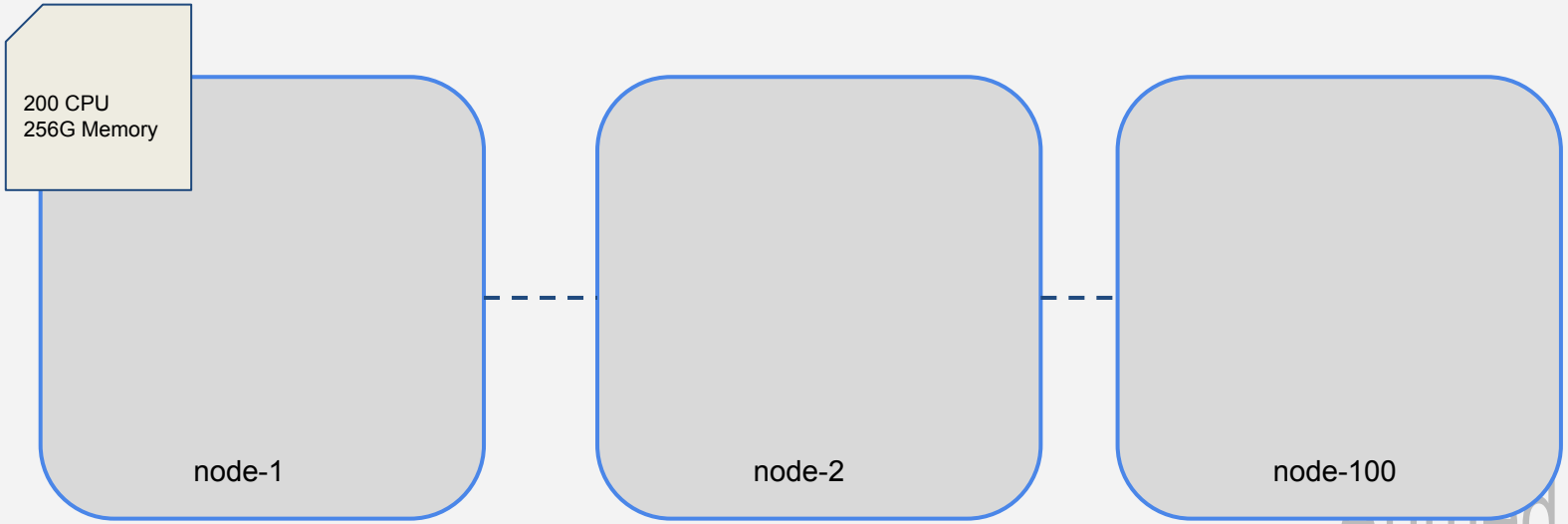




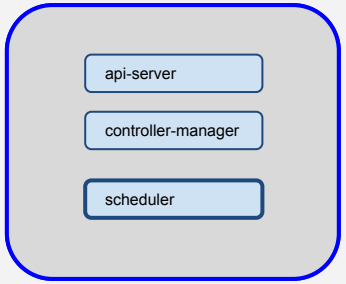
Resources

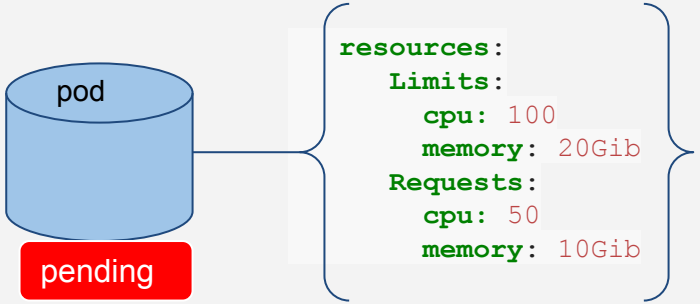


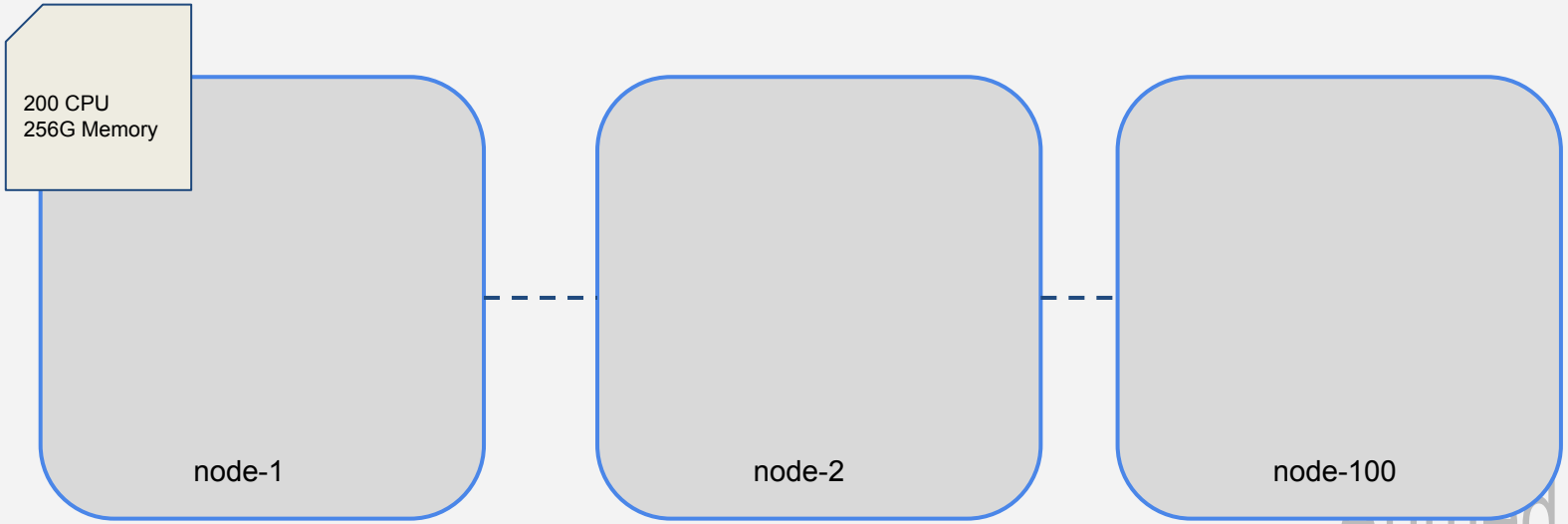




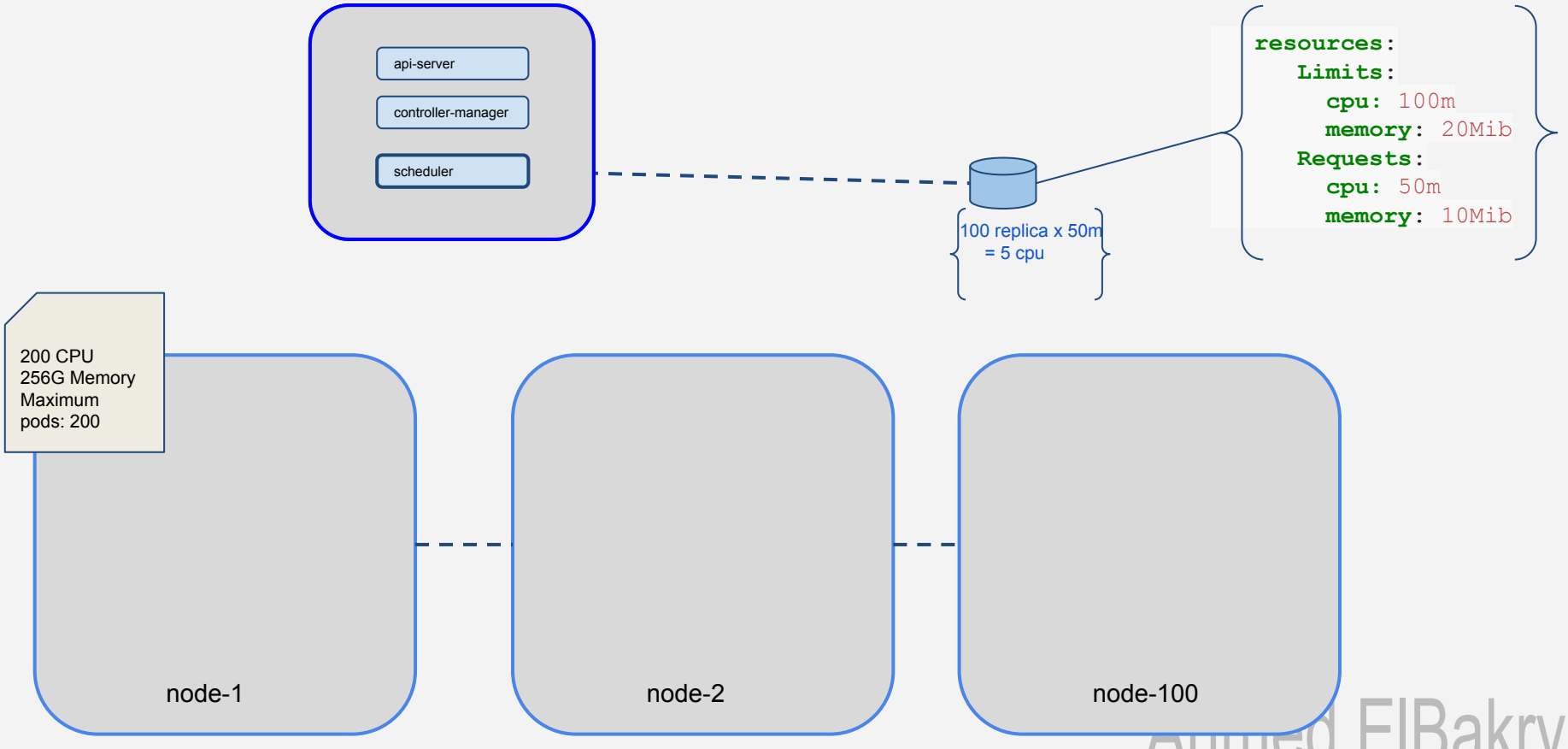
Resources



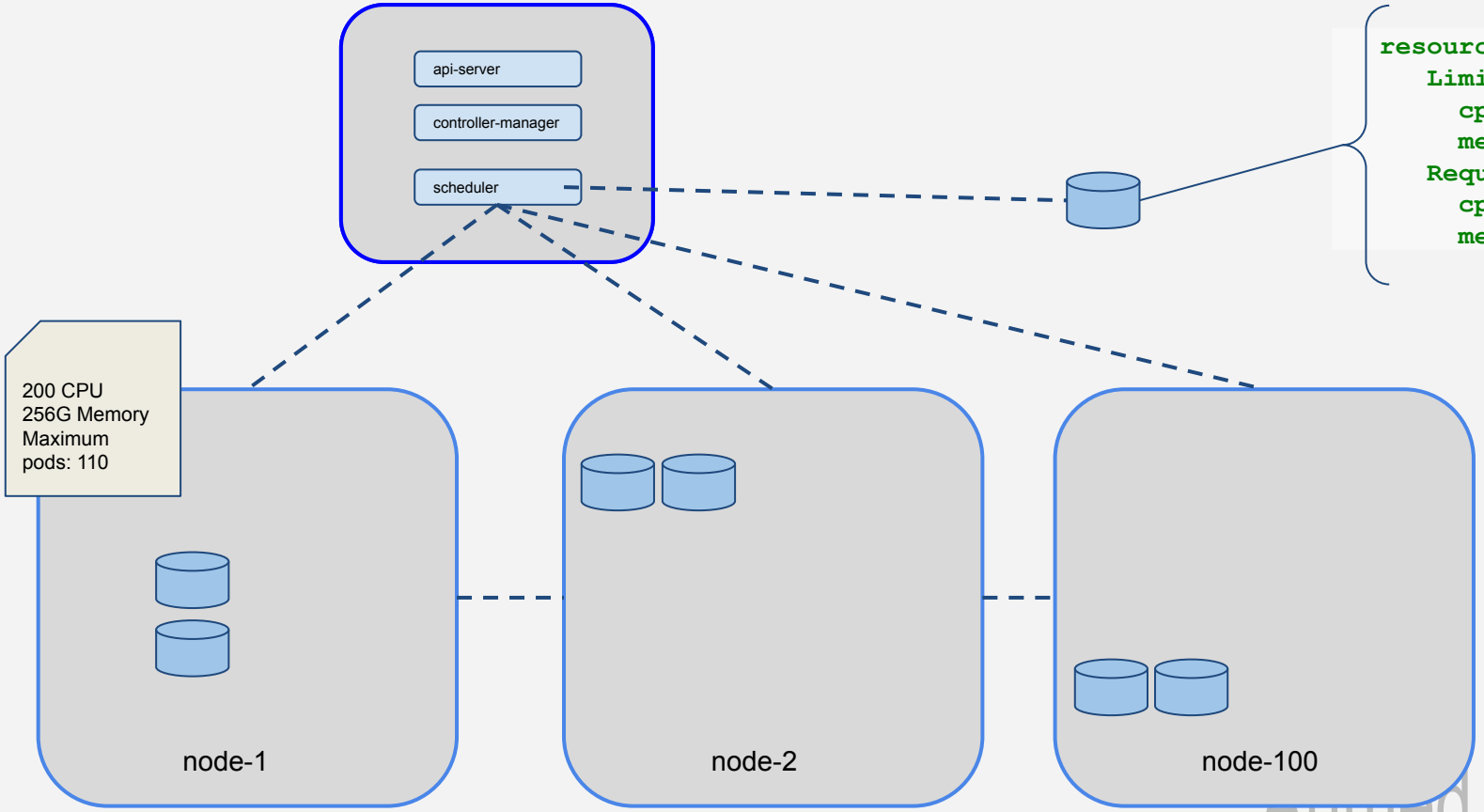




Resources



Resources



Limit Ranges

```yaml
apiVersion: v1
kind: LimitRange
metadata:
  name: limitrange-demo
spec:
  limits:
  - max:
      memory: 1Gi
    min:
      memory: 500Mi
    type: Container
```

Pod creation will fail if:

- Memory is less than 500Mi
- Memory is more than 1Gi

Limit Ranges

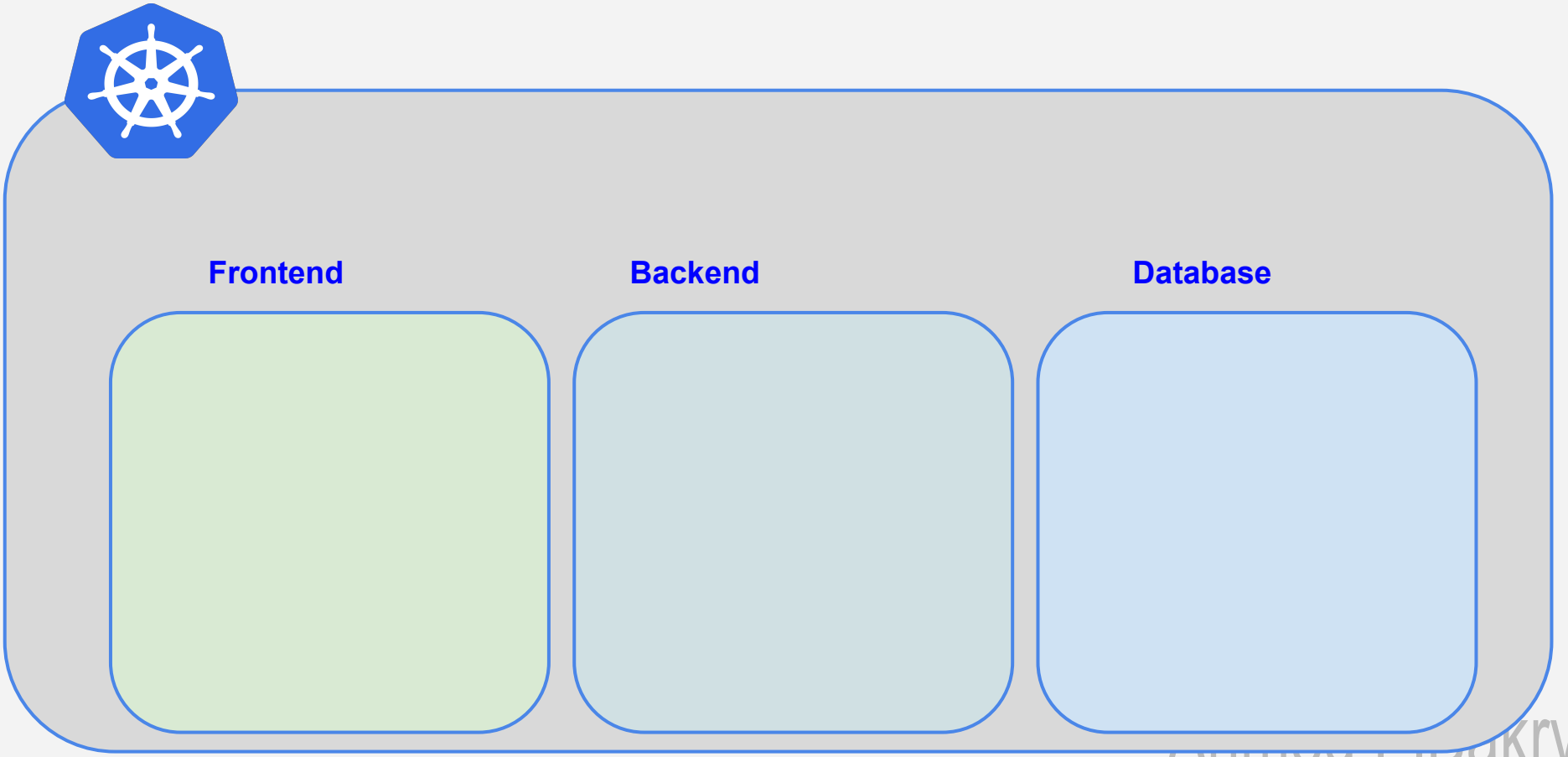
```yaml
apiVersion: v1
kind: LimitRange
metadata:
  name: limitrange-demo
spec:
  limits:
  - max:
      memory: 1Gi
    min:
      memory: 500Mi
    type: Container
```

```yaml
apiVersion: v1
kind: LimitRange
metadata:
  name: limitrange-demo-2
spec:
  limits:
  - type: Container
    max:
      cpu: 2
      memory: 500Mi
    Min:
      cpu: 100m
      memory: 50Mi
    defaultRequest:
      cpu: 200m
      memory: 100Mi
```

Pod creation will fail if:

- Memory is less than 50Mi or more than 500Mi
- CPU is less than 100m or more than 2 cpus.

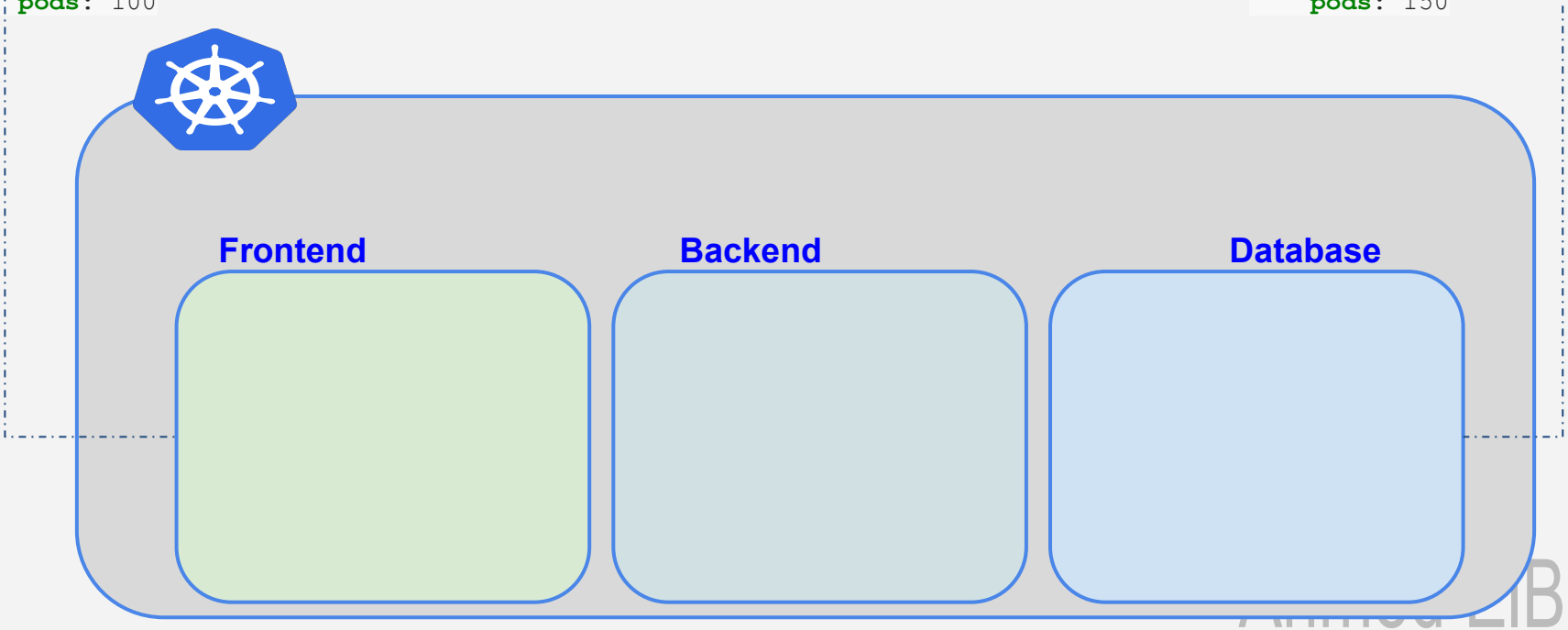
Resource Quota



Resource Quota

```yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: frontend-rq
spec:
  hard:
    requests.cpu: "50"
    requests.memory: 100Gi
    limits.cpu: "100"
    limits.memory: 200Gi
    pods: 100
```

```yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: database-rq
spec:
  hard:
    requests.cpu: "100"
    requests.memory: 500Gi
    limits.cpu: "200"
    limits.memory: 800Gi
    pods: 150
```




DEMO
Ahmed ElBakry


THANK YOU
Ahmed ElBakry

Kubernetes From Scratch - Hands-on

14 - Resources Demo

Ahmed ElBakry

Resource Quota

Demo

- ❏ Check node allocatable space
- ❏ Create pods with different resources
  - ❏ Resources not available in the nodes
  - ❏ Very tiny resources
  - ❏ No resources
- ❏ Create LimitRanges
- ❏ Create Resource Quota
- ❏

Kubernetes From Scratch - Hands-on

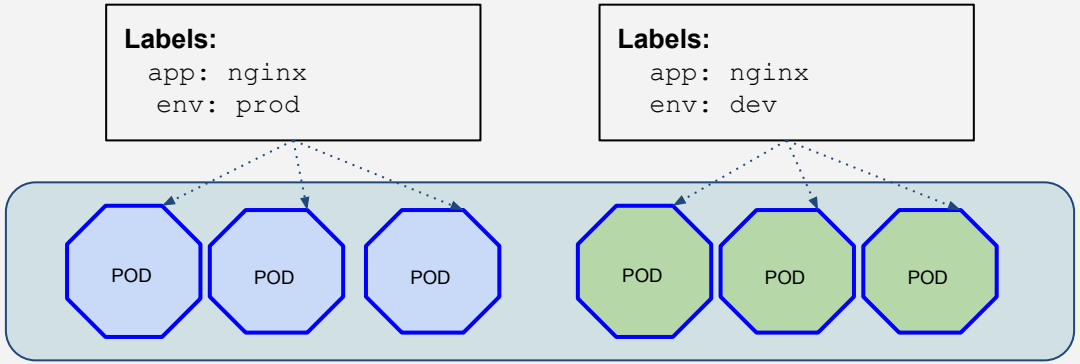
14 - Labels and Selectors

Ahmed ElBakry

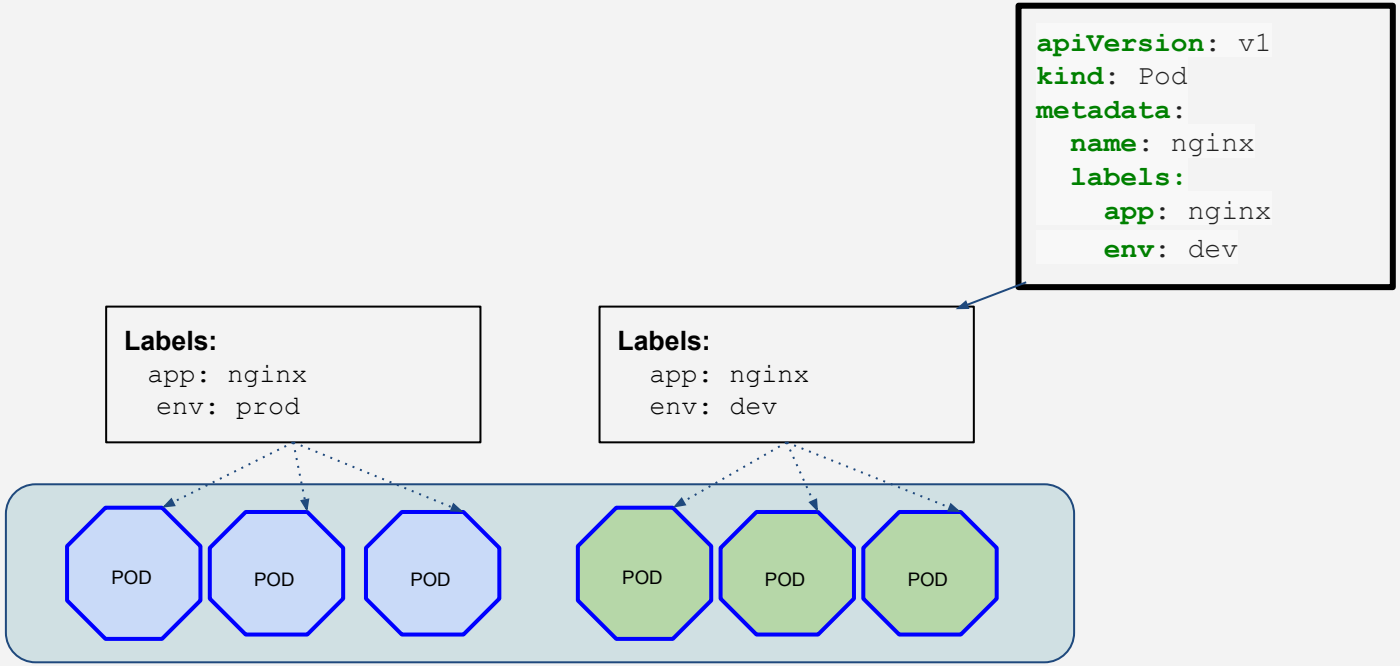
Labels and Selectors

- 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

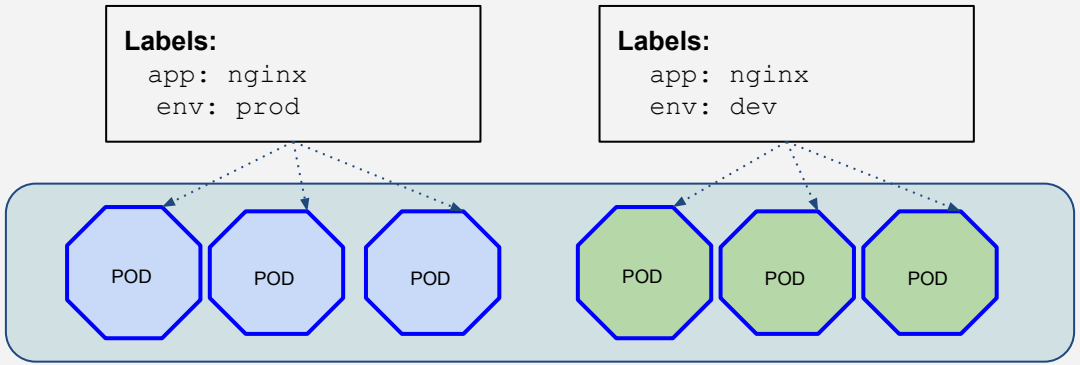


Labels and Selectors

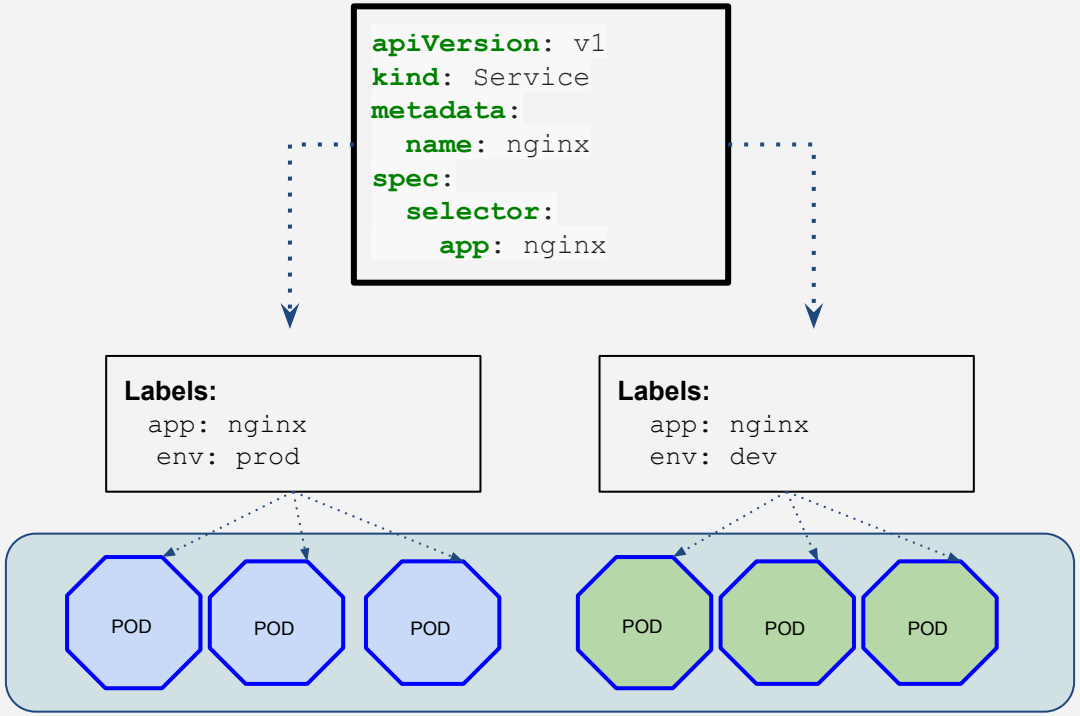


Labels and Selectors

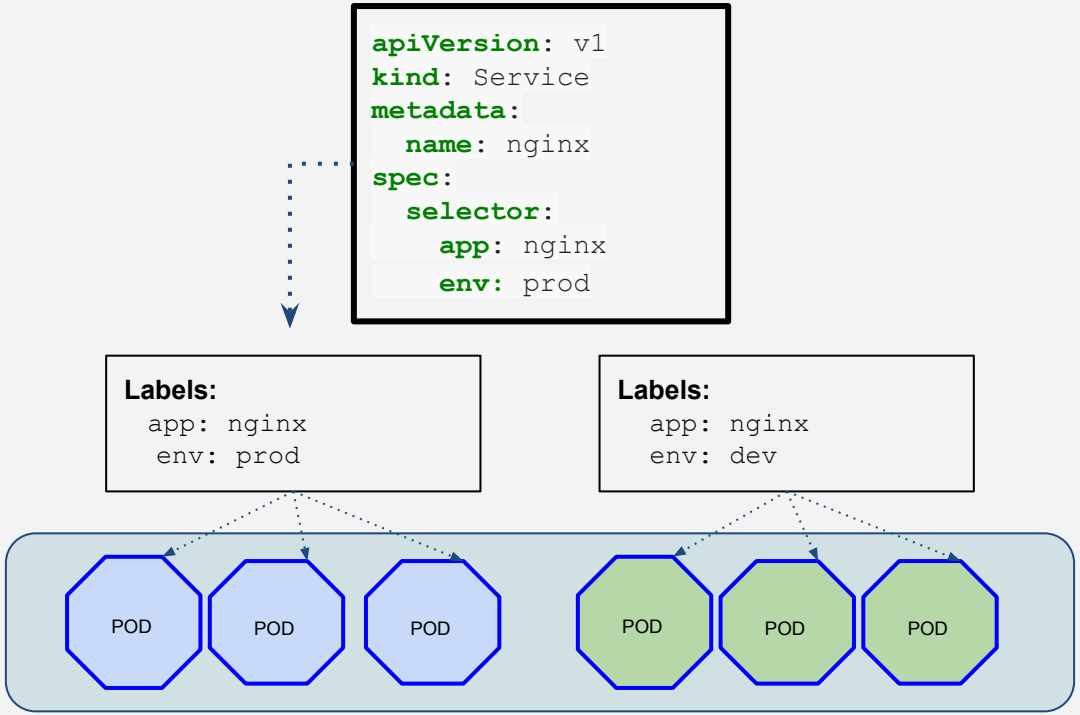
```
apiVersion: v1
kind: Service
metadata:
  name: nginx
spec:
  selector:
    app: nginx
```



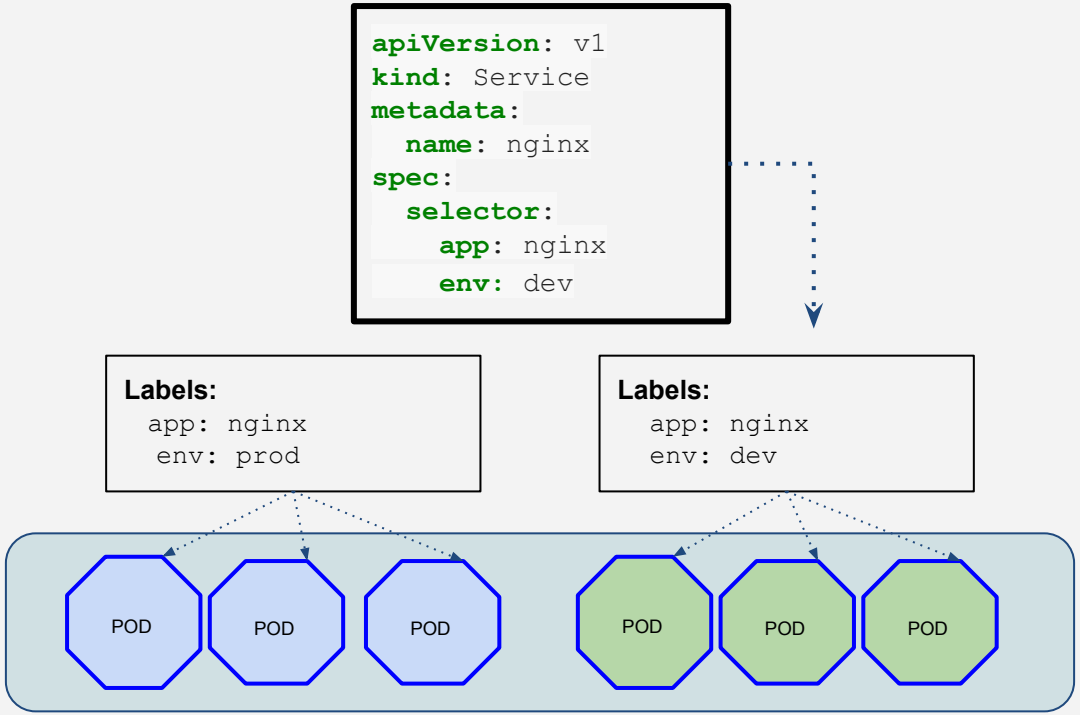
Labels and Selectors



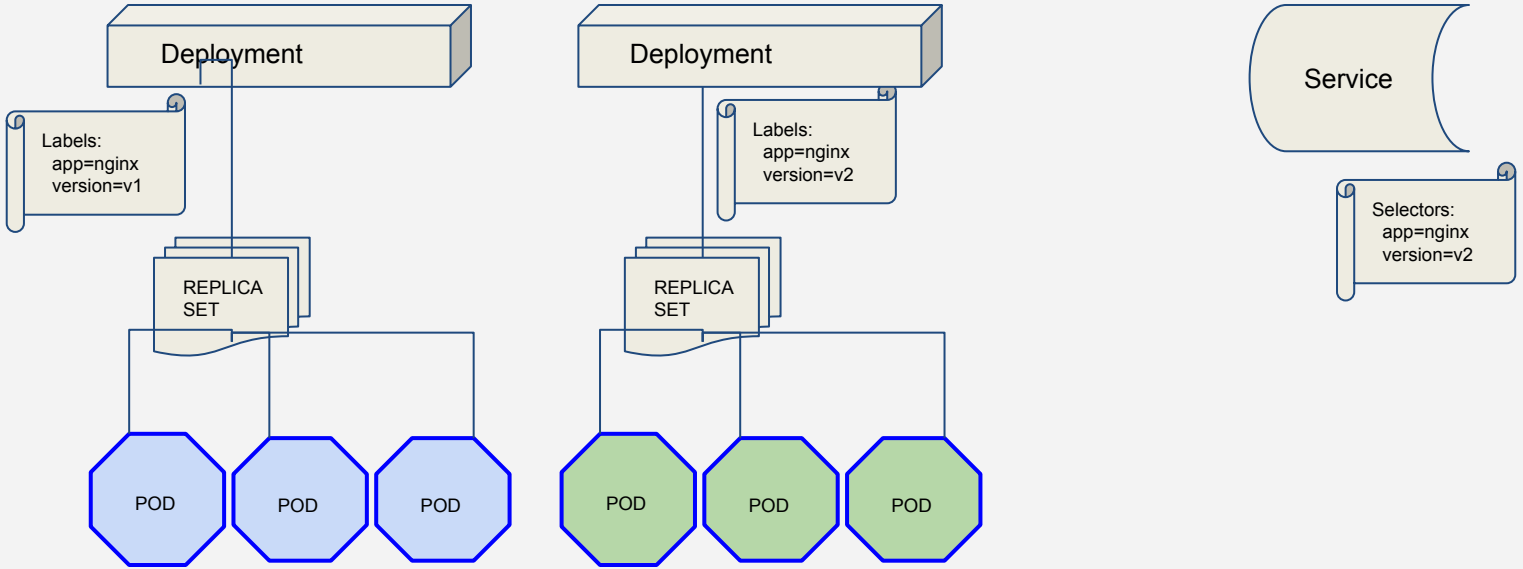
Labels and Selectors



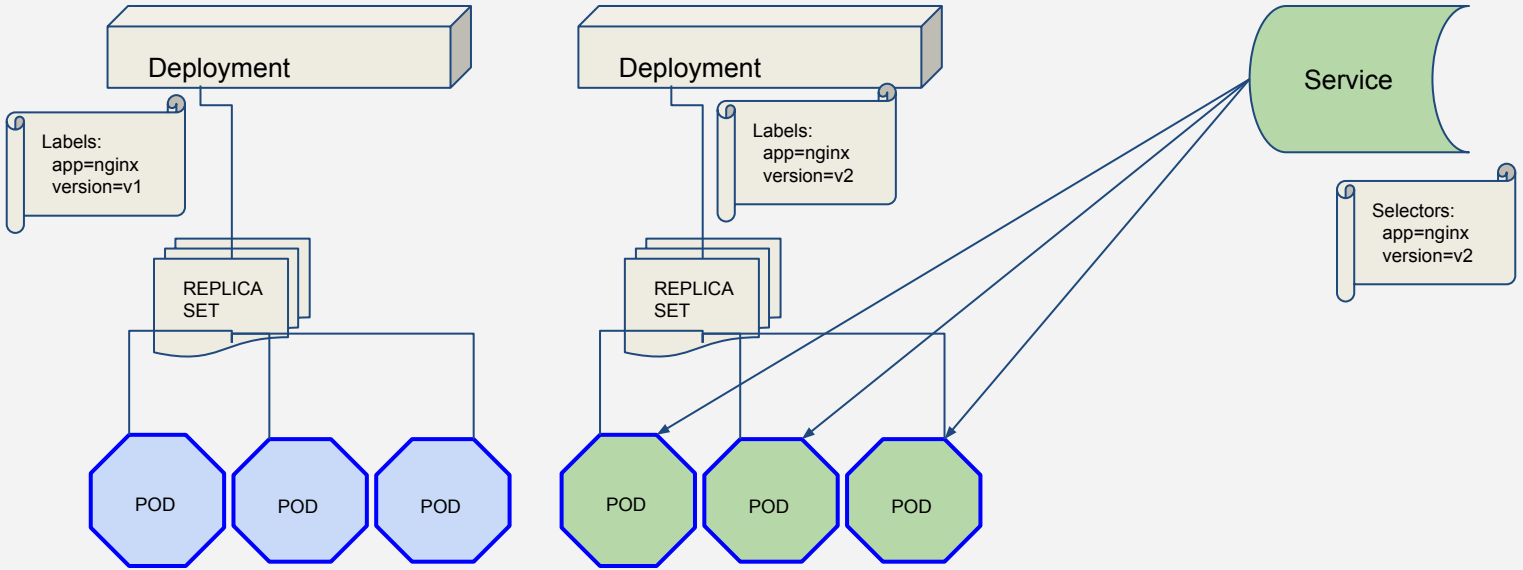
Labels and Selectors



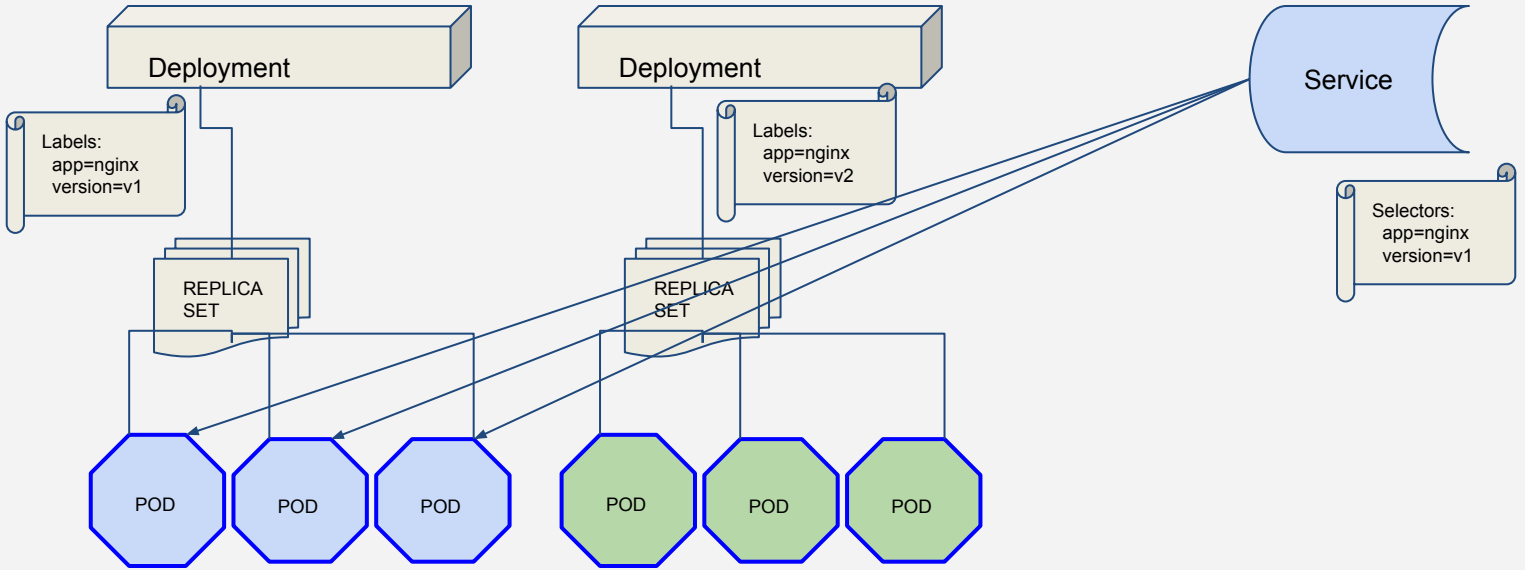
Labels and Selectors



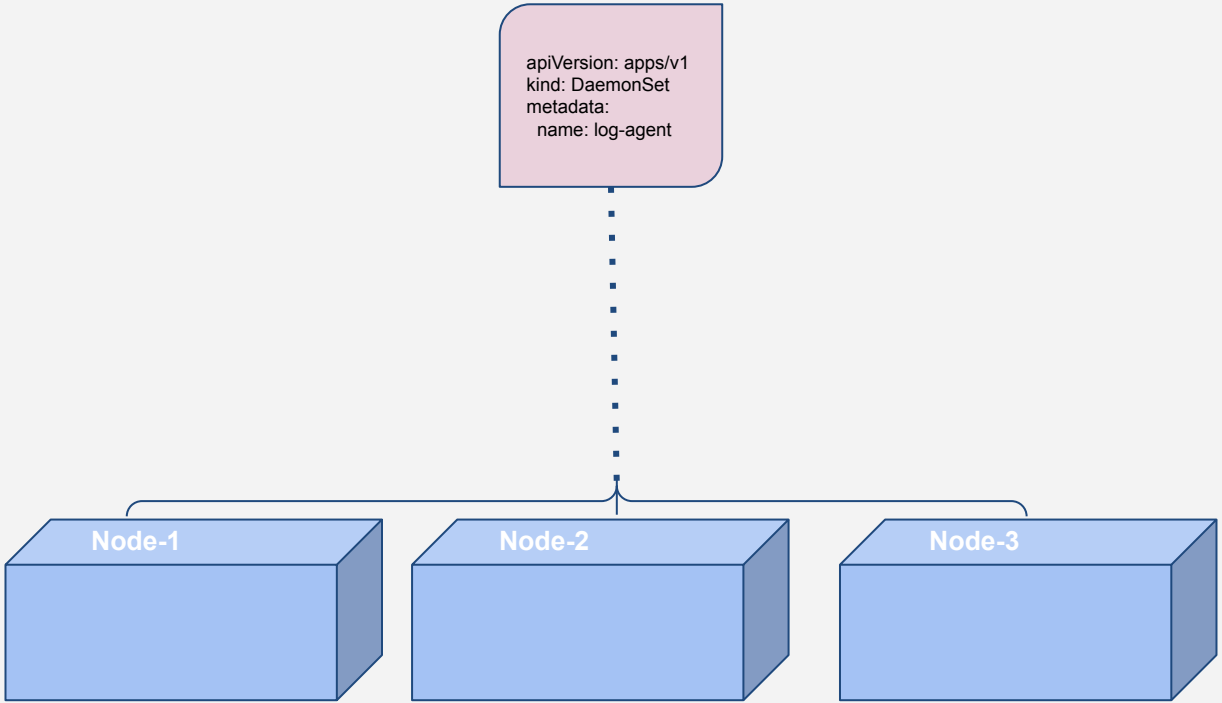
Labels and Selectors



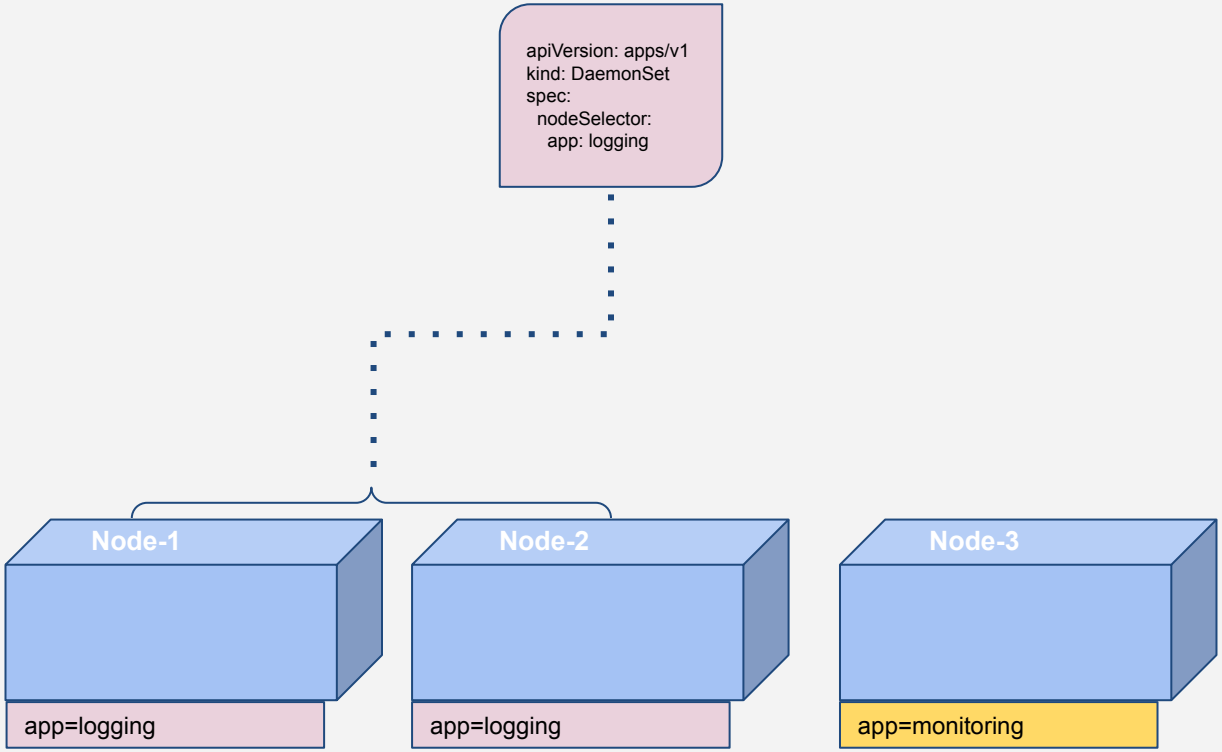
Labels and Selectors



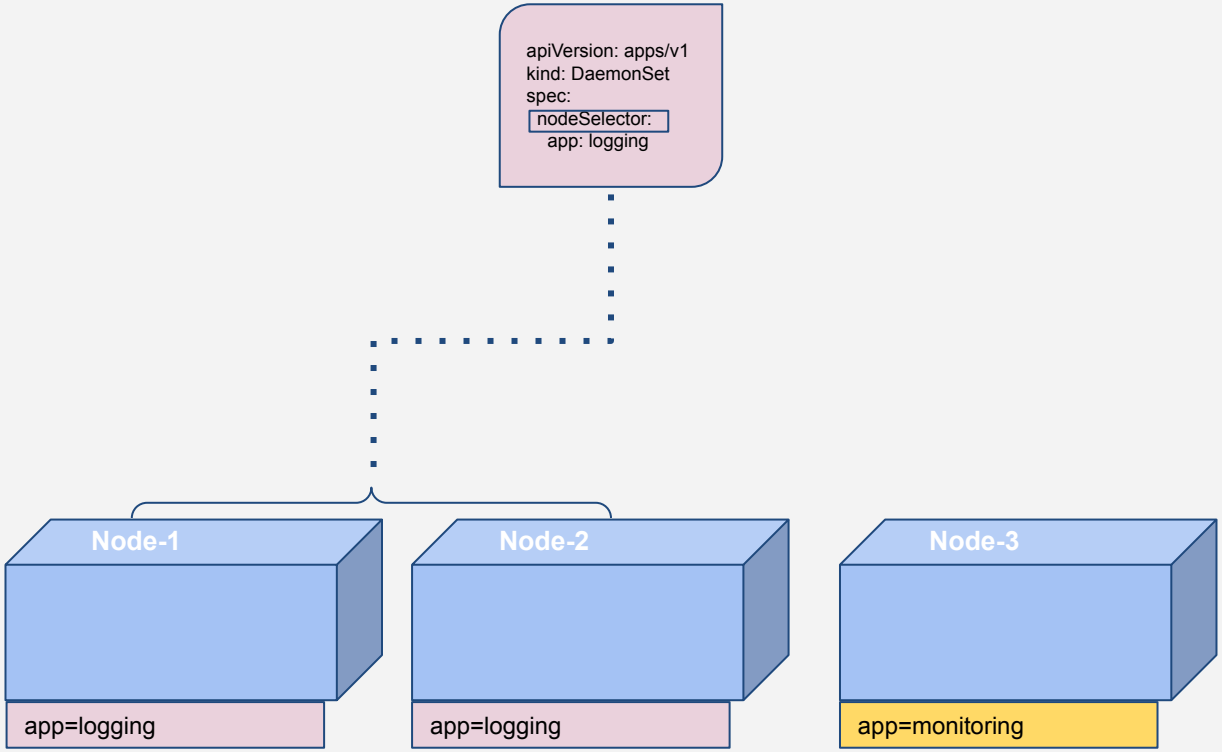
Labels and Selectors



Labels and Selectors

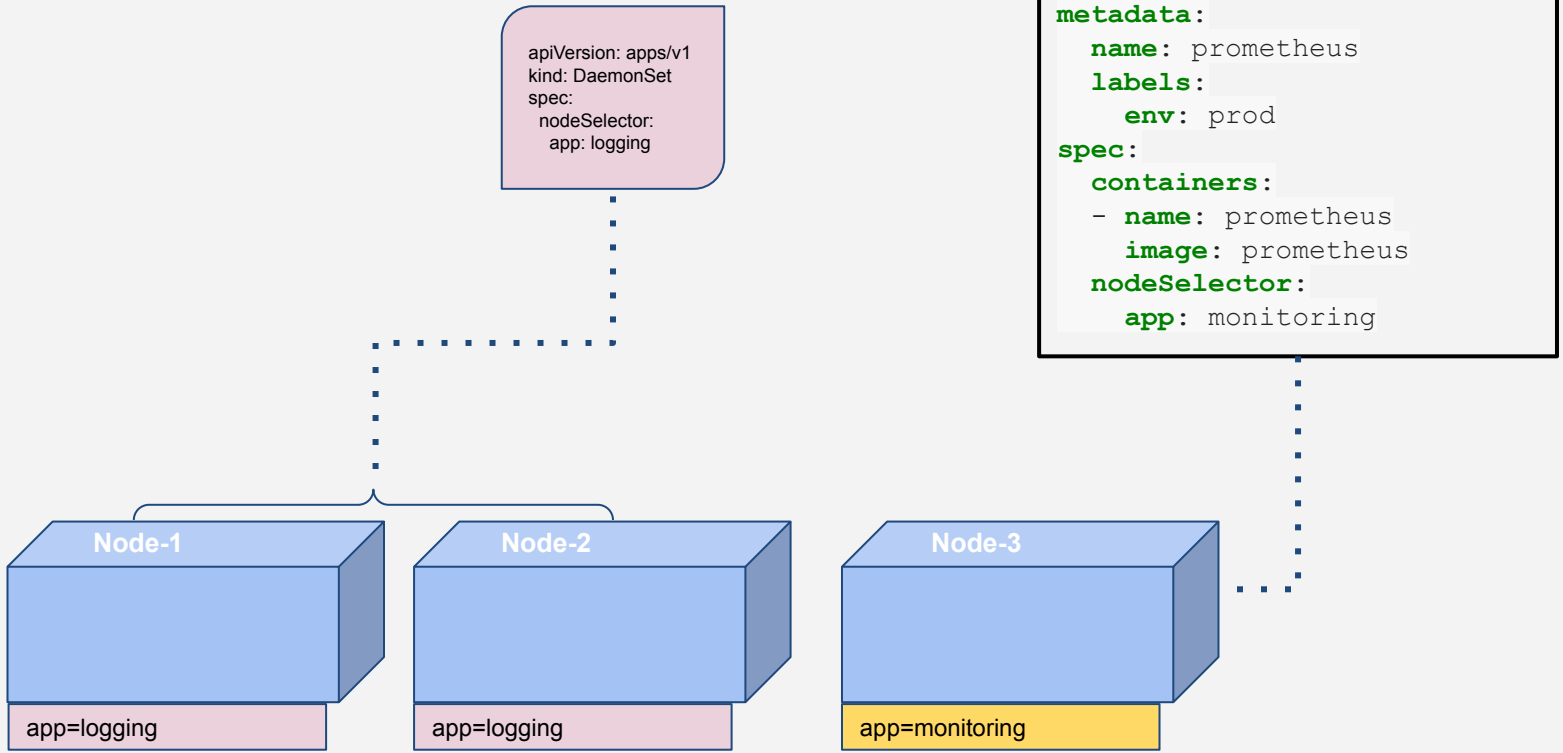


Labels and Selectors



Labels and Selectors

```
apiVersion: v1
kind: Pod
metadata:
  name: prometheus
  labels:
    env: prod
spec:
  containers:
  - name: prometheus
    image: prometheus
  nodeSelector:
    app: monitoring
```




DEMO
Ahmed ElBakry


THANK YOU
Ahmed ElBakry

Kubernetes From Scratch - Hands-on

15 - Nodes

Ahmed ElBakry

Nodes

Nodes

Demo

- ❑ Capacity and Allocatable
- ❑ Check service running on nodes
- ❑ Check running pods on the nodes
- ❑ Taints and Tolerations
- ❑ Check node status
- ❑ Taint the node
- ❑ Drain the node
- ❑
- ❑


THANK YOU
Ahmed ElBakry

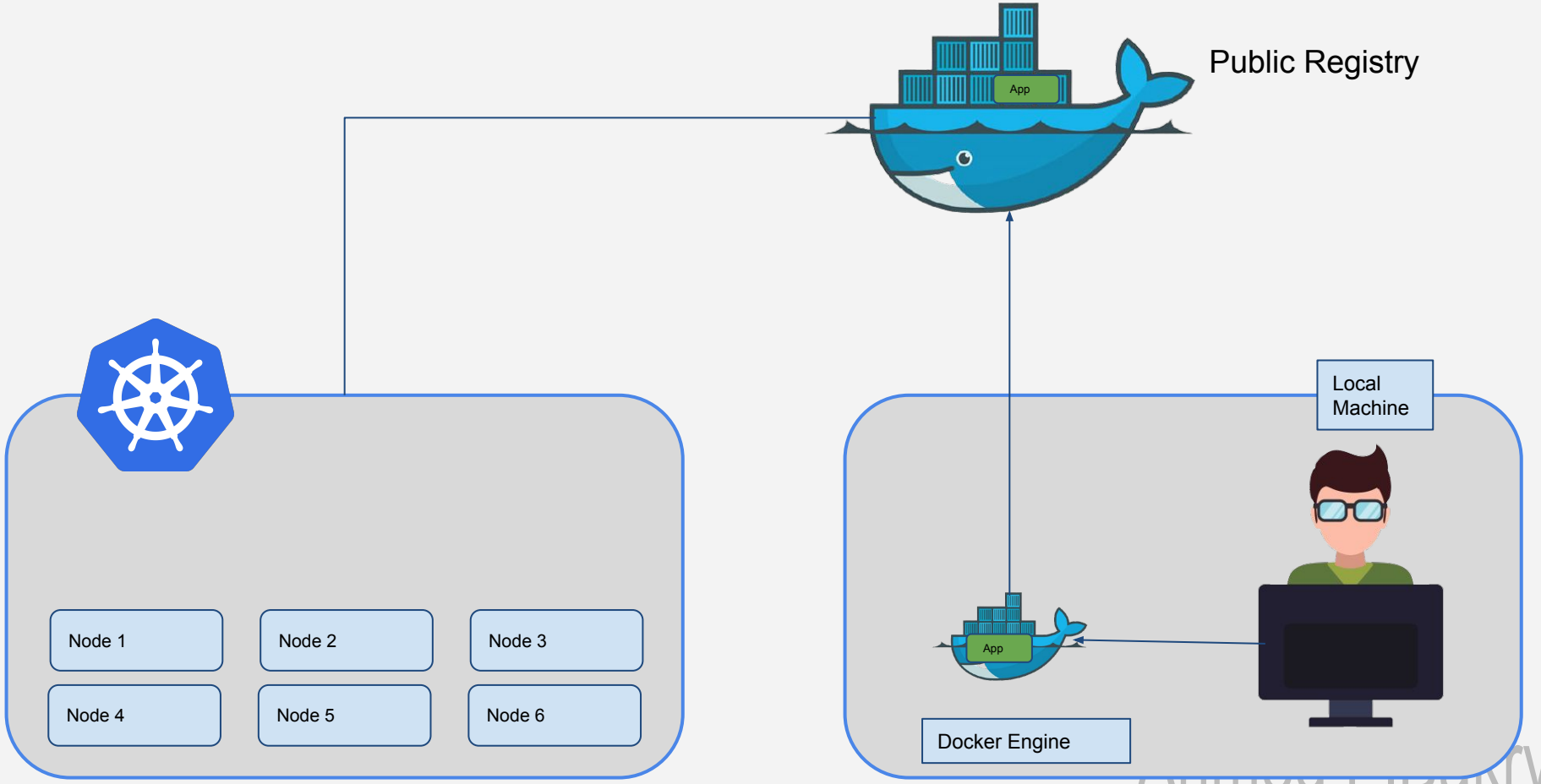
Module 5

Pods

Pods

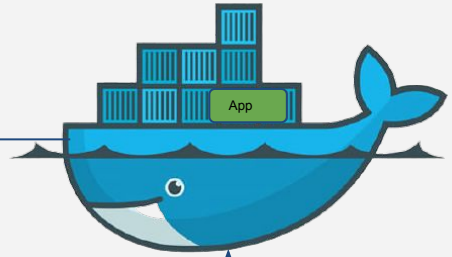
- Pods are the smallest deployable units in Kubernetes.
- A Pod can be one or more containers.

Pods


App
Public Registry
Local Machine
Node 1
Node 2
Node 3
Node 4
Node 5
Node 6
App
Docker Engine

Pods

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



Public Registry

Node 1

Node 2

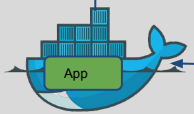
Node 3

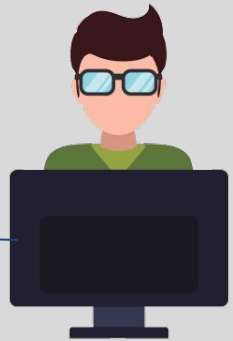
Node 4

Node 5

Node 6

Local Machine

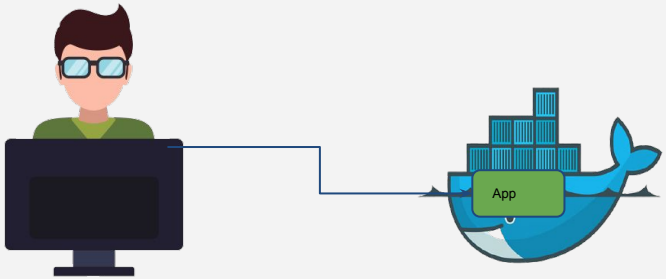




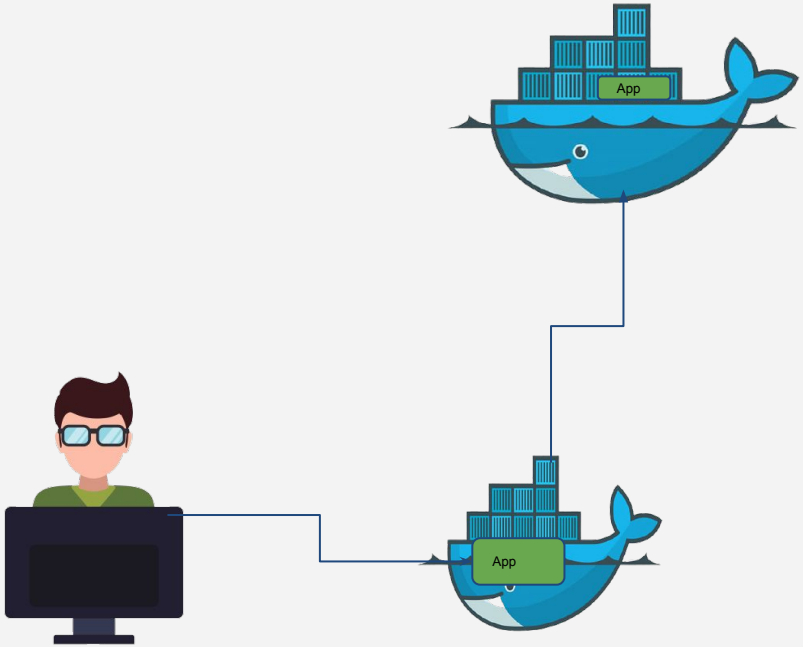
Docker Engine

Pods

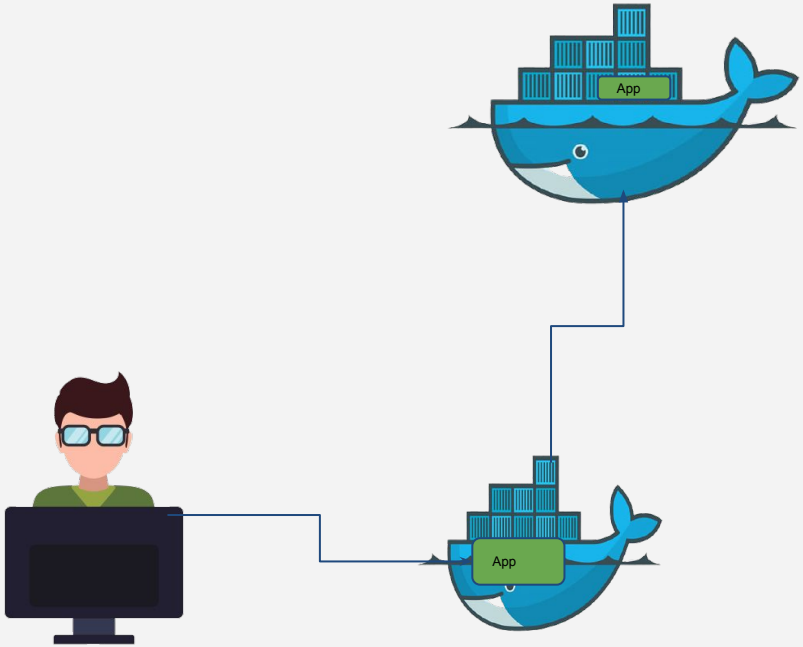
Pods



Pods

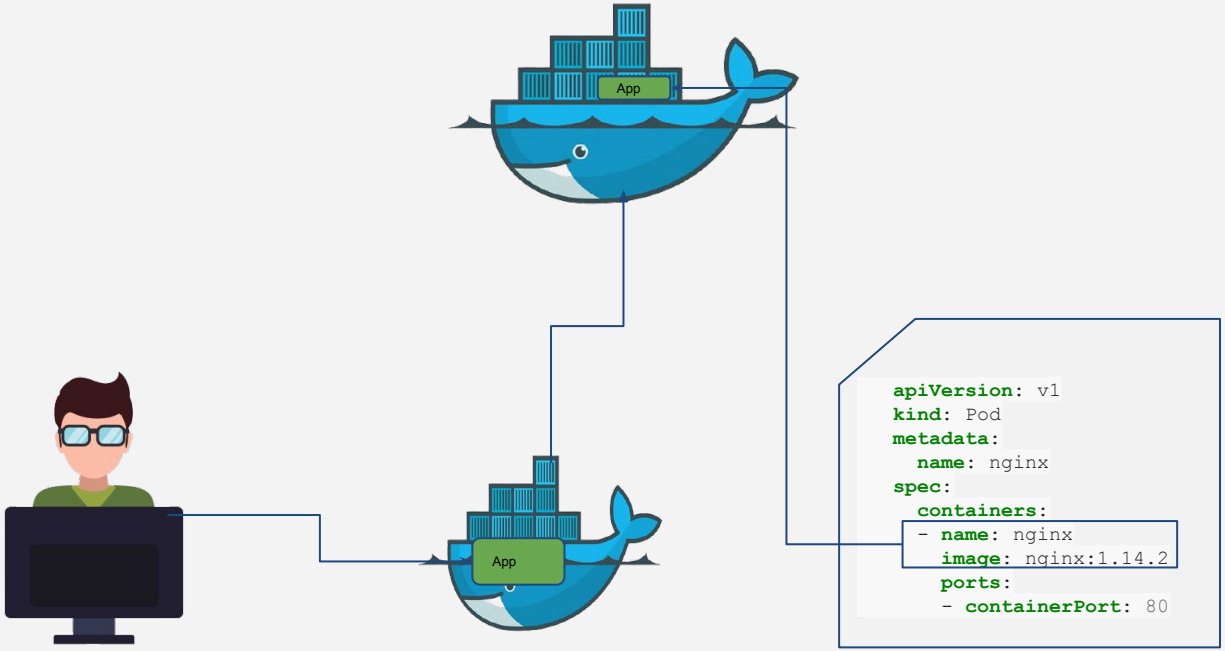


Pods

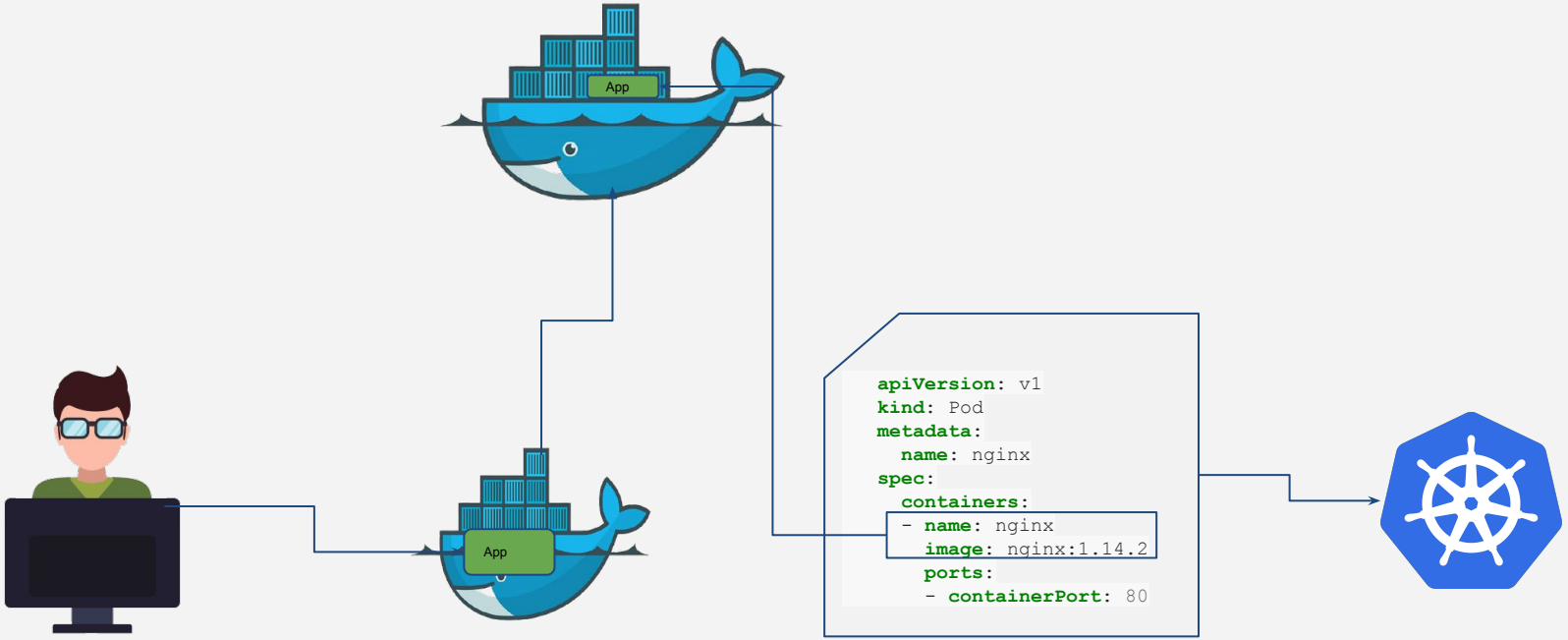



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```

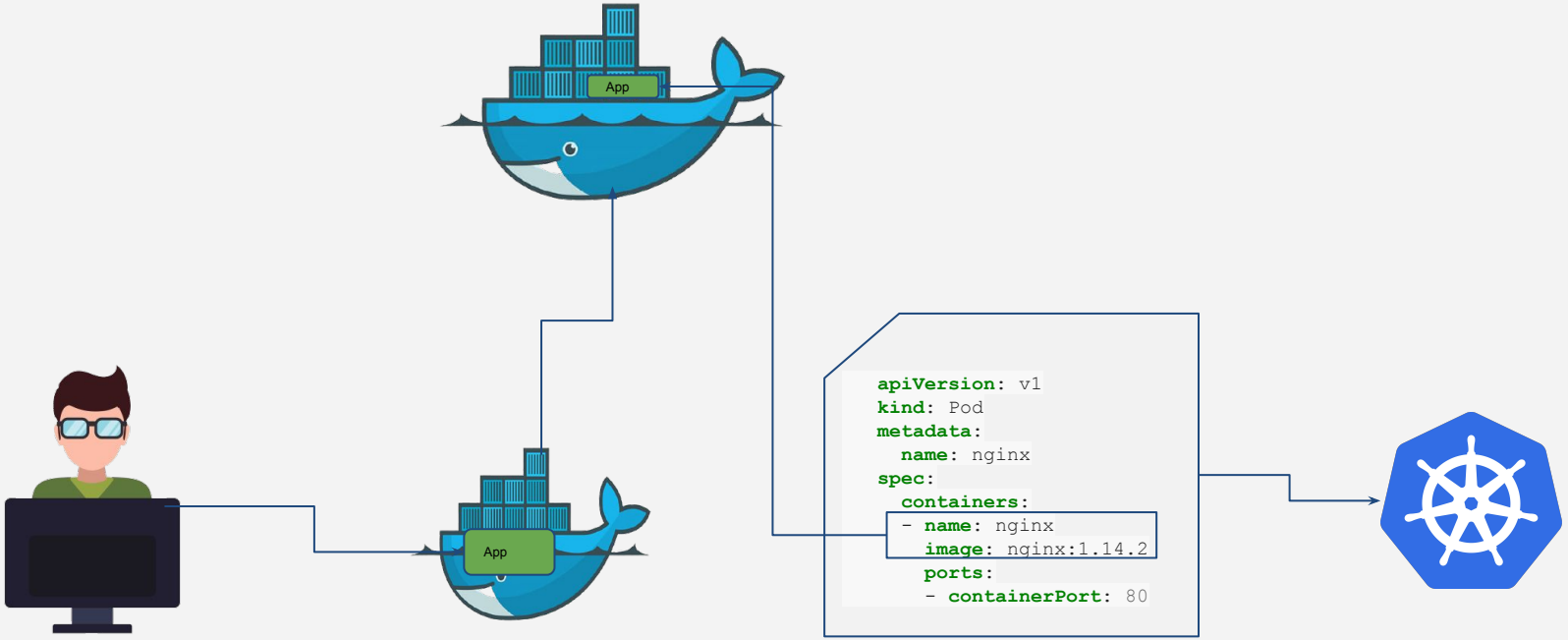
Pods



Pods



Pods



Pods

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

Pods

```
(⎈ |kind-c11:default)-> kubectl explain pod
KIND:     Pod
VERSION:  v1

DESCRIPTION:
     Pod is a collection of containers that can run on a host. This resource is
     created by clients and scheduled onto hosts.

FIELDS:
   apiVersion           <string>
     APIVersion defines the versioned schema of this representation of an
     object. Servers should convert recognized schemas to the latest internal
     value, and may reject unrecognized values. More info:
     https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources

   kind      <string>
     Kind is a string value representing the REST resource this object
     represents. Servers may infer this from the endpoint the client submits
     requests to. Cannot be updated. In CamelCase. More info:
     https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds

   metadata             <Object>
     Standard object's metadata. More info:
     https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata

   spec      <Object>
     Specification of the desired behavior of the pod. More info:
     https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status

   status    <Object>
     Most recently observed status of the pod. This data may not be up to date.
     Populated by the system. Read-only. More info:
     https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status
```

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

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

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

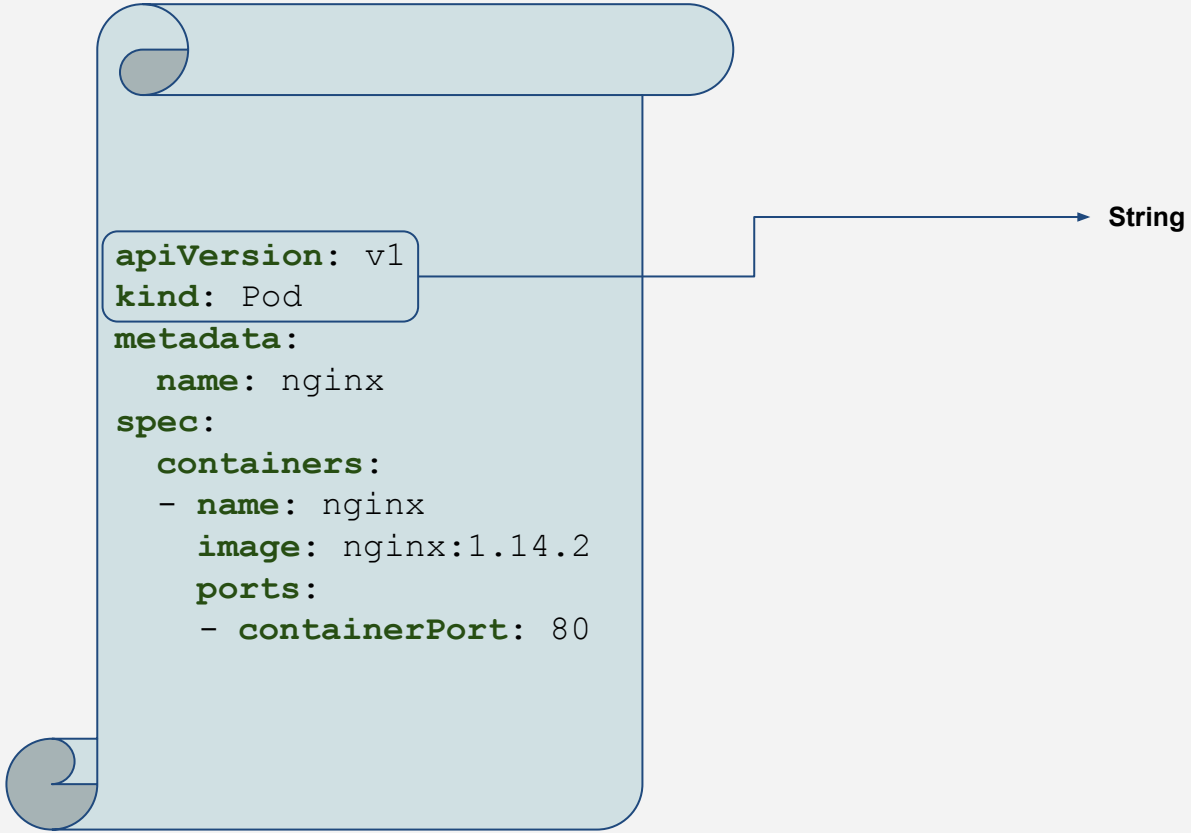
Pods

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

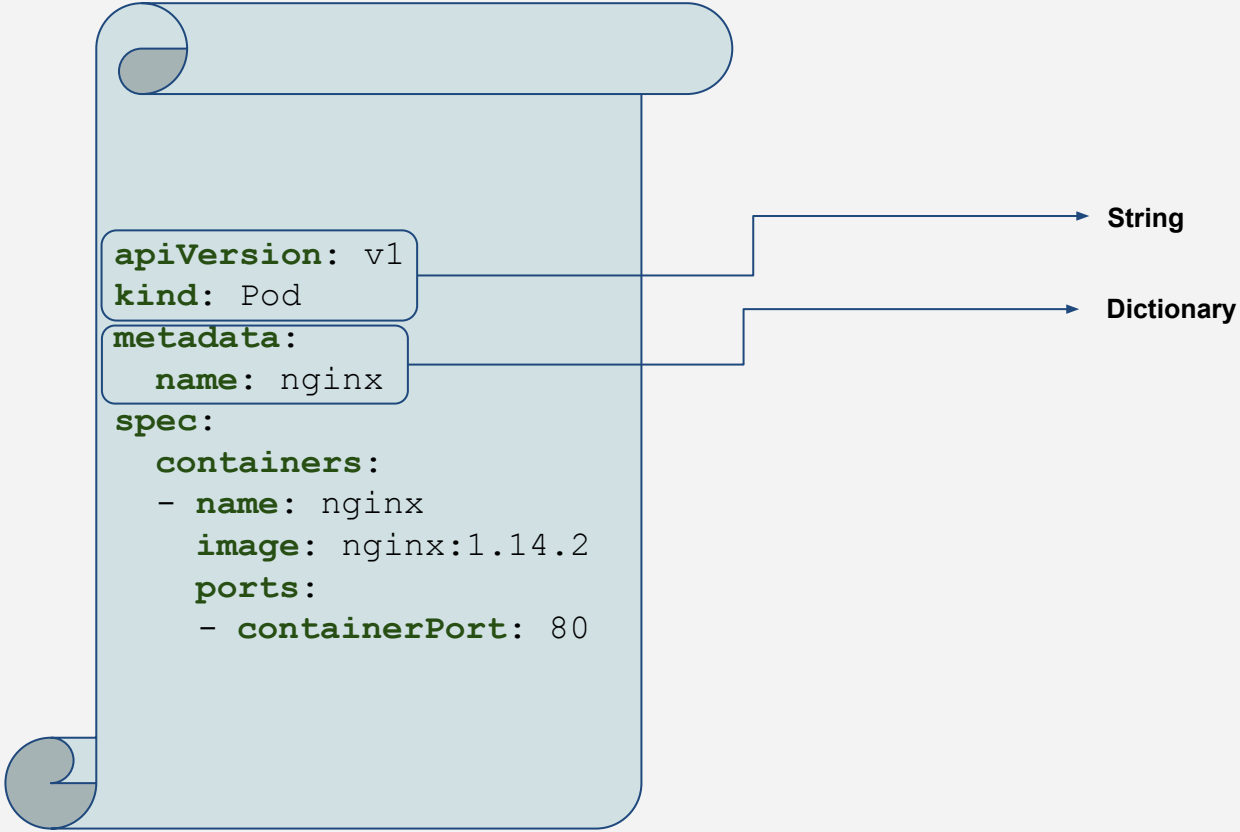
Pods

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

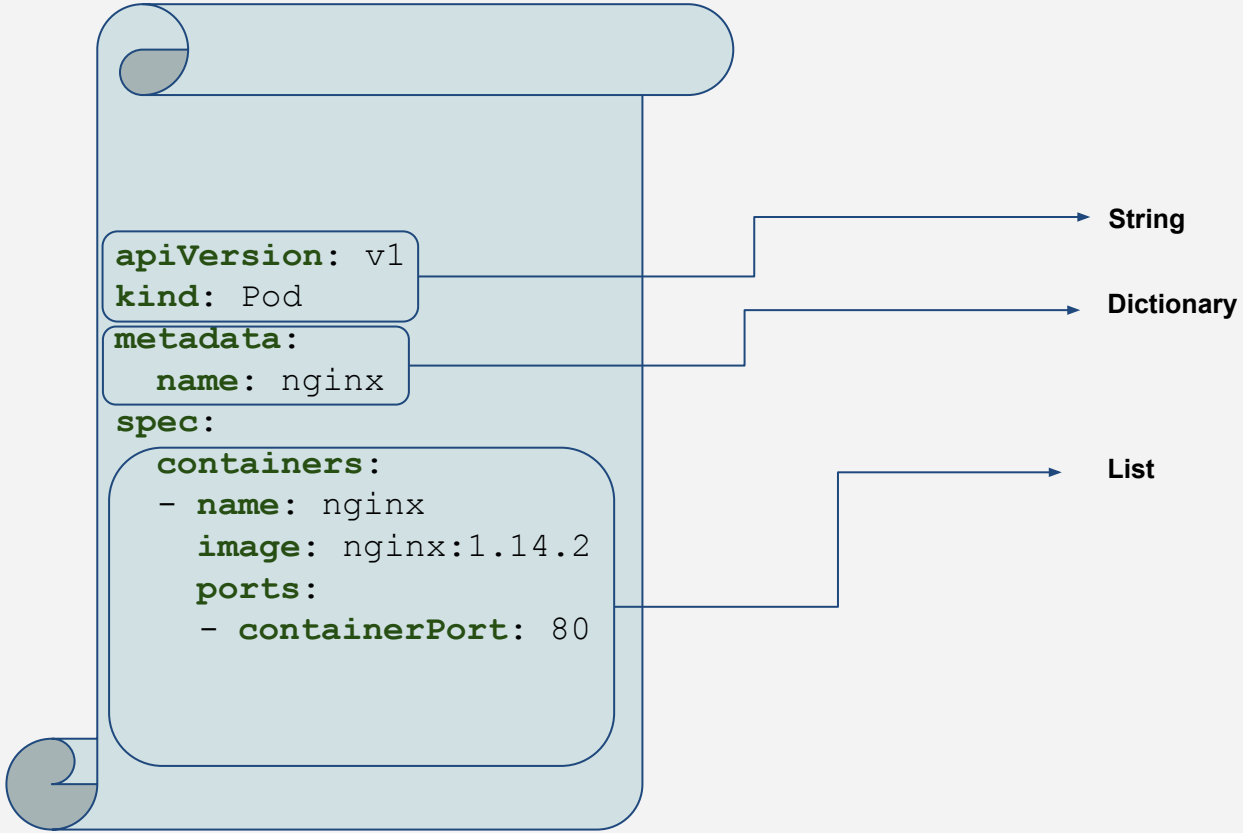
Pods



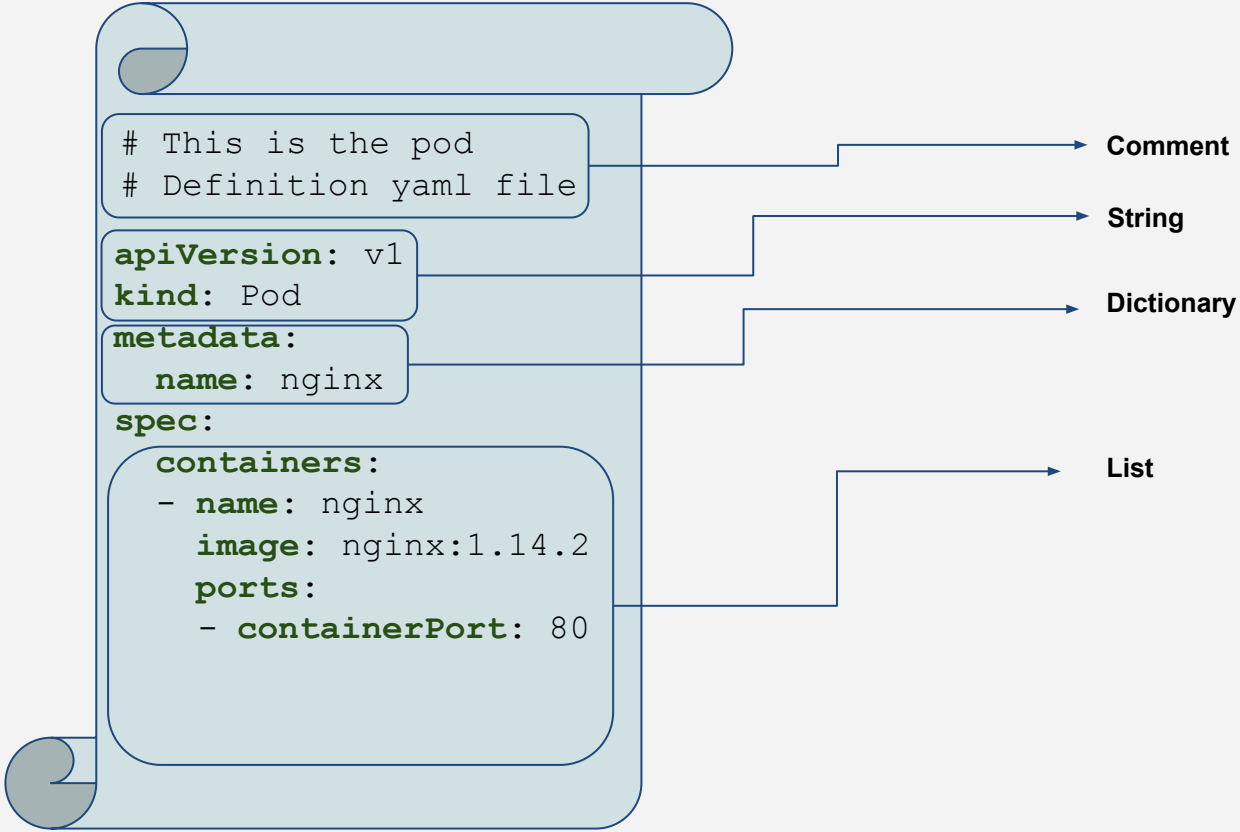
Pods



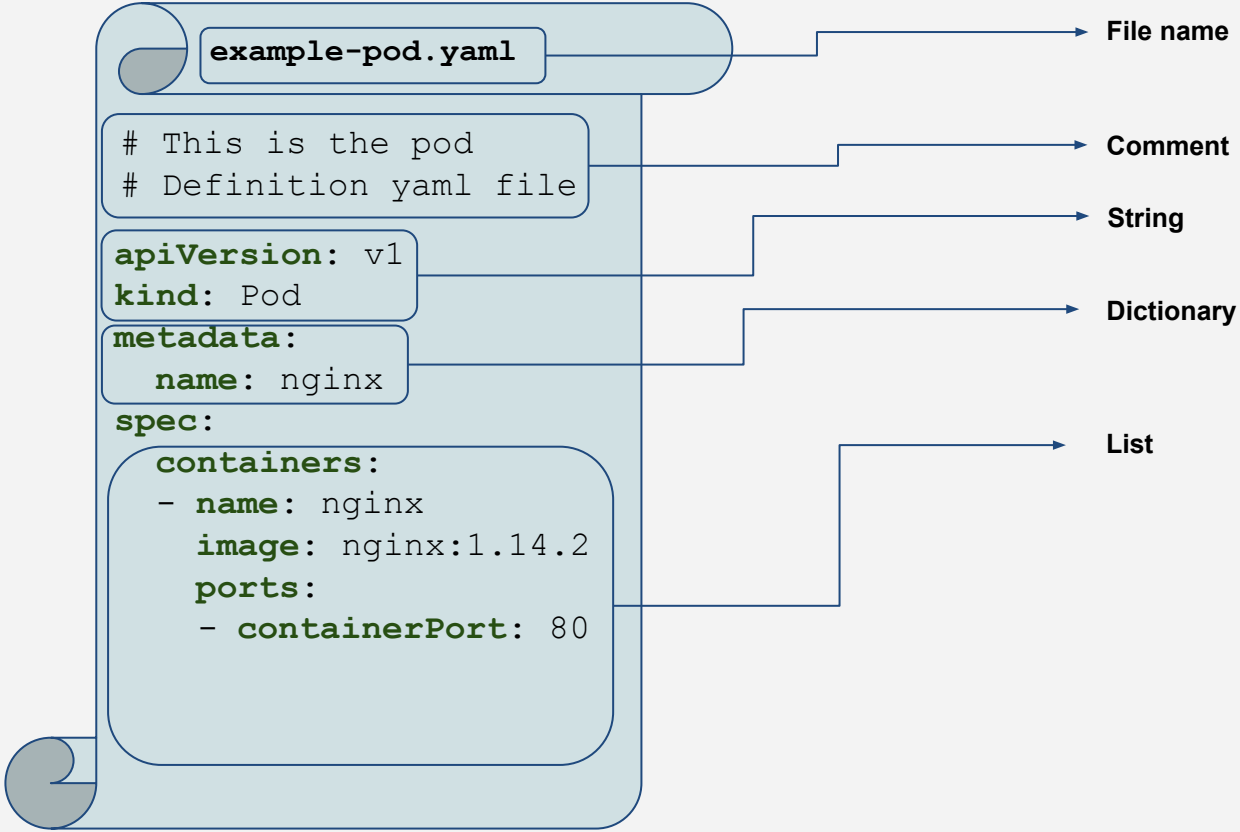
Pods



Pods



Pods



Pods

```
apiVersion: v1
kind: Pod
metadata:
  name: example-pod
spec:
  containers:
    - name: busybox
      image: busybox:1.20
      volumeMounts:
        - name: config-vol
          mountPath: /config
      command:
        - /bin/sh
        - -ec
        - |
          if [ -f /persistent/log ]; then
            echo -n "Found old state starting at "; head -n1 /persistent/log
          else
            echo -n "Starting with a fresh state"
          fi
          while sleep 5; do date | tee -a /persistent/log; done
  volumes:
  - name: config-vol
    persistentVolumeClaim:
      claimName: my-lamp-site-data
```

Pods

```
apiVersion: v1
kind: Pod
metadata:
  name: example-pod
Spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
              - key: kubernetes.io/hostname
                operator: Exists
  containers:
    - name: busybox
      image: busybox:1.20
      volumeMounts:
        - name: config-vol
          mountPath: /config
      command:
        - /bin/sh
        - -ec
        - |
          if [ -f /persistent/log ]; then
            echo -n "Found old state starting at "; head -n1 /persistent/log
          else
            echo -n "Starting with a fresh state"
          fi
          while sleep 5; do date | tee -a /persistent/log; done
  volumes:
  - name: config-vol
    persistentVolumeClaim:
      claimName: my-lamp-site-data
```

Pods

```
apiVersion: v1
kind: Pod
metadata:
  name: example-pod
spec:
  containers:
    - name: busybox
      image: busybox:1.20
      resources:
        requests:
          memory: "64Mi"
          cpu: "250m"
        limits:
          memory: "128Mi"
          cpu: "500m"
      volumeMounts:
        - name: config-vol
          mountPath: /config
      command:
        - /bin/sh
        - -ec
        - |
          if [ -f /persistent/log ]; then
            echo -n "Found old state starting at "; head -n1 /persistent/log
          else
            echo -n "Starting with a fresh state"
          fi
          while sleep 5; do date | tee -a /persistent/log; done
  volumes:
  - name: config-vol
    persistentVolumeClaim:
      claimName: my-lamp-site-data
```

Pods

```
apiVersion: v1
kind: Pod
metadata:
  name: example-pod
spec:
  containers:
    - name: busybox
      image: busybox:1.20
      resources:
        requests:
          memory: "64Mi"
          cpu: "250m"
        limits:
          memory: "128Mi"
          cpu: "500m"
      volumeMounts:
        - name: config-vol
          mountPath: /config
      command:
        - /bin/sh
        - -ec
        - |
          if [ -f /persistent/log ]; then
            echo -n "Found old state starting at "; head -n1 /persistent/log
          else
            echo -n "Starting with a fresh state"
          fi
          while sleep 5; do date | tee -a /persistent/log; done
  volumes:
  - name: config-vol
    persistentVolumeClaim:
      claimName: my-lamp-site-data
```

Pods

```
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
    ports:
    - containerPort: 80
```

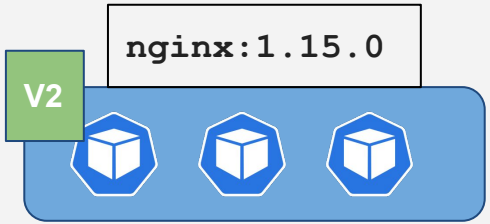
```
(⎈ |kind-c1:default)-> kubectl create -f /tmp/pod.yaml
The Pod "nginx" is invalid: spec.containers[0].image: Required value
```

Pods

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: 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
```



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

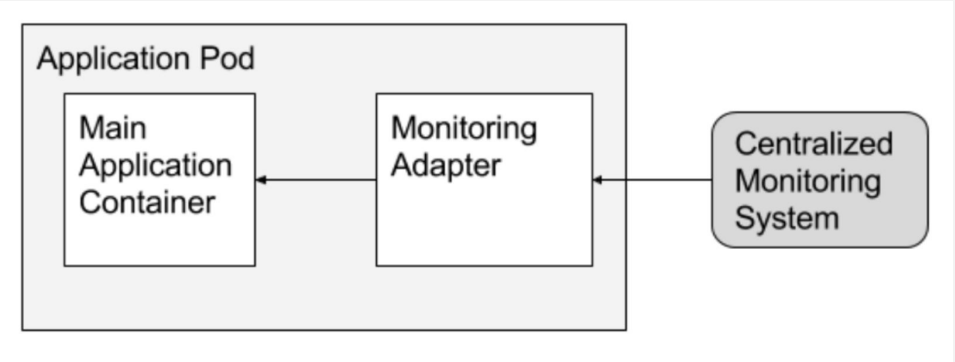
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
  - name: php-fpm
    image: php-fpm:1.14.2
    resources:
      limits:
        memory: "200Mi"
      requests:
        memory: "100Mi"
```

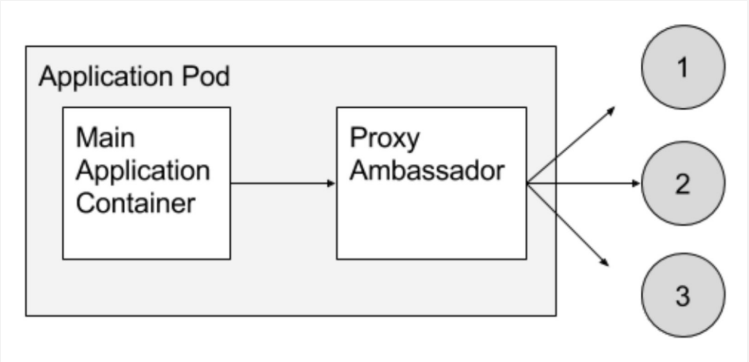
Pods

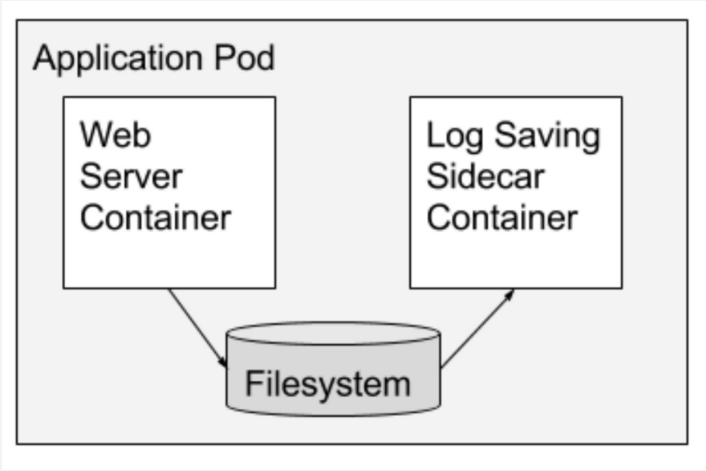
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
  - name: php-fpm
    image: php-fpm:1.12
    resources:
      limits:
        memory: "200Mi"
      requests:
        memory: "100Mi"
```

```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
```

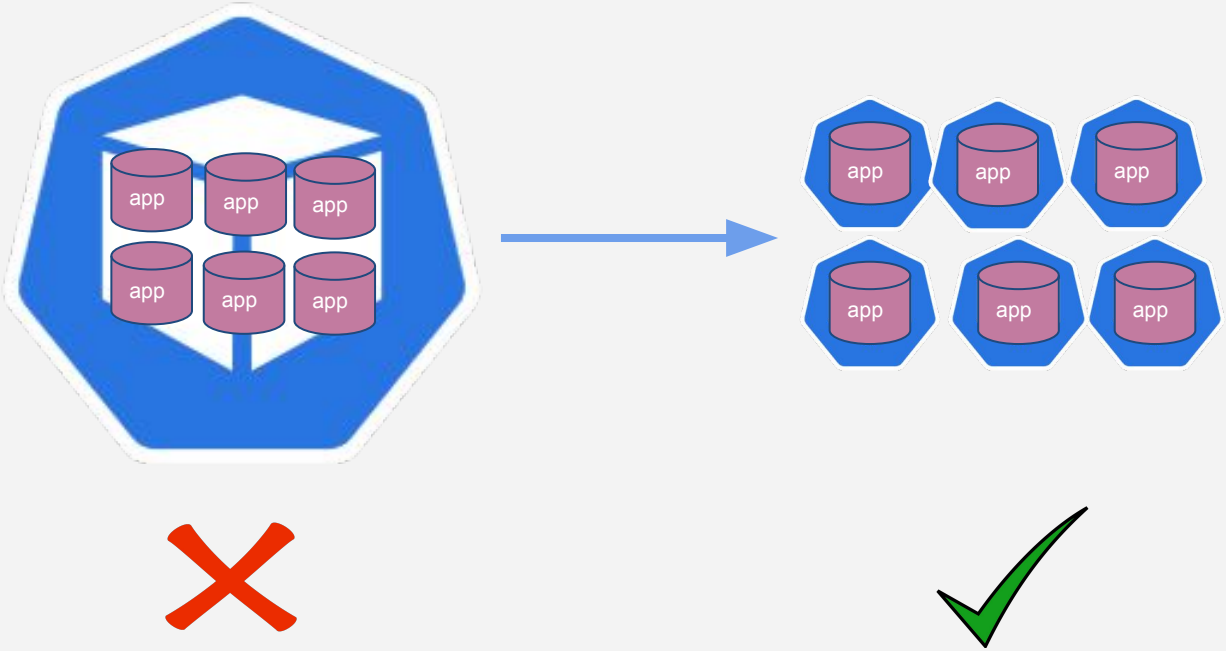
Multi Container Pod



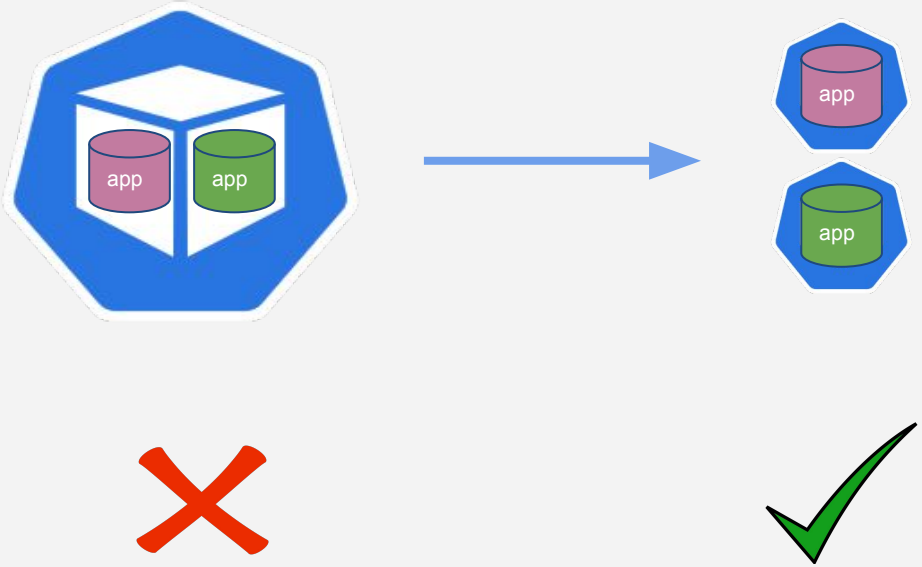




Multi Container Pod



Multi Container Pod



Module 6

Deployments

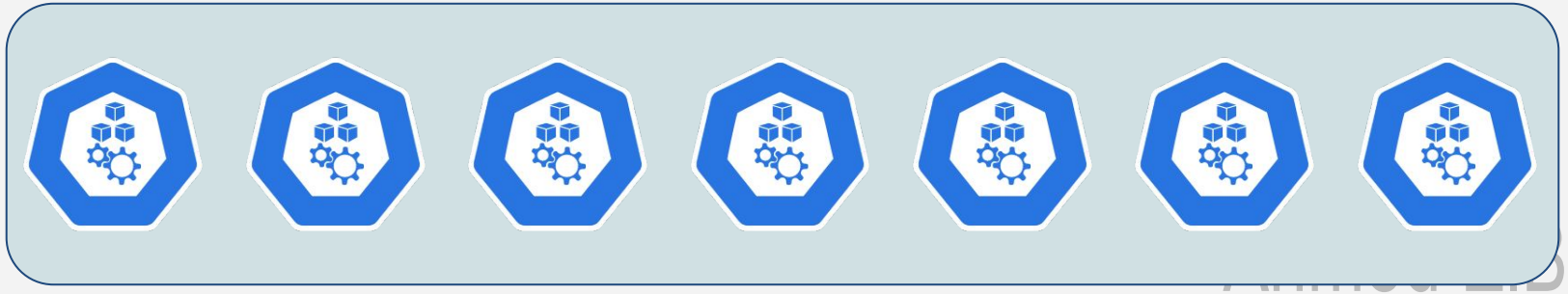
Deployments

What is Kubernetes Deployment

Managing The Lifecycle of The Application:

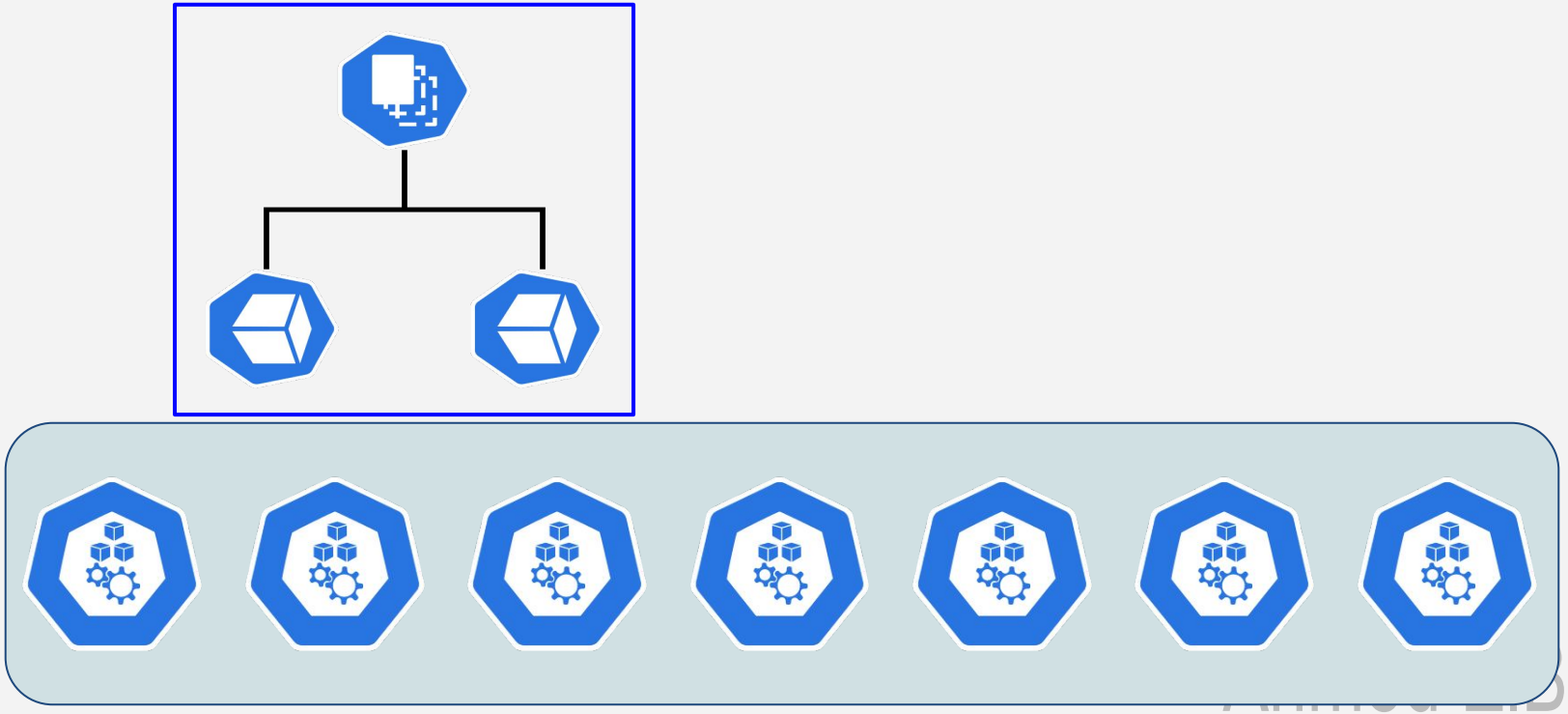
- Create the Pods
- Rollout new version of the application
- Rollback to previous version
- Scale the application
- Self healing

Deployments

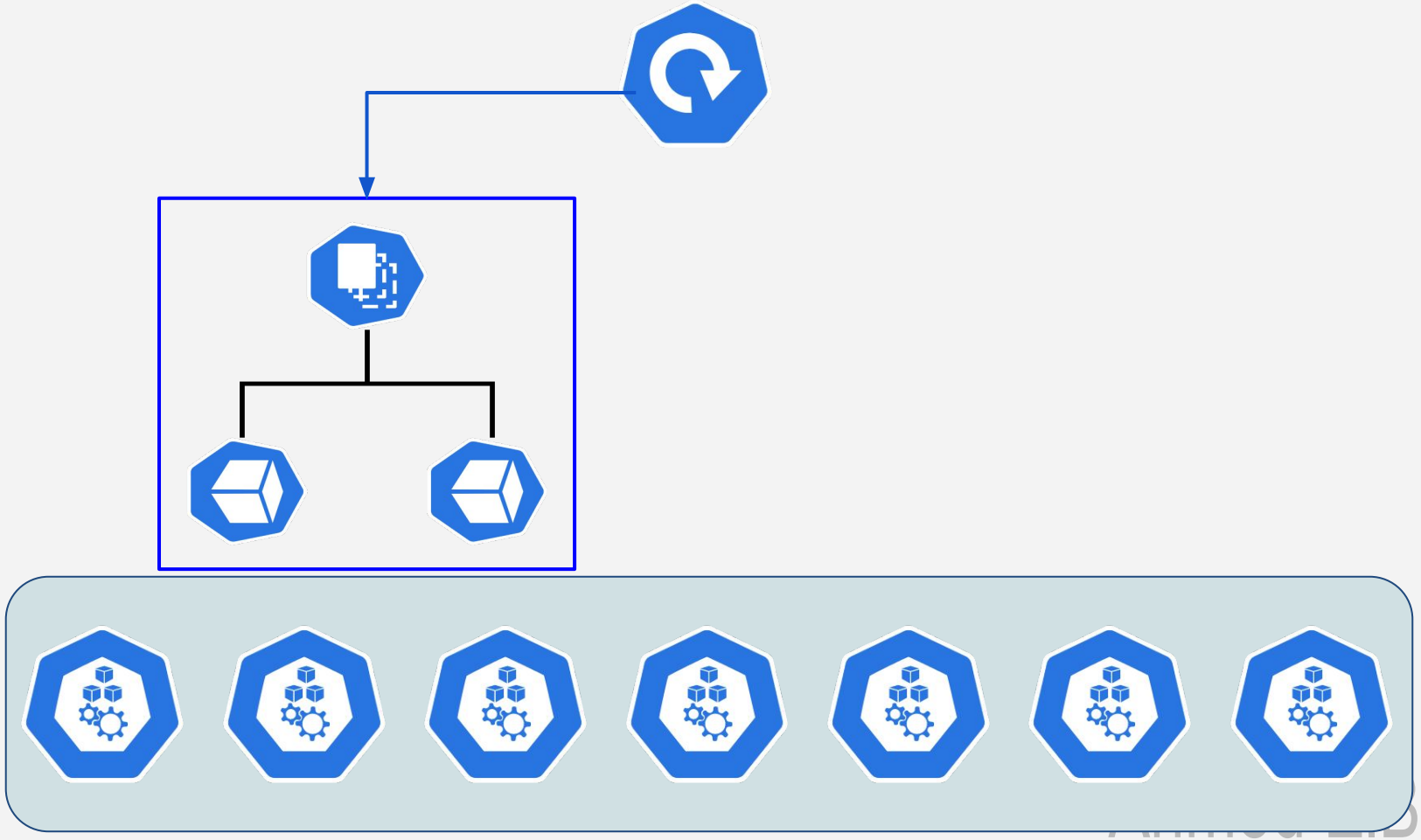


Deployments

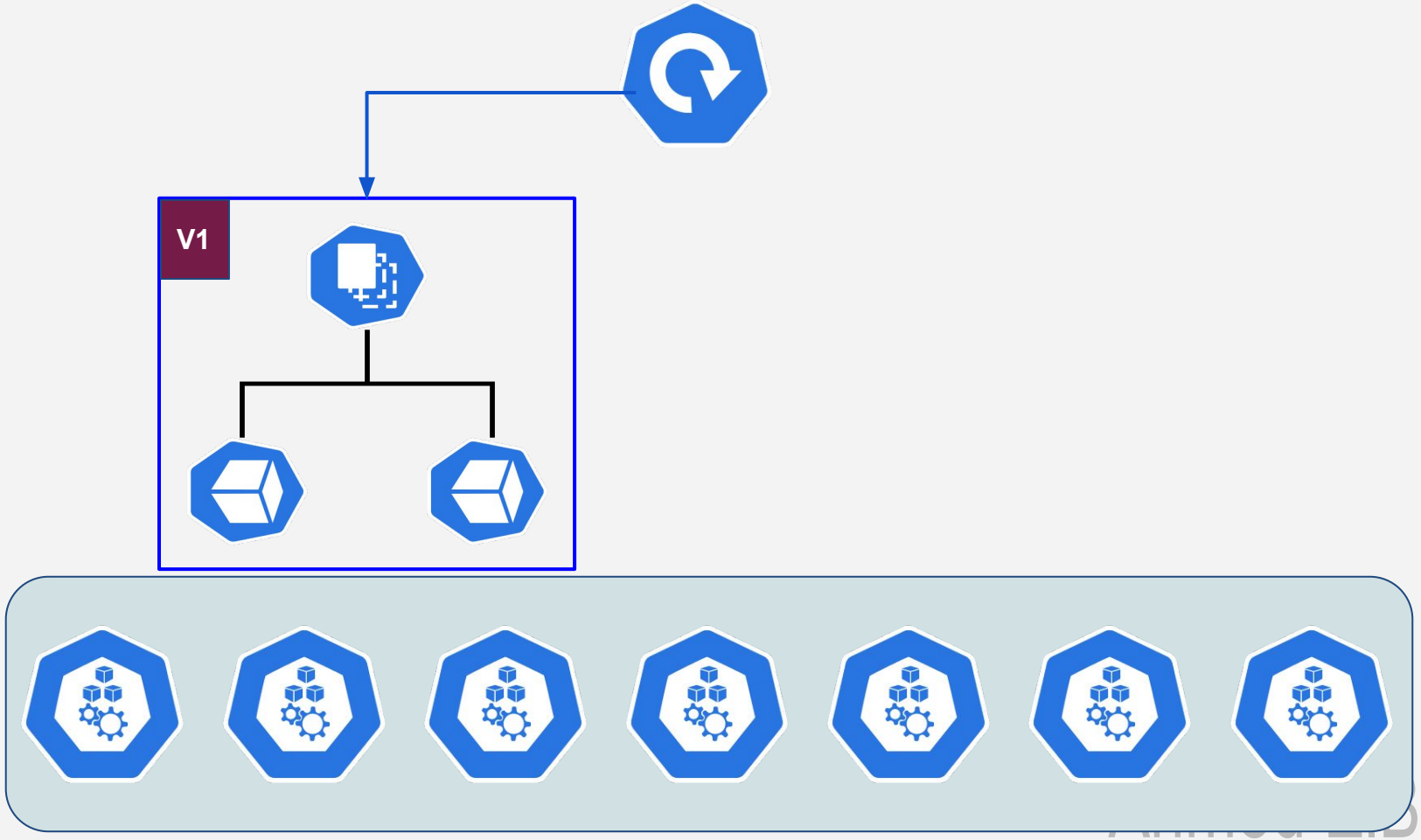
Deployments



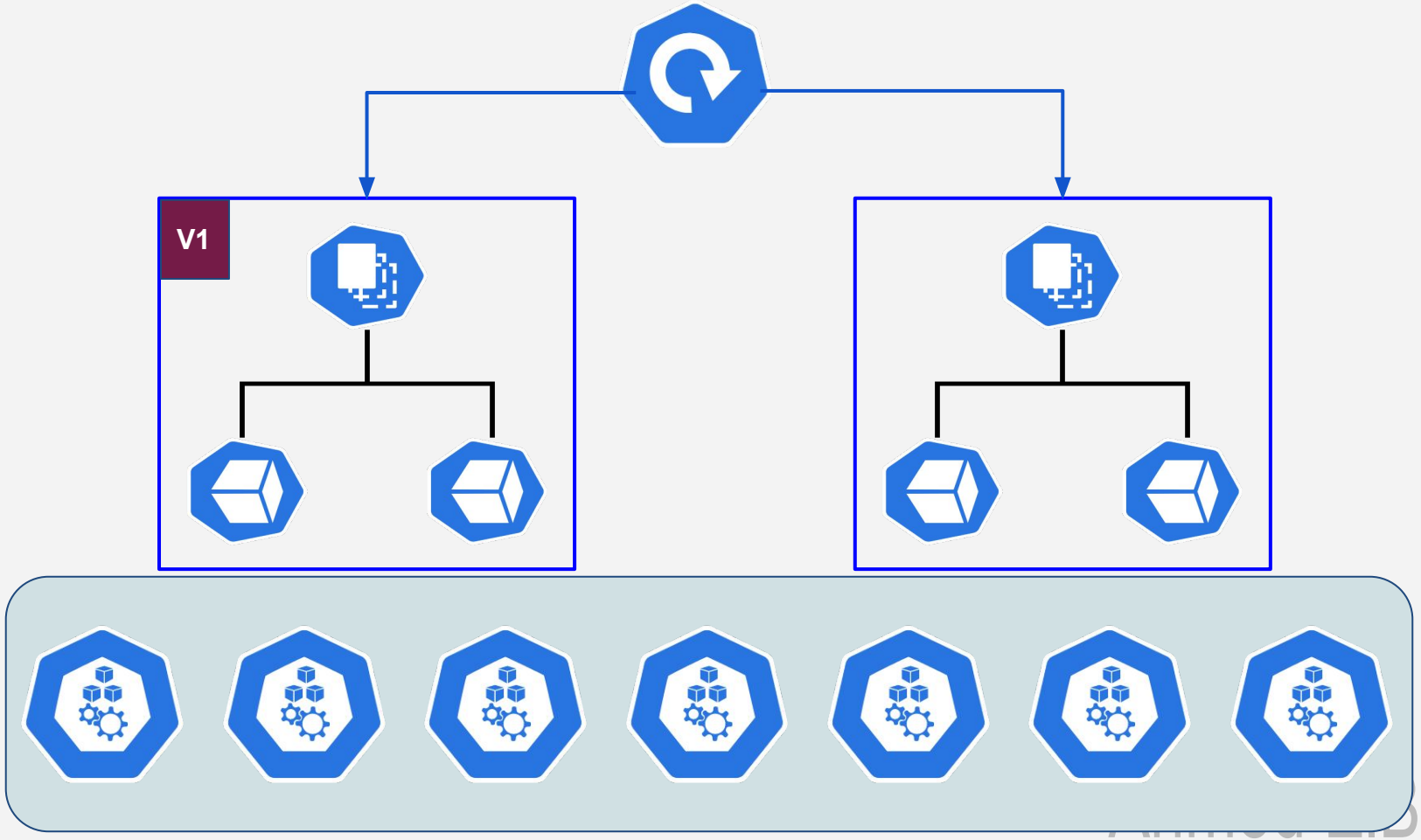
Deployments



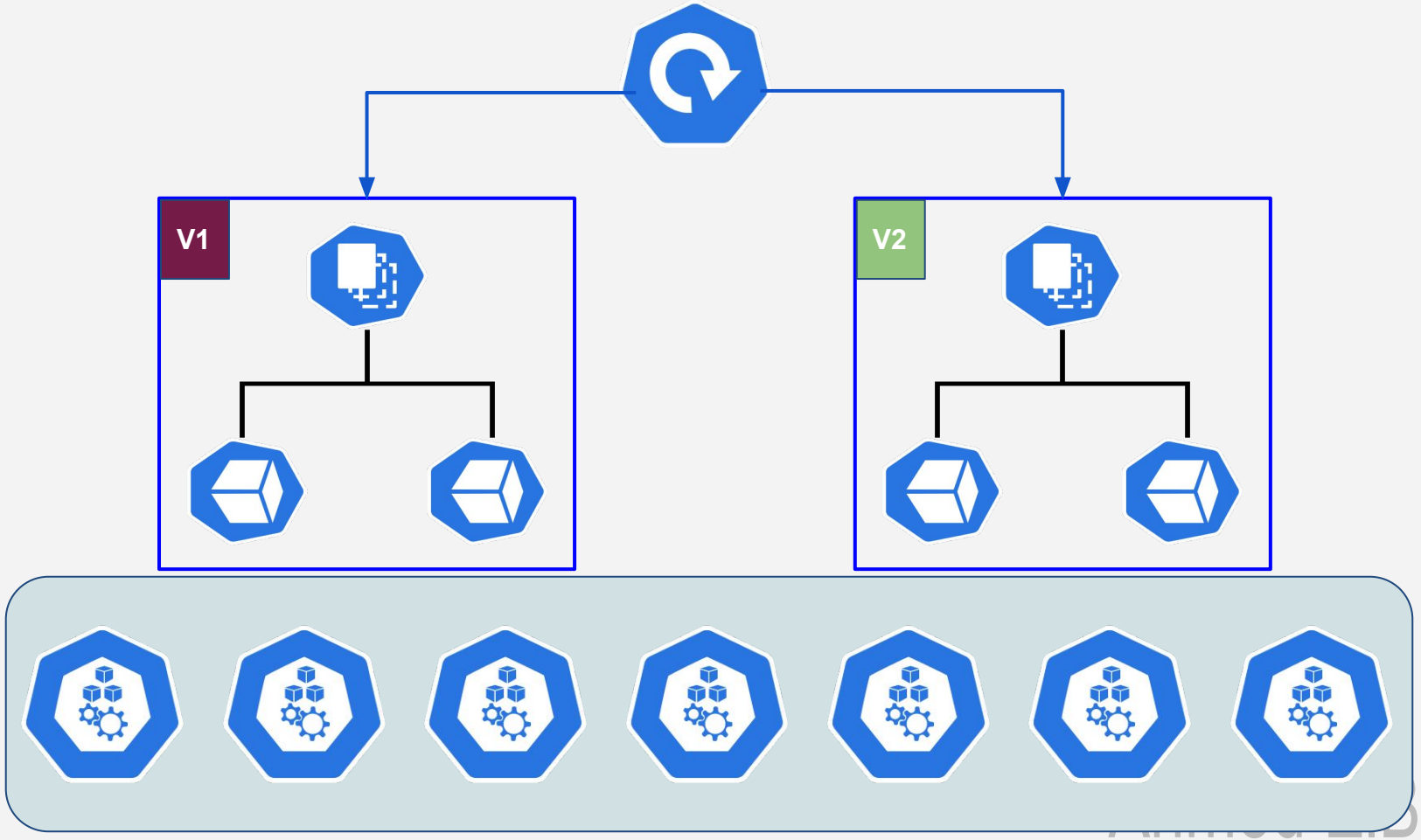
Deployments



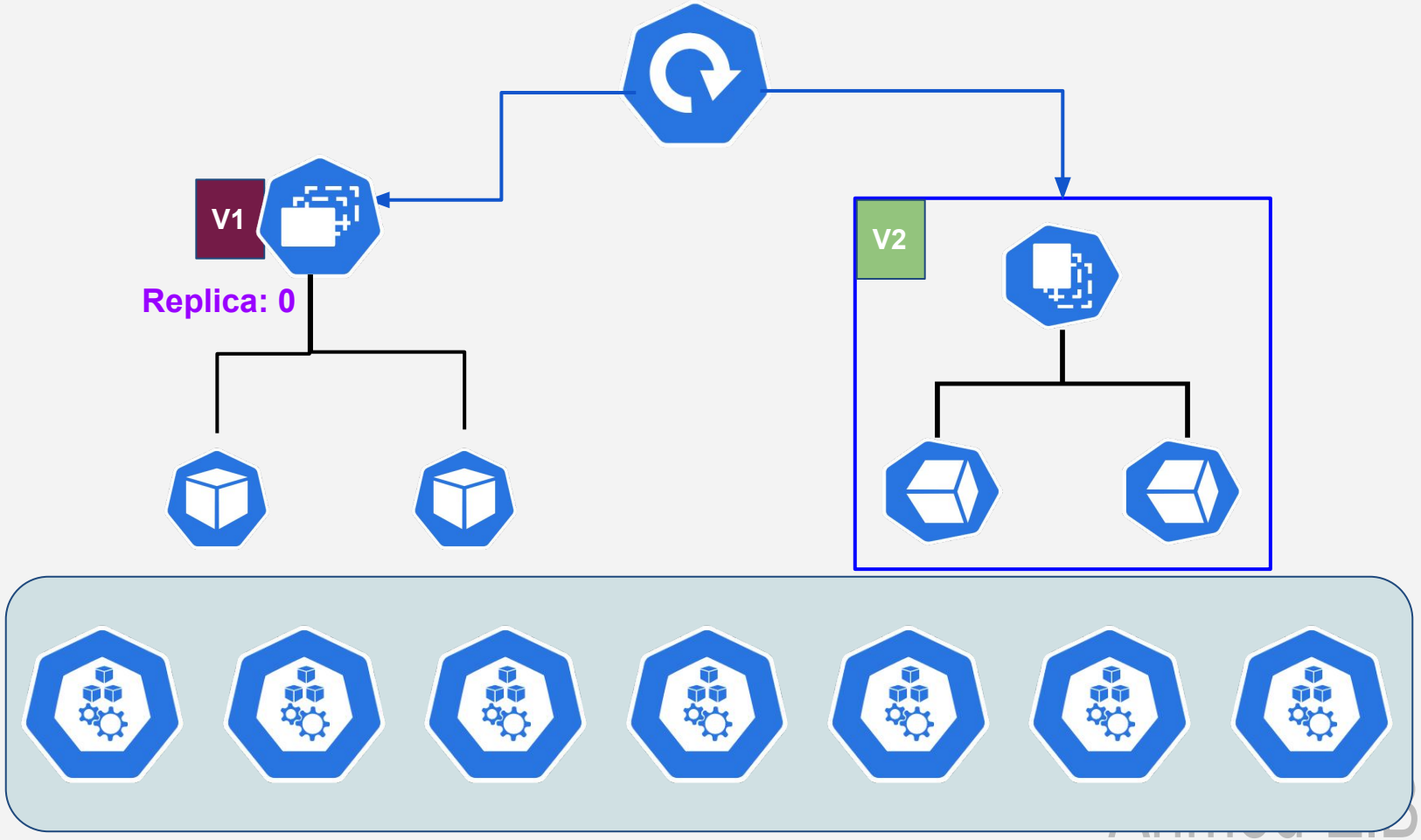
Deployments



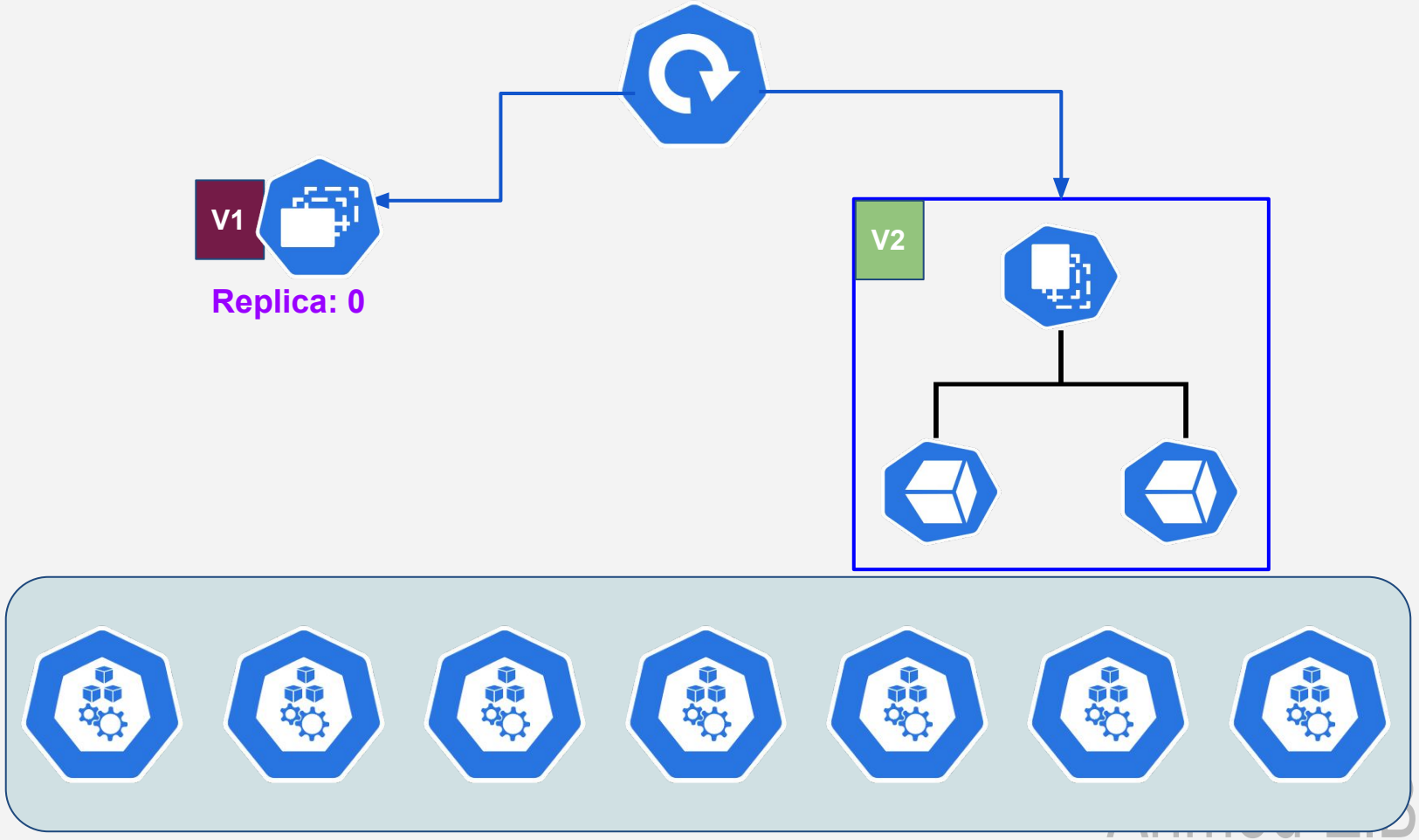
Deployments



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

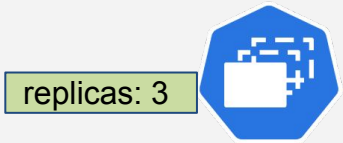
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2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

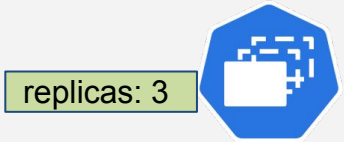
Deployments

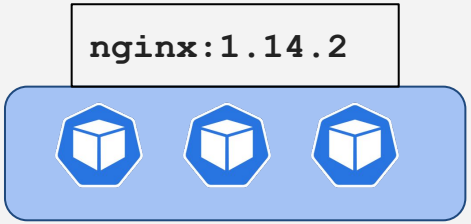
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2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



Deployments

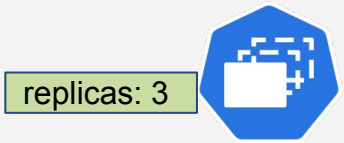
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2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

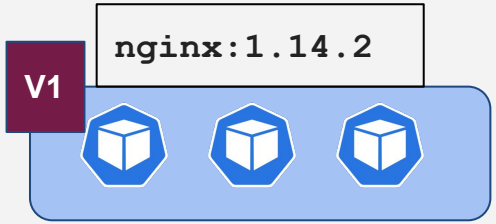




Deployments

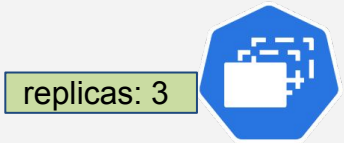
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2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

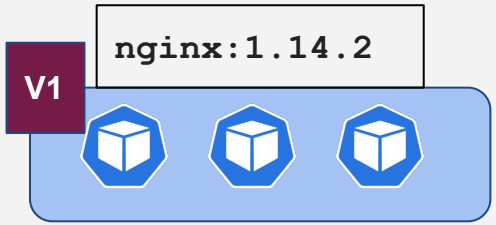




Deployments

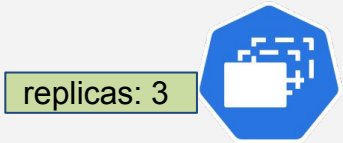
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.15.0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

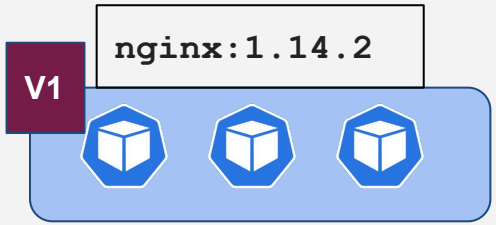


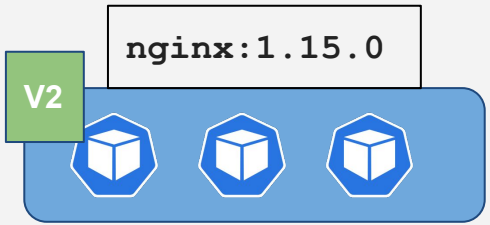


Deployments

```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:1.15.0
        ports:
        - containerPort: 80
```

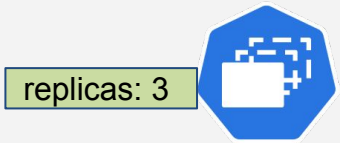


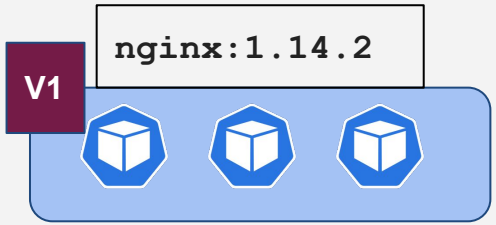


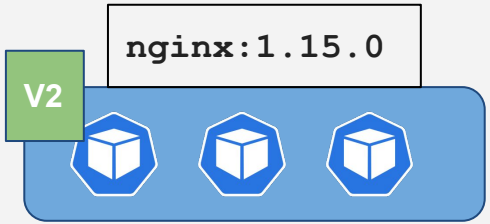


Deployments

```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:1.15.0
        ports:
        - containerPort: 80
```

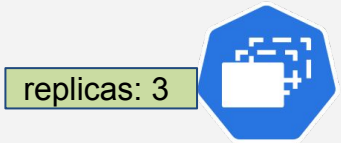


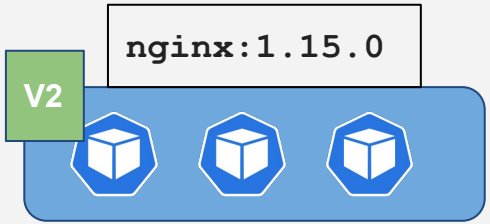




Deployments

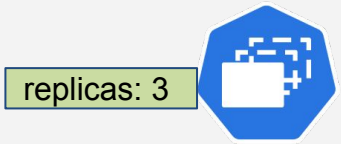
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.15.0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



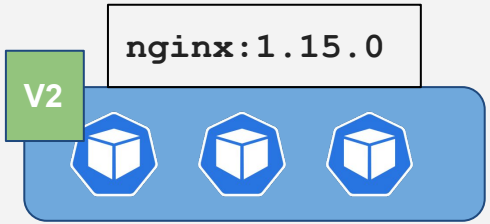


Deployments

```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:1.15.0
        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

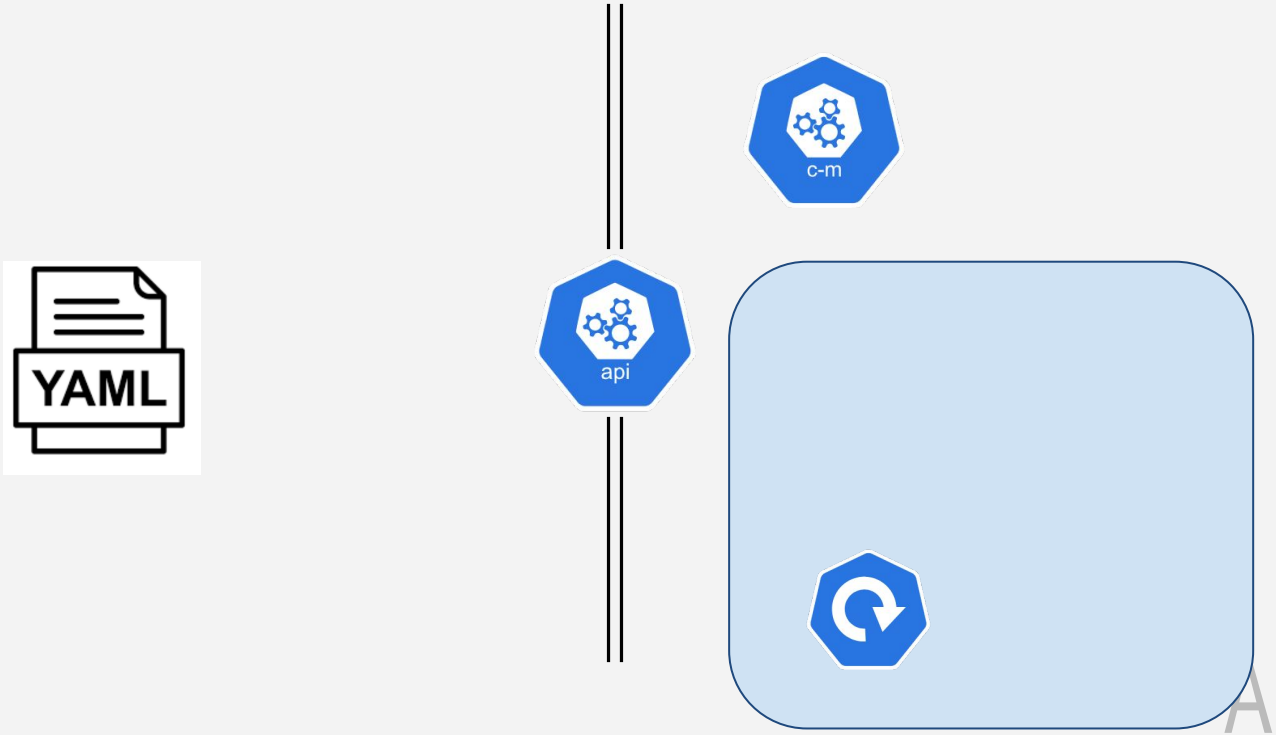
(⎈ |kind-c11:default)-> kubectl create -f deploy.yaml

Imperative

(⎈ |kind-c11:default)-> kubectl create deployment nginx-deployment
--image=nginx:1.14.2
--replicas=3
--port=80

Deployments

Deployment Object vs Deployment Controller



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/


DEMO
Ahmed ElBakry

Deployment Strategies

Deployment Strategies

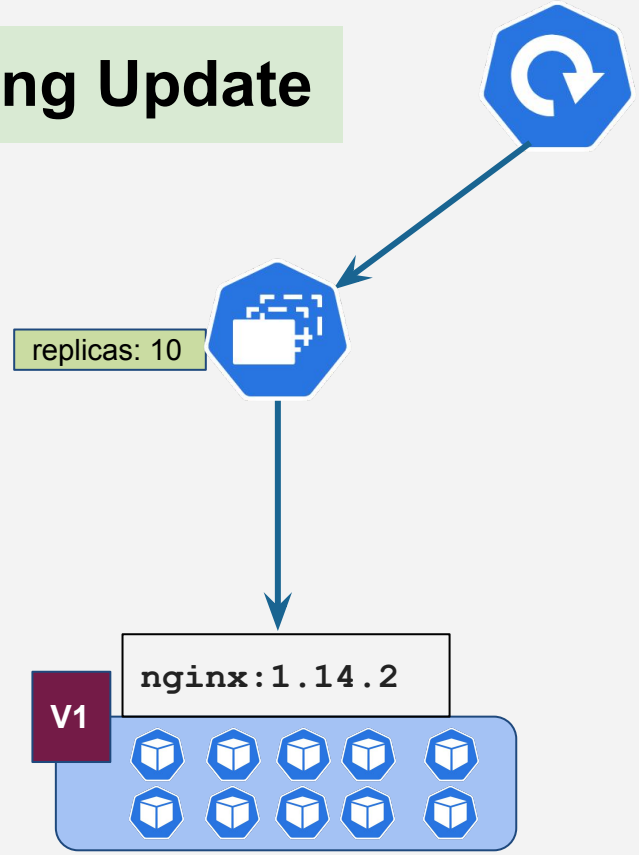
Rolling Update

```yaml
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:
        - containerPort: 80
```

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
```

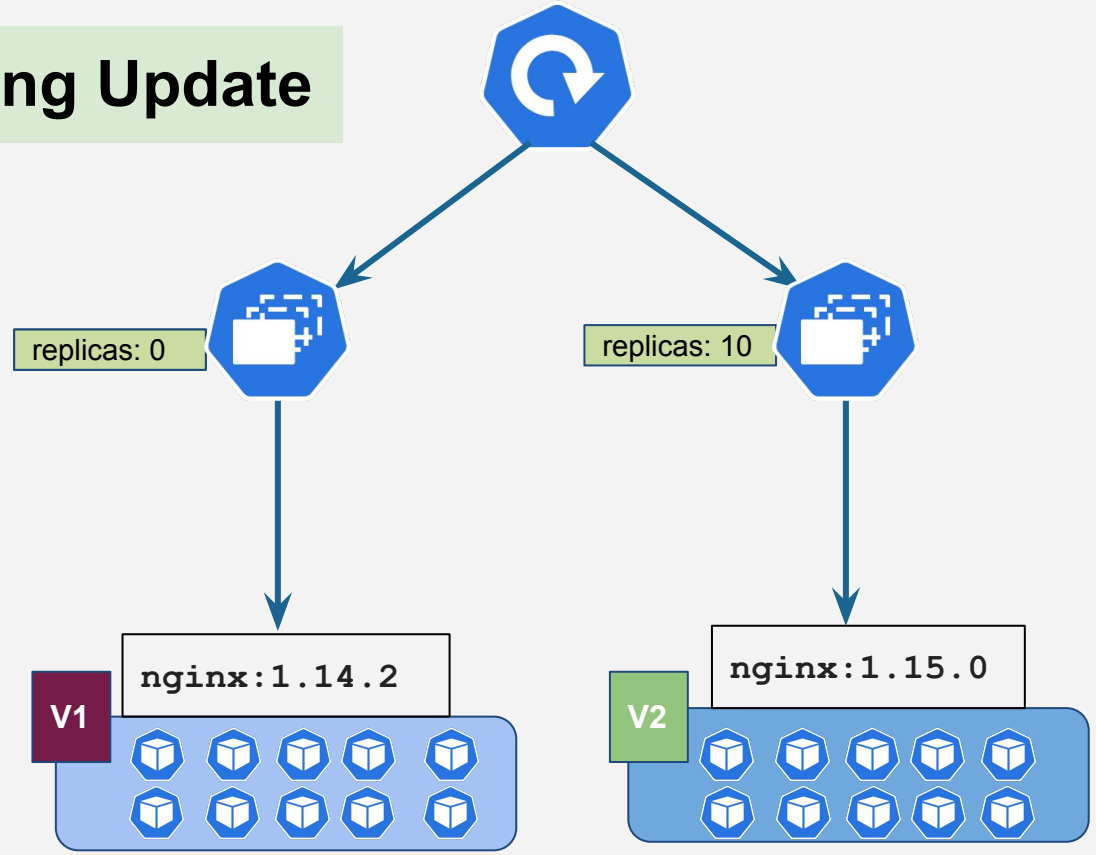
Rolling Update



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
```

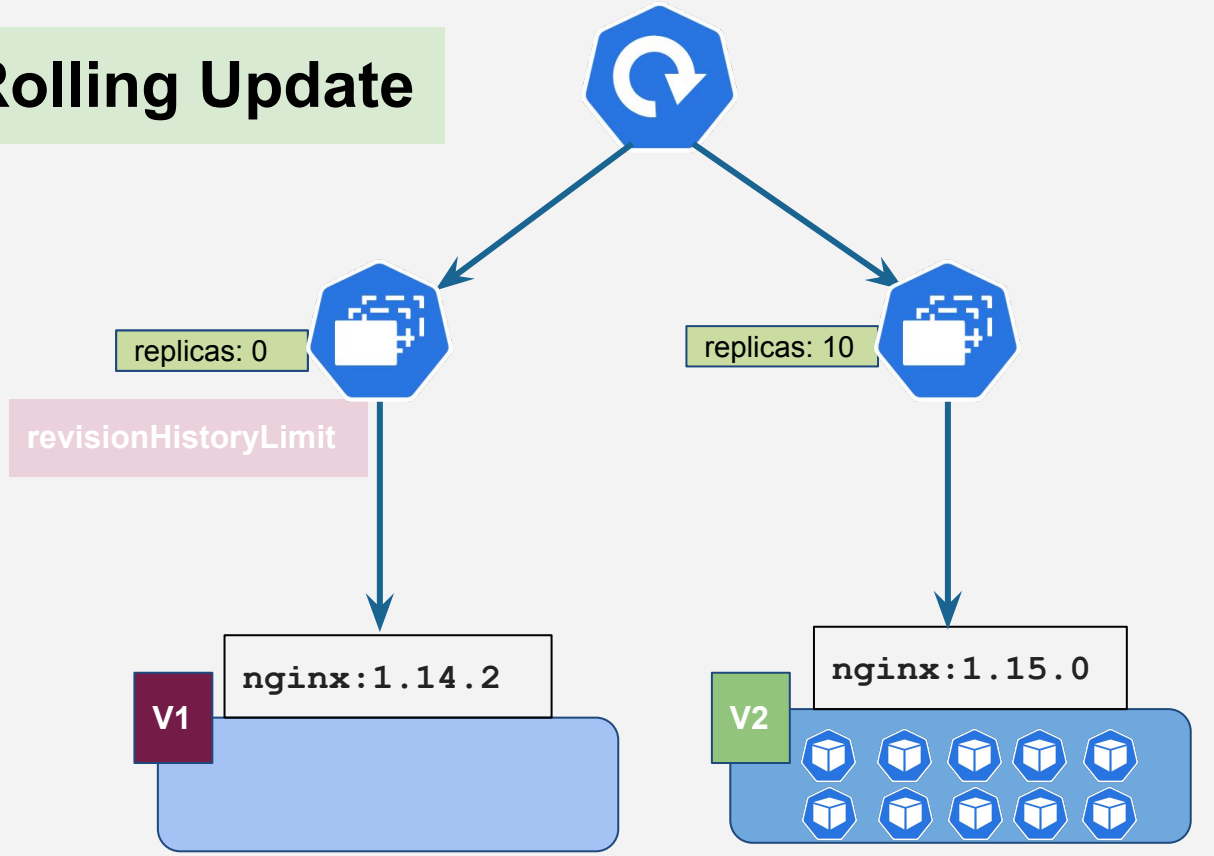
Rolling Update



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
```

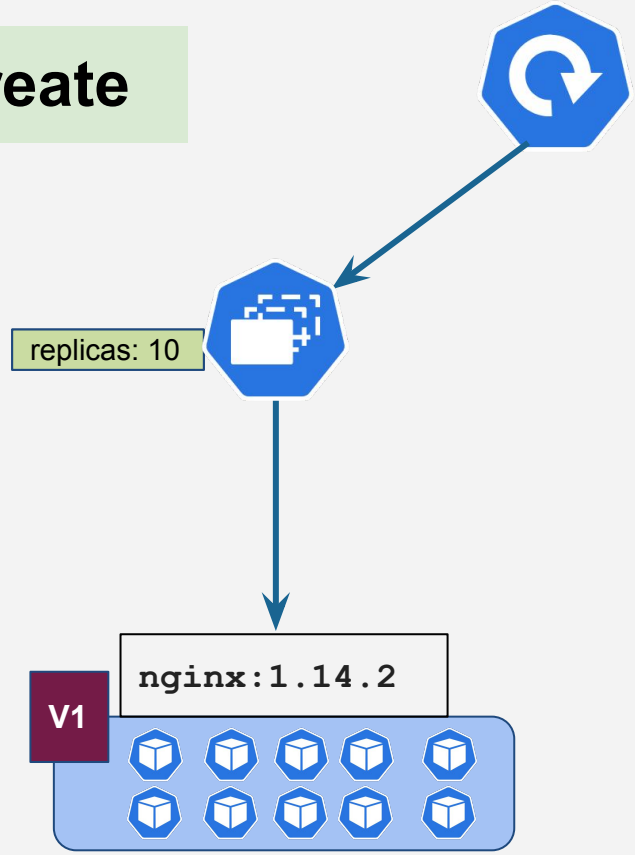
Rolling Update



Deployment Strategies

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

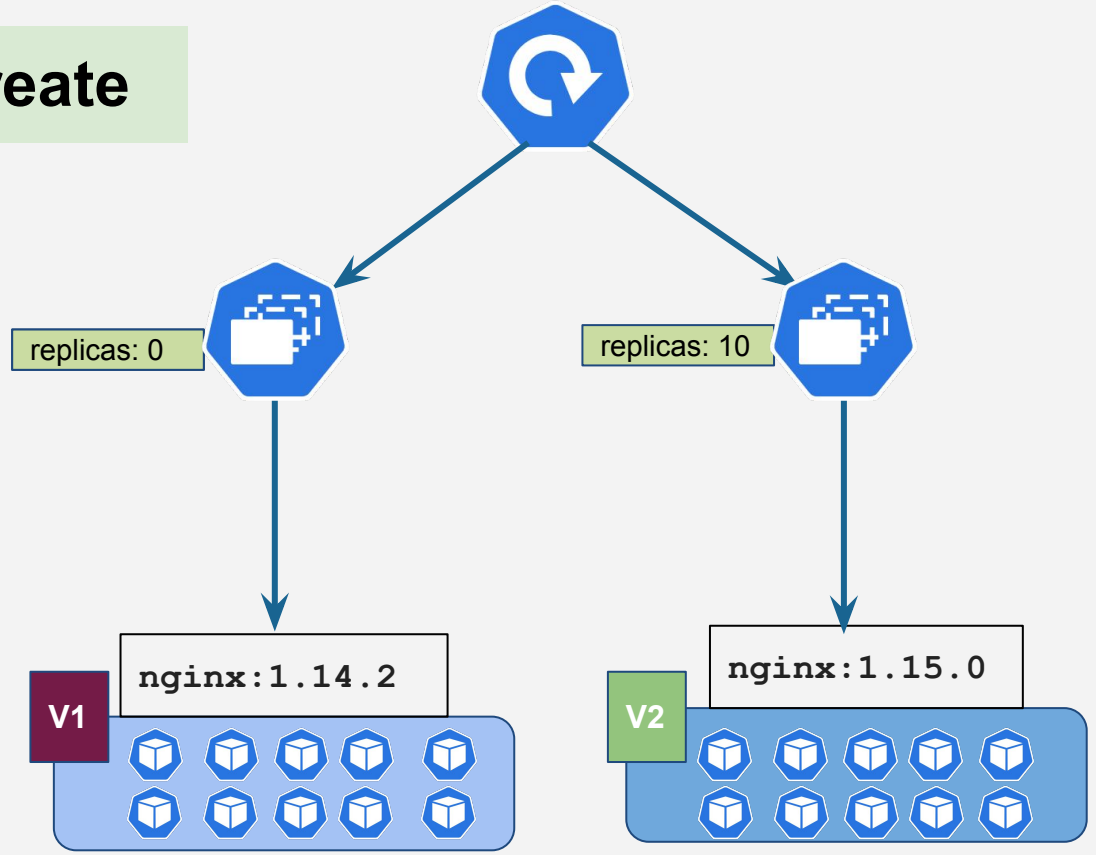
Recreate



Deployment Strategies

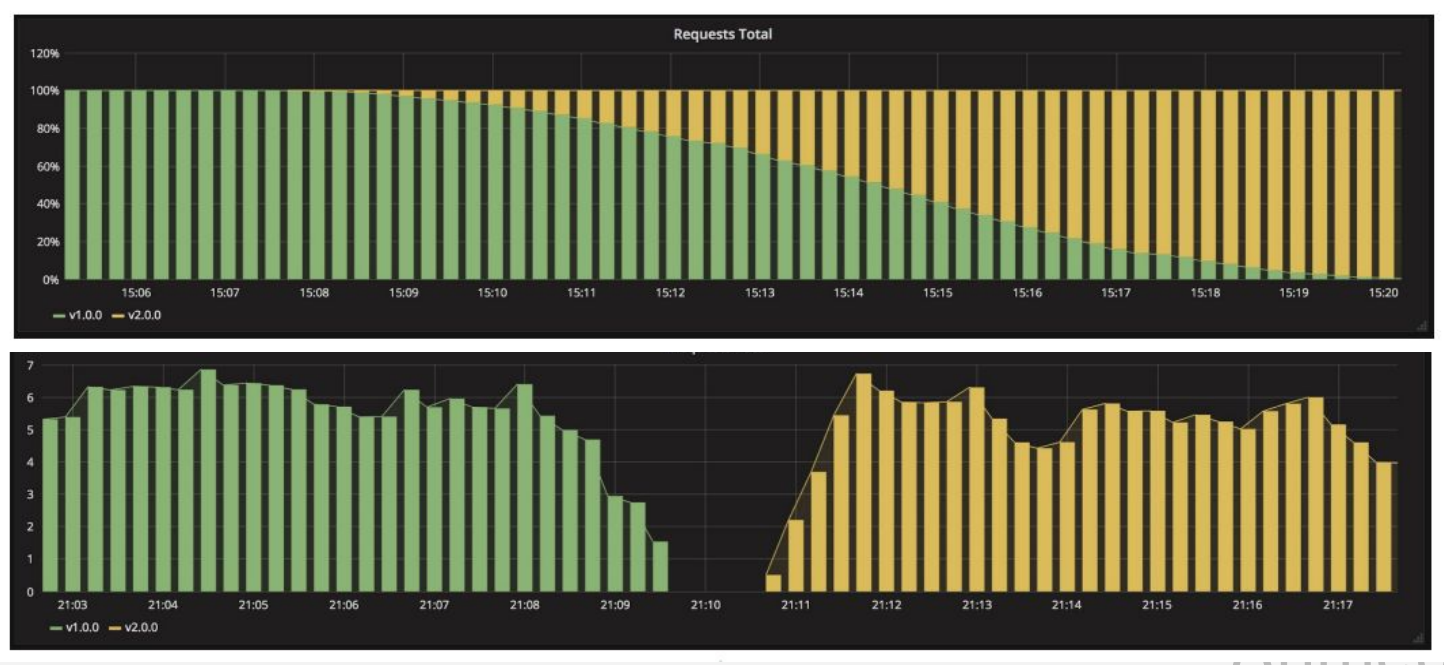
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 10
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: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

Recreate



Deployment Strategies

Traffic Pattern



Module 7

Services

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.

Services

01

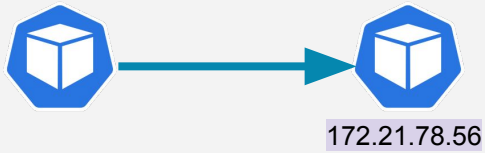
Service Discovery

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.

Services

01

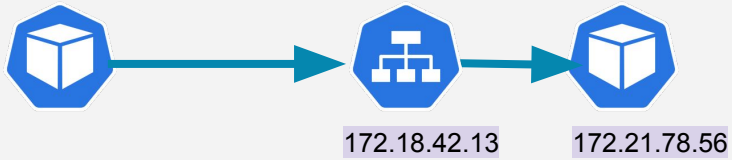
Service Discovery

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

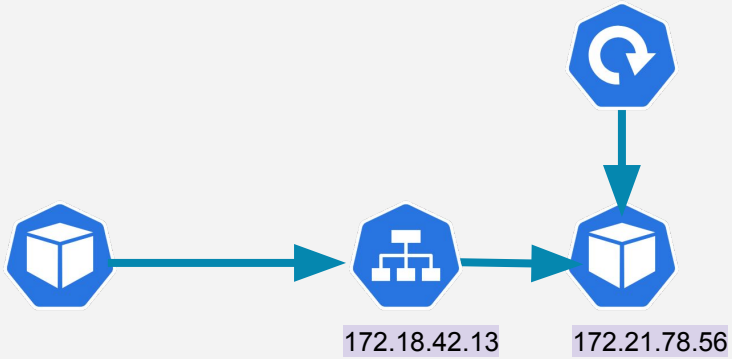
Service Discovery

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.



Services

01

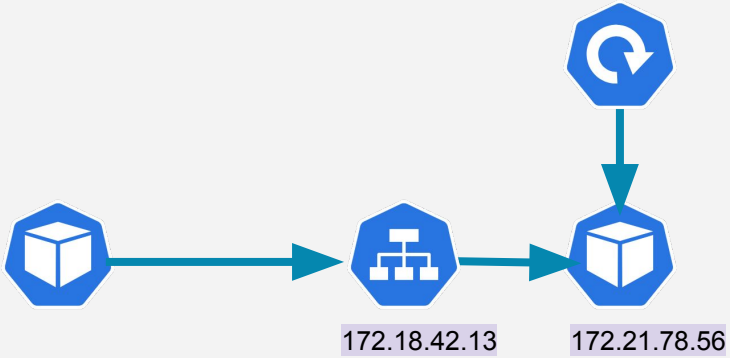
Service Discovery

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.

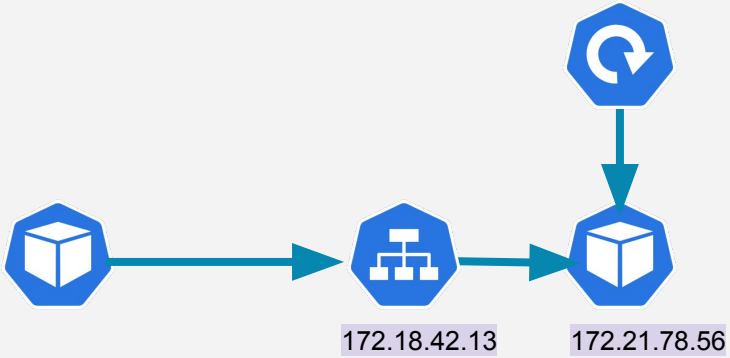


Services

01

Service Discovery

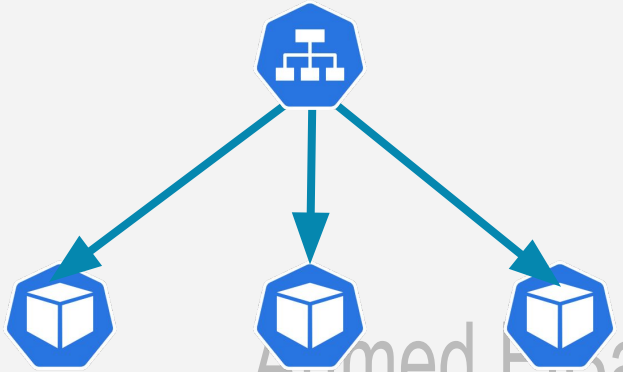
A way to connect different services or components with a single stable endpoint



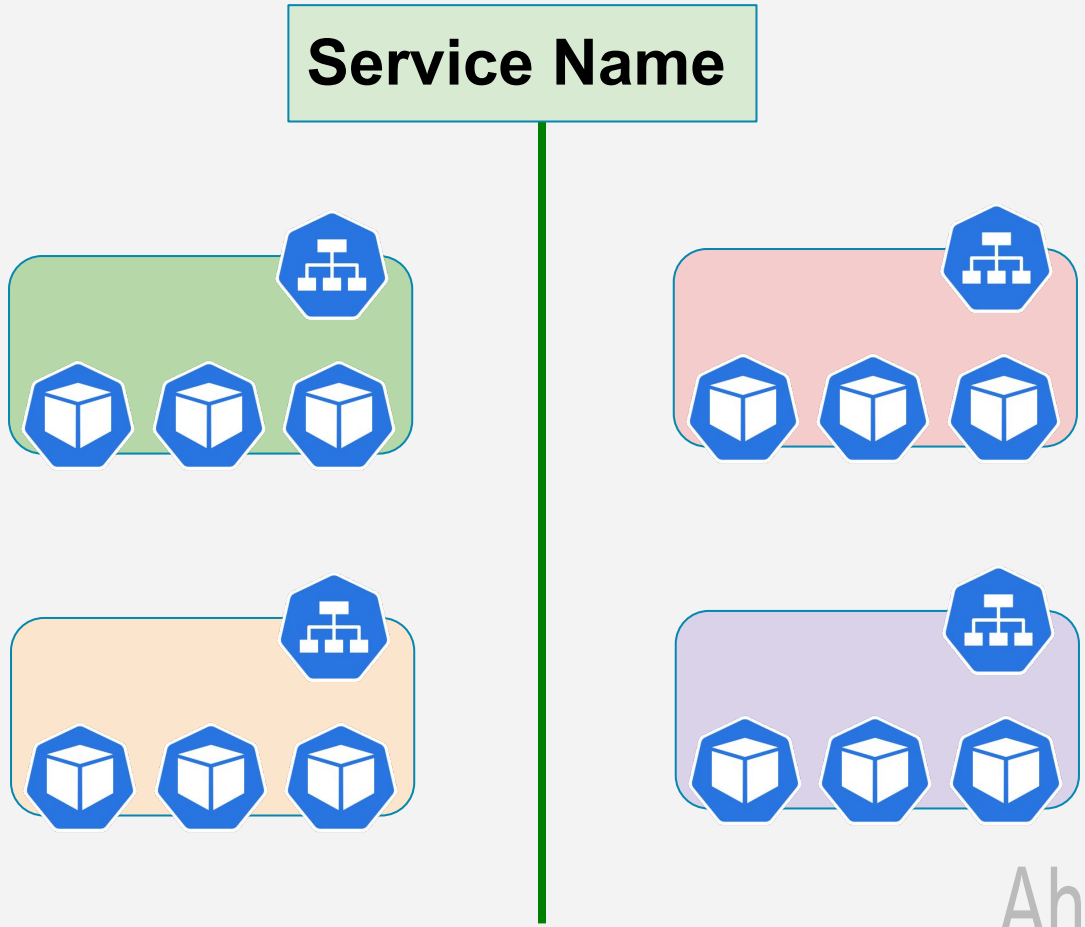
02

Load Balancing

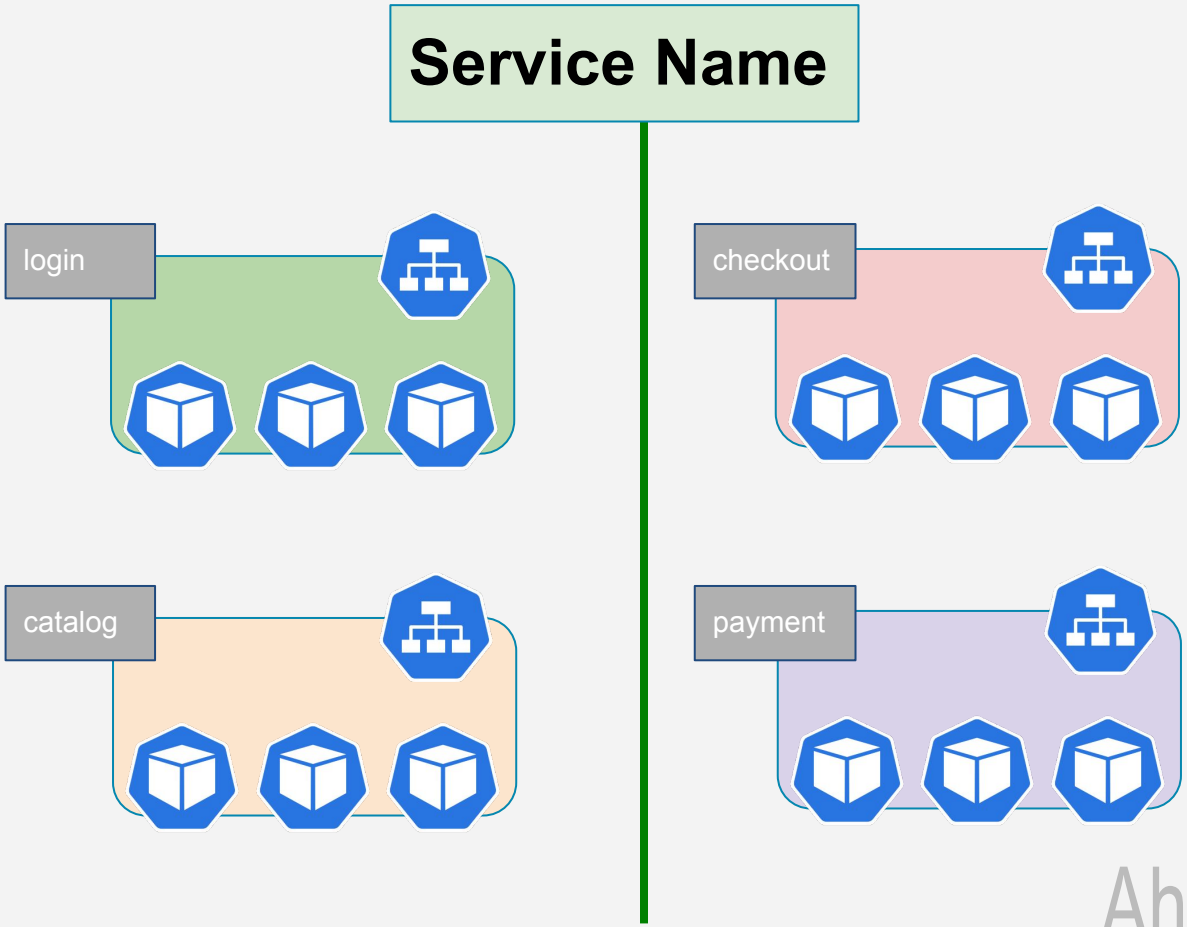
Kubernetes automatically distributes traffic between different application backends to optimize performance and availability.



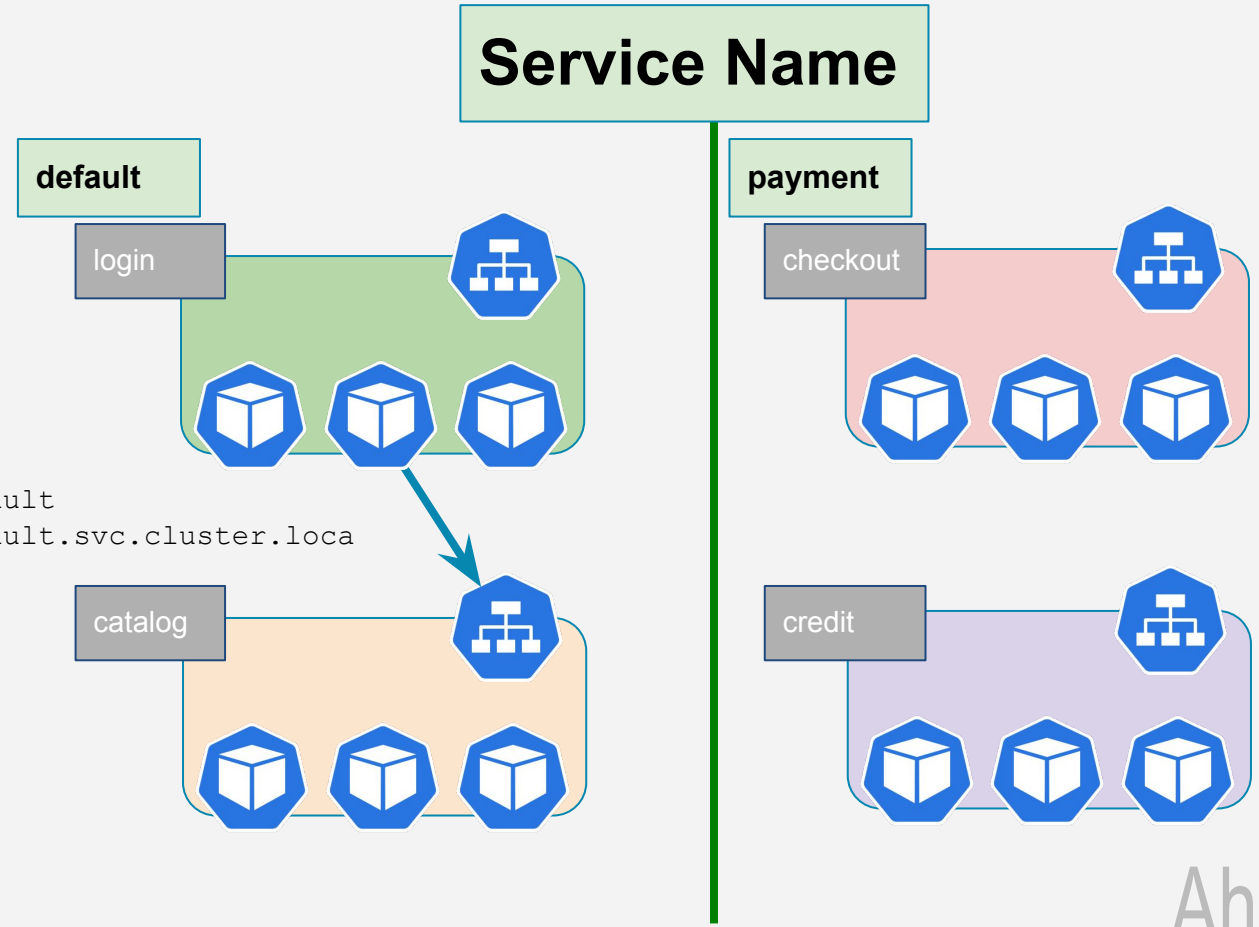
Services



Services

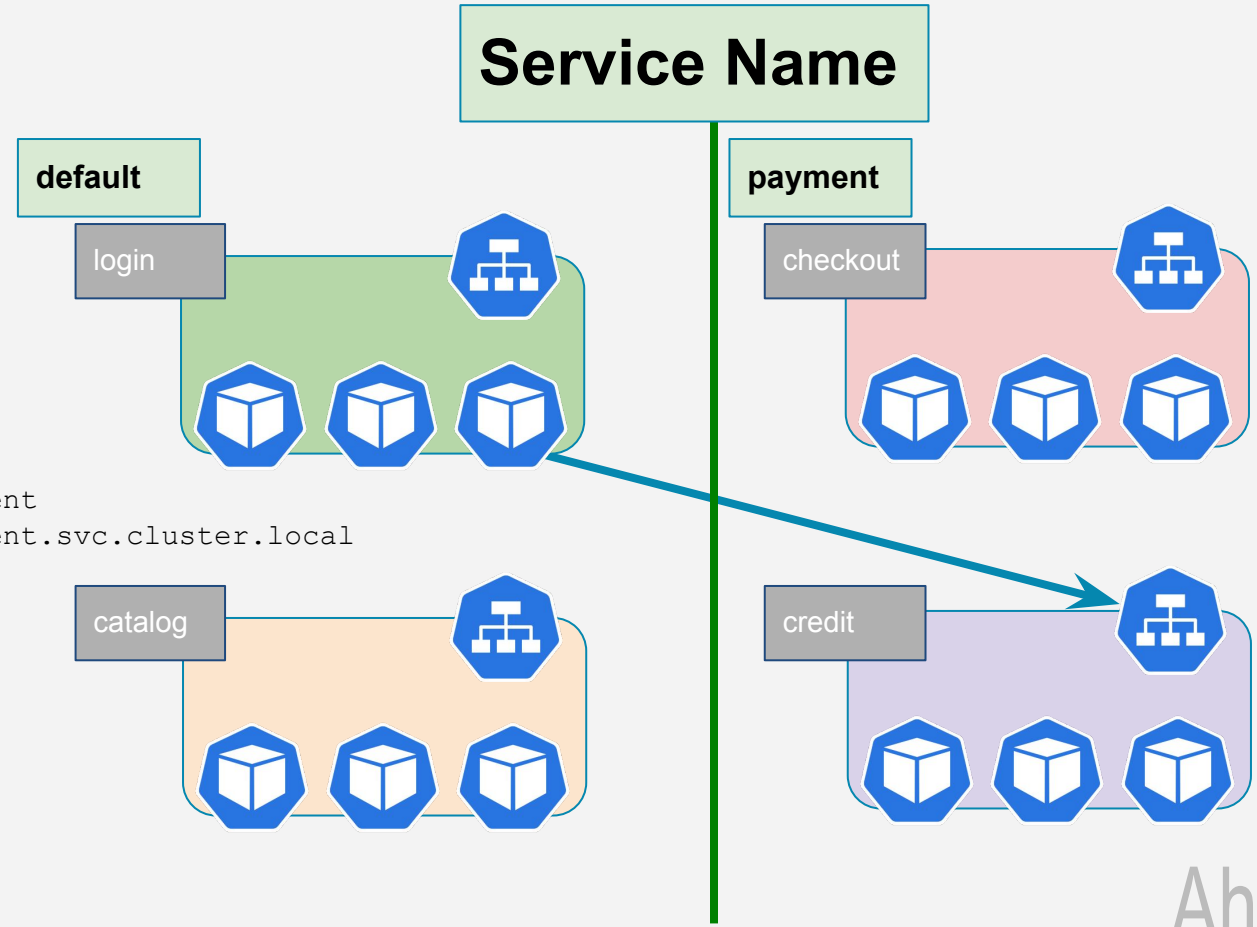


Services




```
catalog
catalog.default
catalog.default.svc.cluster.loca
l
```

Services




```
credit   x
credit.payment
credit.payment.svc.cluster.local
```

Services

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

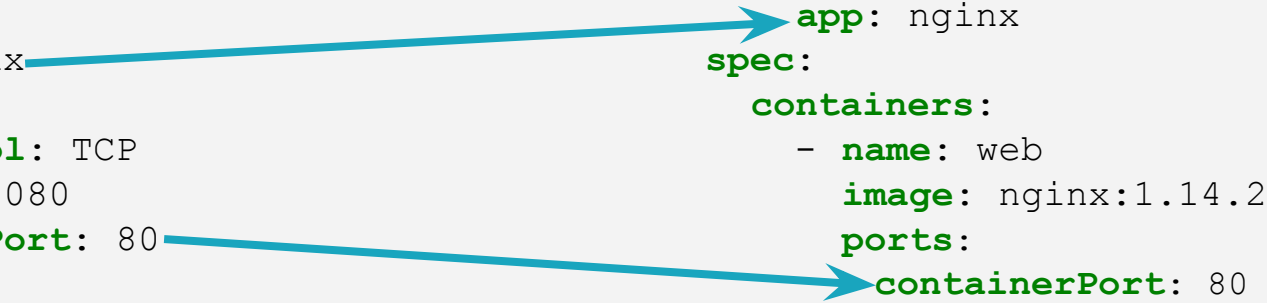
```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 80
```

```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:
  type: ClusterIP
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```

Services

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

Services

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

Services

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
Spec:
  clusterIP: None
  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: 80
      targetPort: 9376
```

Services

```yaml
apiVersion: v1
kind: Service
metadata:
  name: test-svc
spec:
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 9376
```

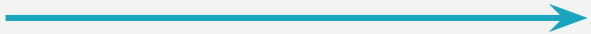
Services

```yaml
apiVersion: v1
kind: Service
metadata:
  name: test-svc
spec:
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 9376
```

```yaml
apiVersion: v1
kind: Endpoints
metadata:
  name: test-svc
subsets:
  - addresses:
      - ip: 10.96.87.20
    ports:
      - port: 9376
```

Services

```yaml
apiVersion: v1
kind: Service
metadata:
  name: test-svc
spec:
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 9376
```



```yaml
apiVersion: v1
kind: Endpoints
metadata:
  name: test-svc
subsets:
  - addresses:
      - ip: 10.96.87.20
    ports:
      - port: 9376
```

Services

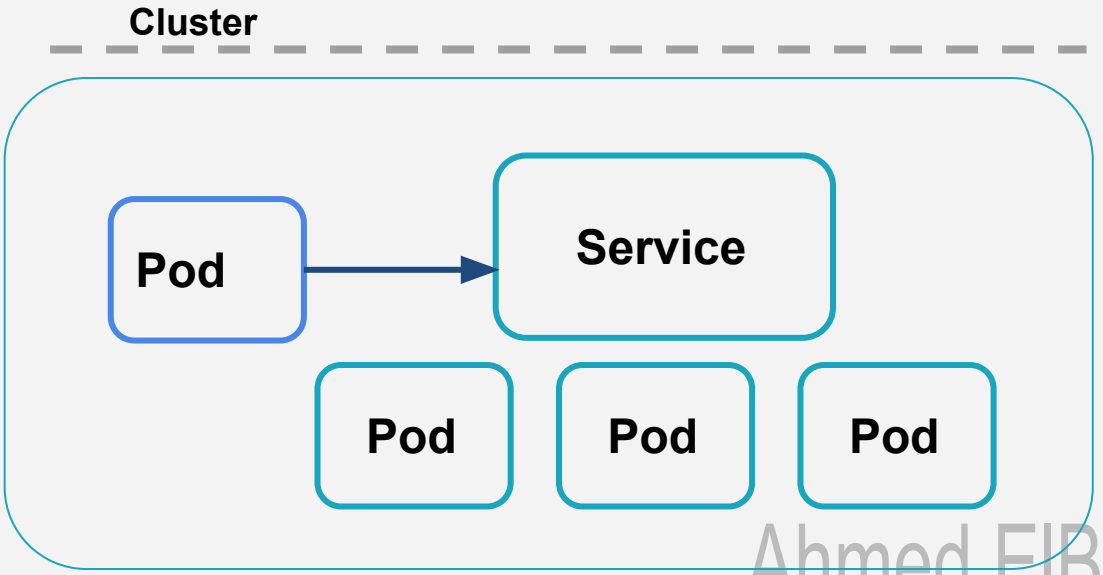
```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
Spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      name: http
      port: 80
      targetPort: 9376
    - name: https
      protocol: TCP
      port: 443
      targetPort: 9377
```

Service Types

- ClusterIP
- NodePort
- LoadBalancer
- Headless
- ExternalName
- ExternalIP

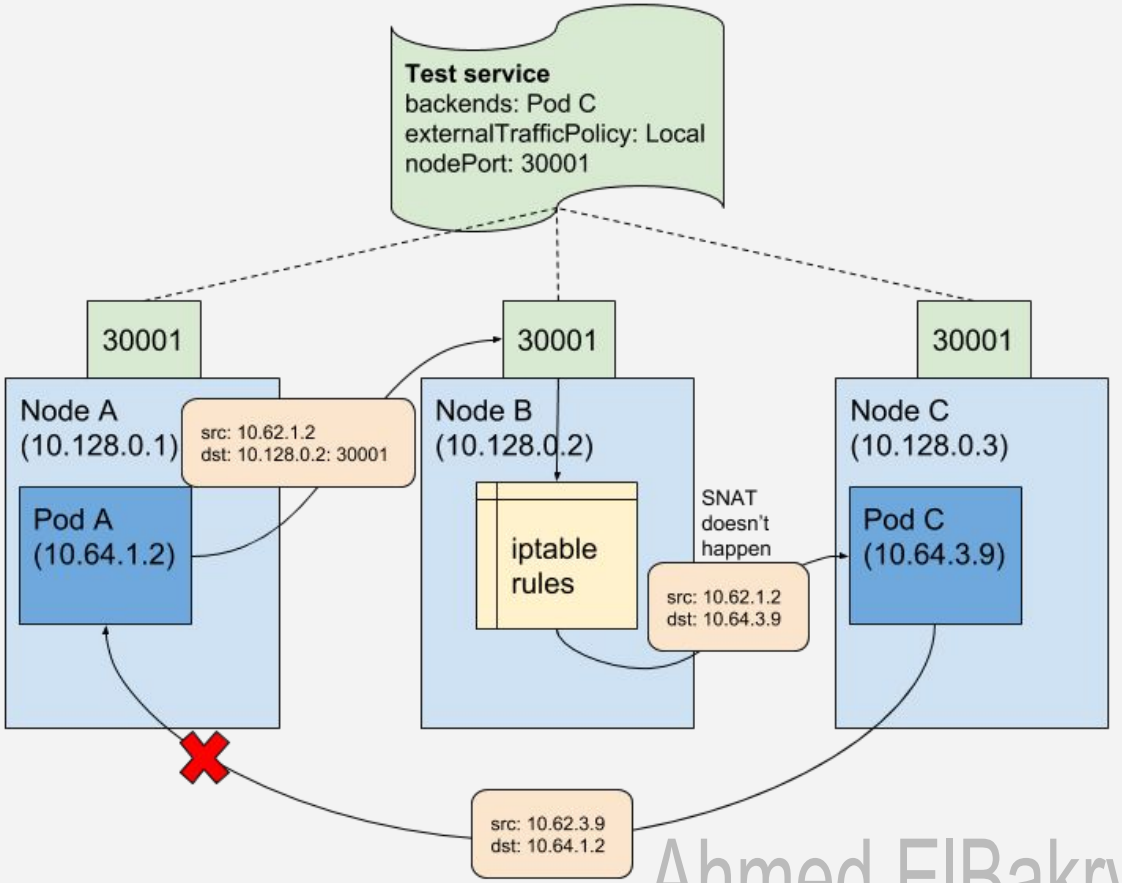
Services

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



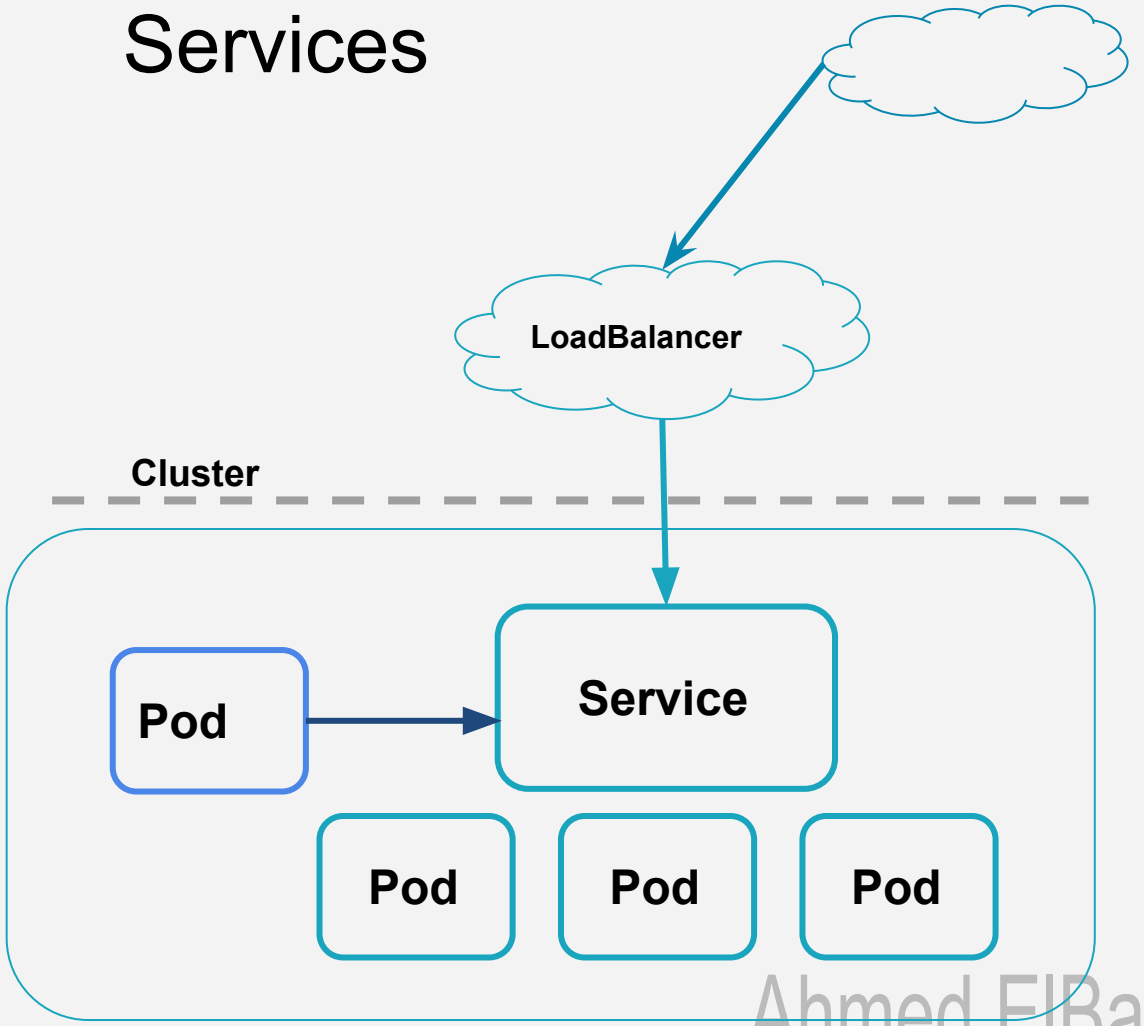
Services

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



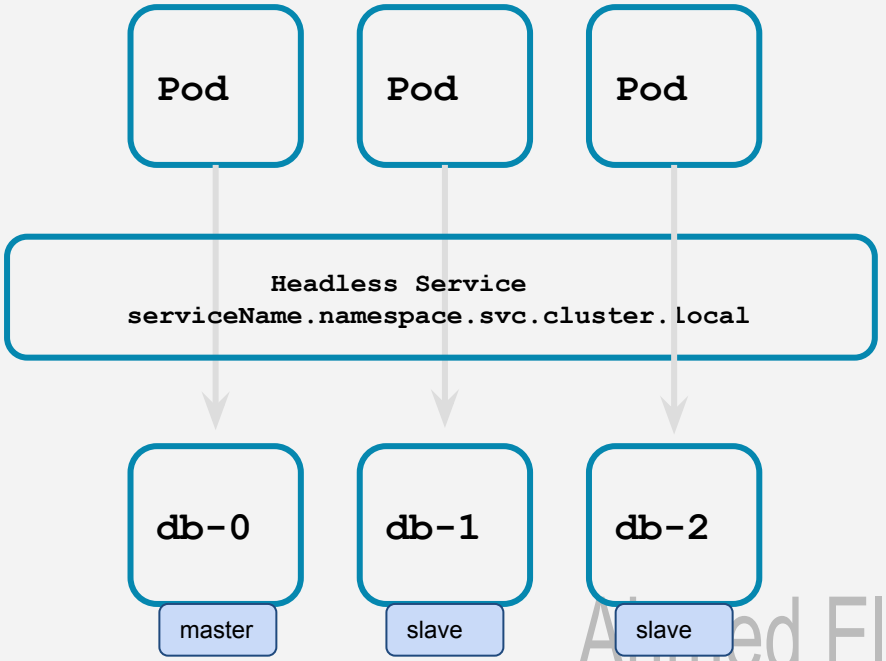
Services

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



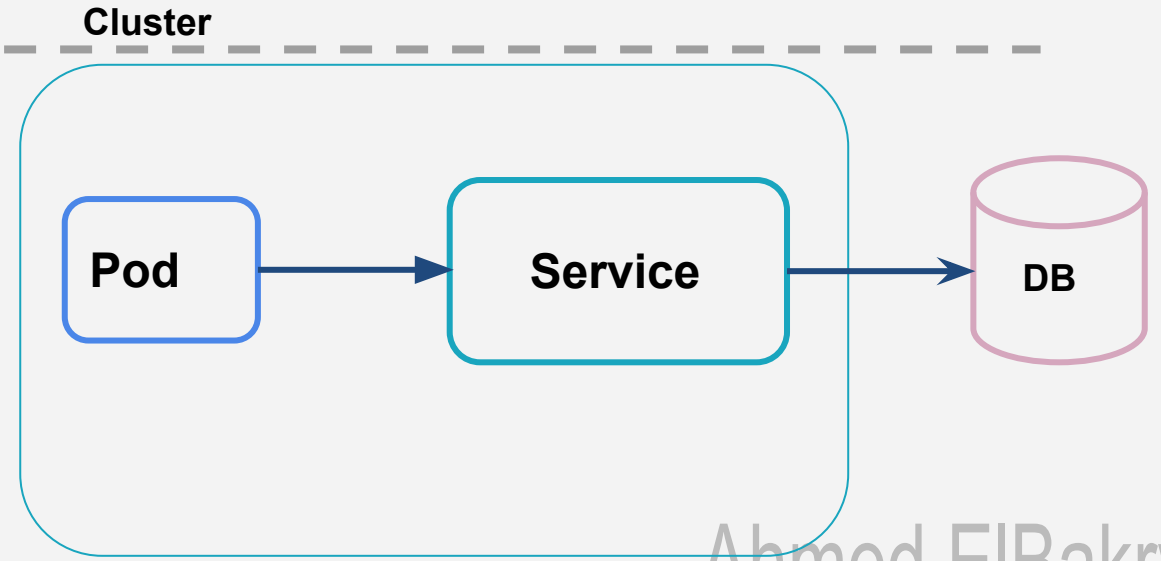
Services

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
Spec:
  clusterIP: None
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 3306
      targetPort: 3306
```



Services

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
Spec:
  type: ExternalName
  externalName:
mysql.external.domain.com
  ports:
    - protocol: TCP
      port: 3306
```

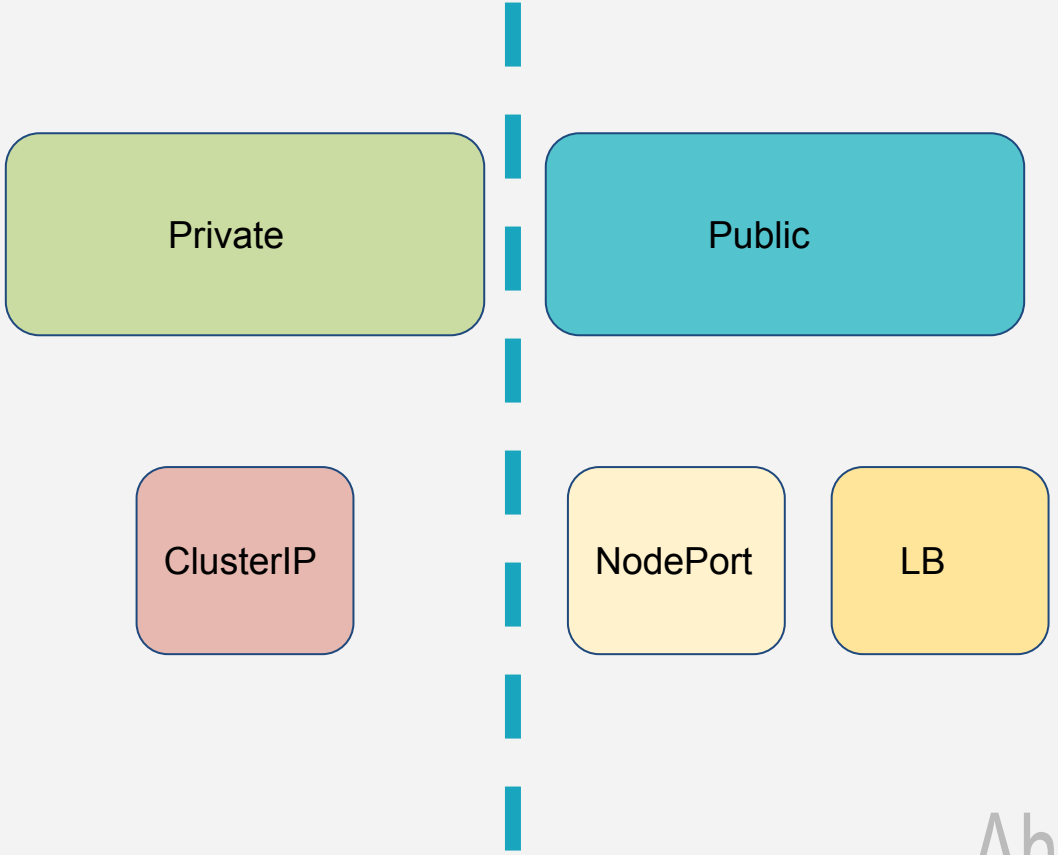



DEMO
Ahmed ElBakry

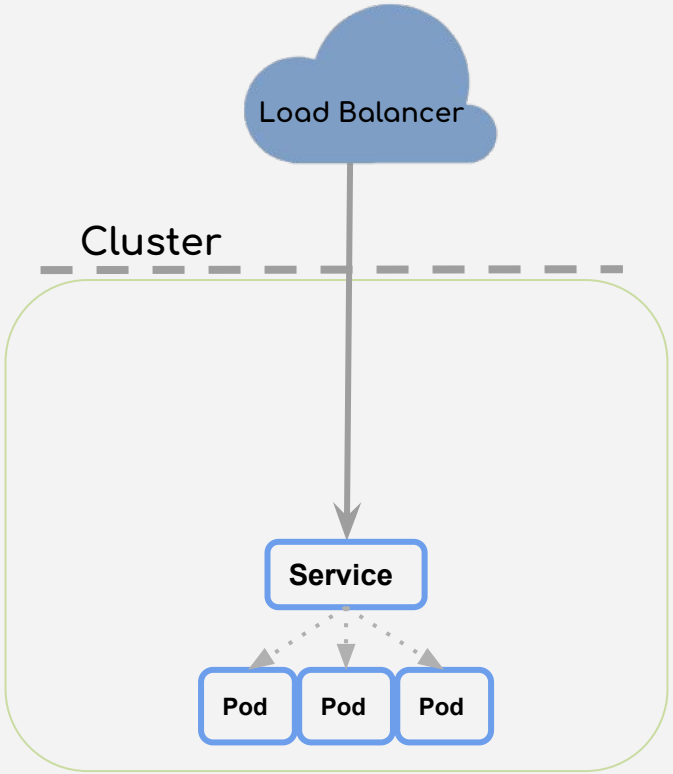
Module 8

Ingress

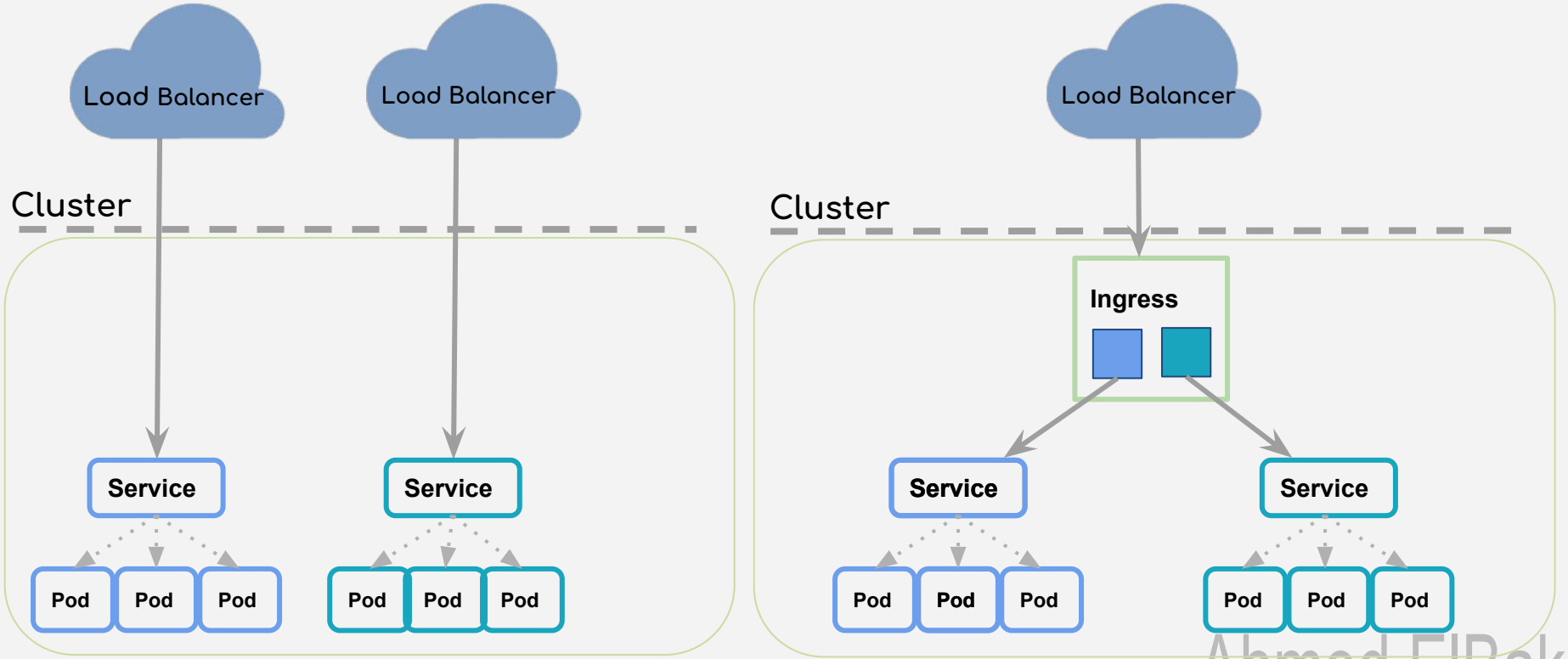
Service Types



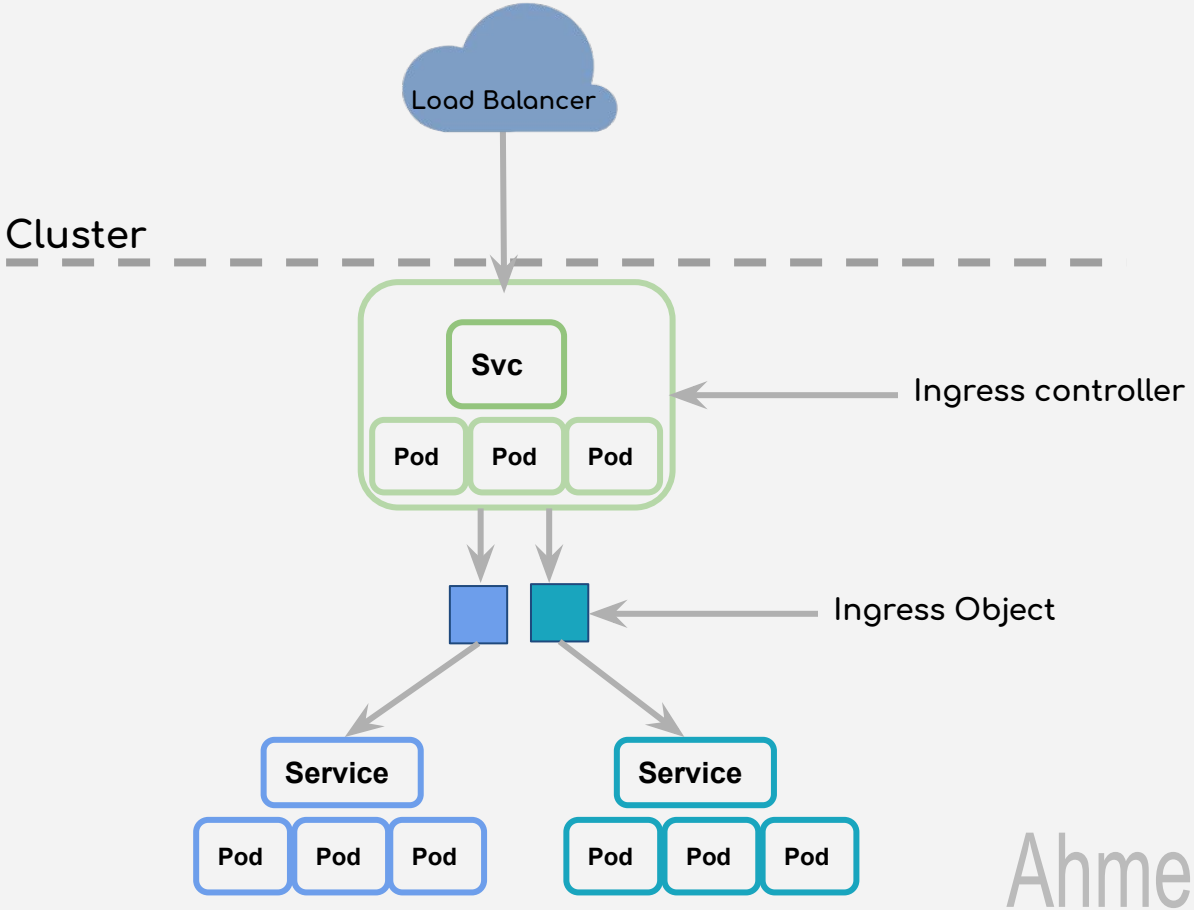
Ingress



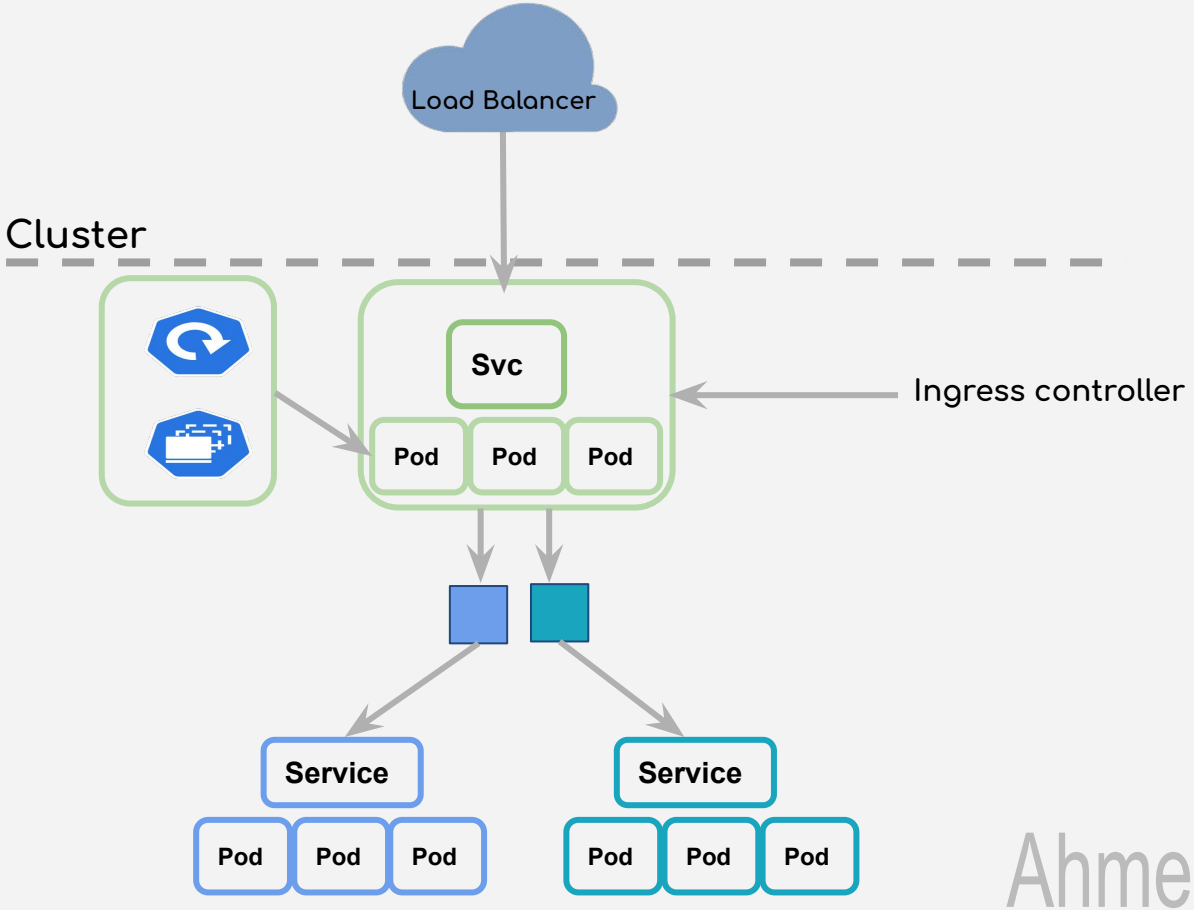
Ingress



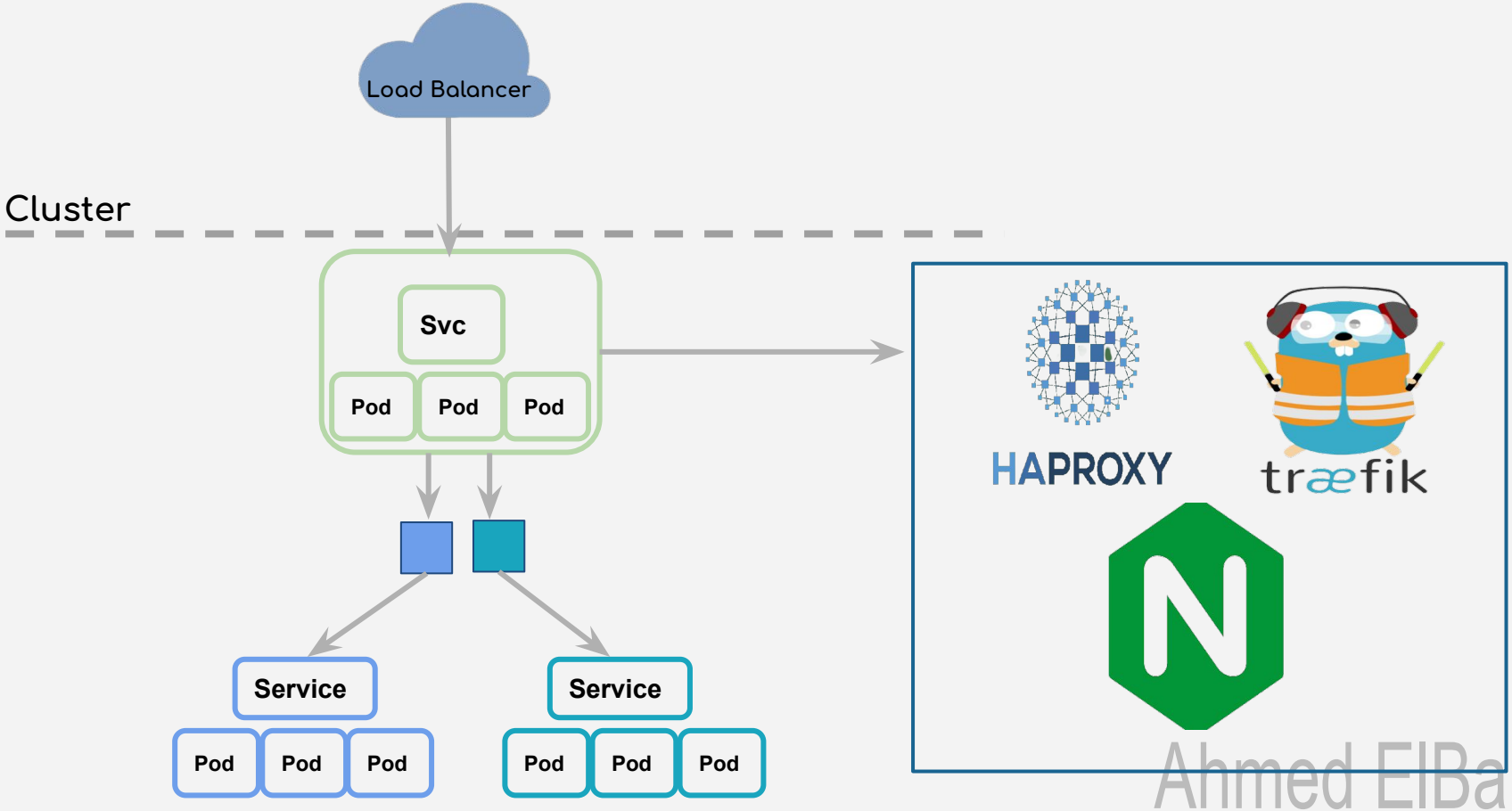
Ingress



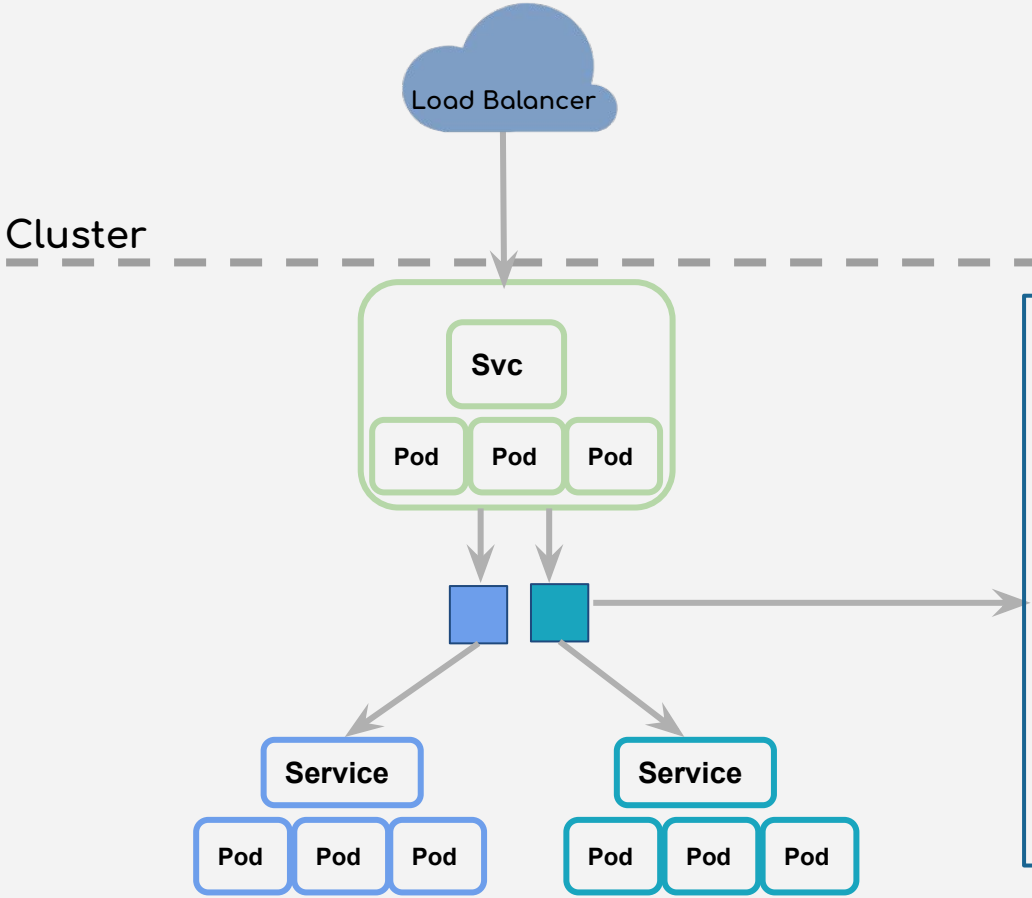
Ingress



Ingress

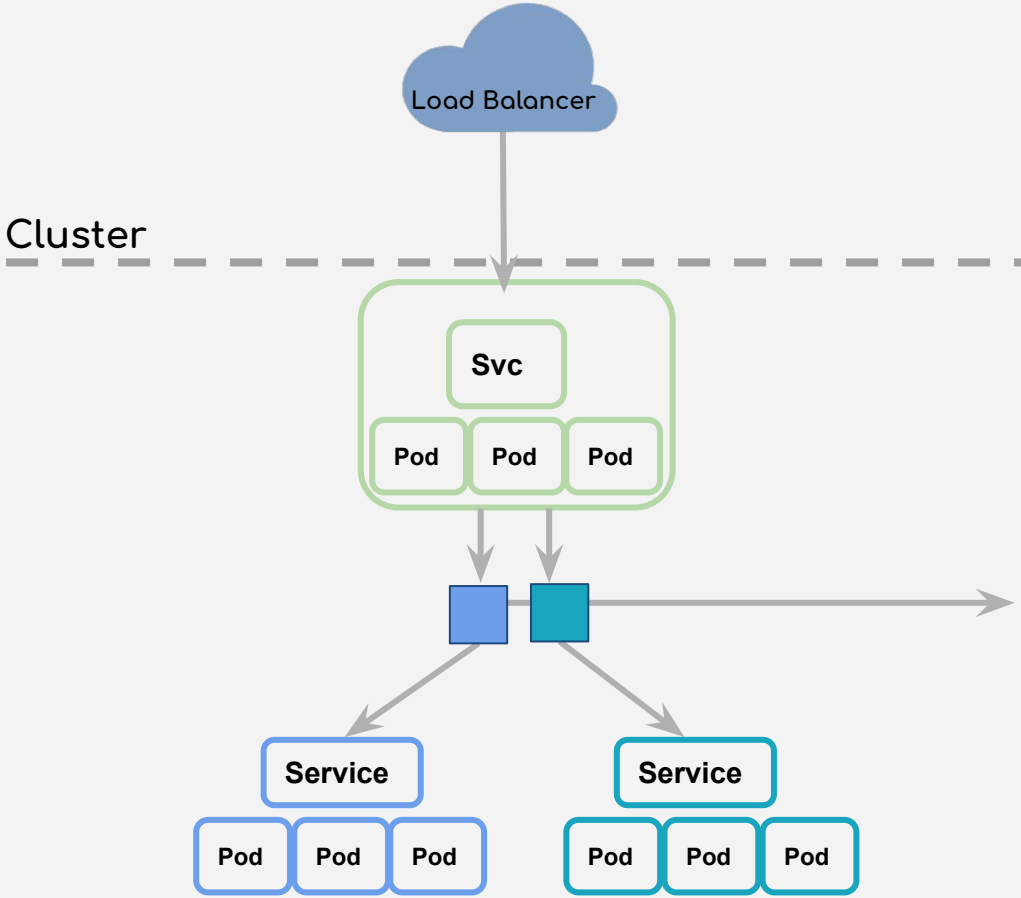


Ingress



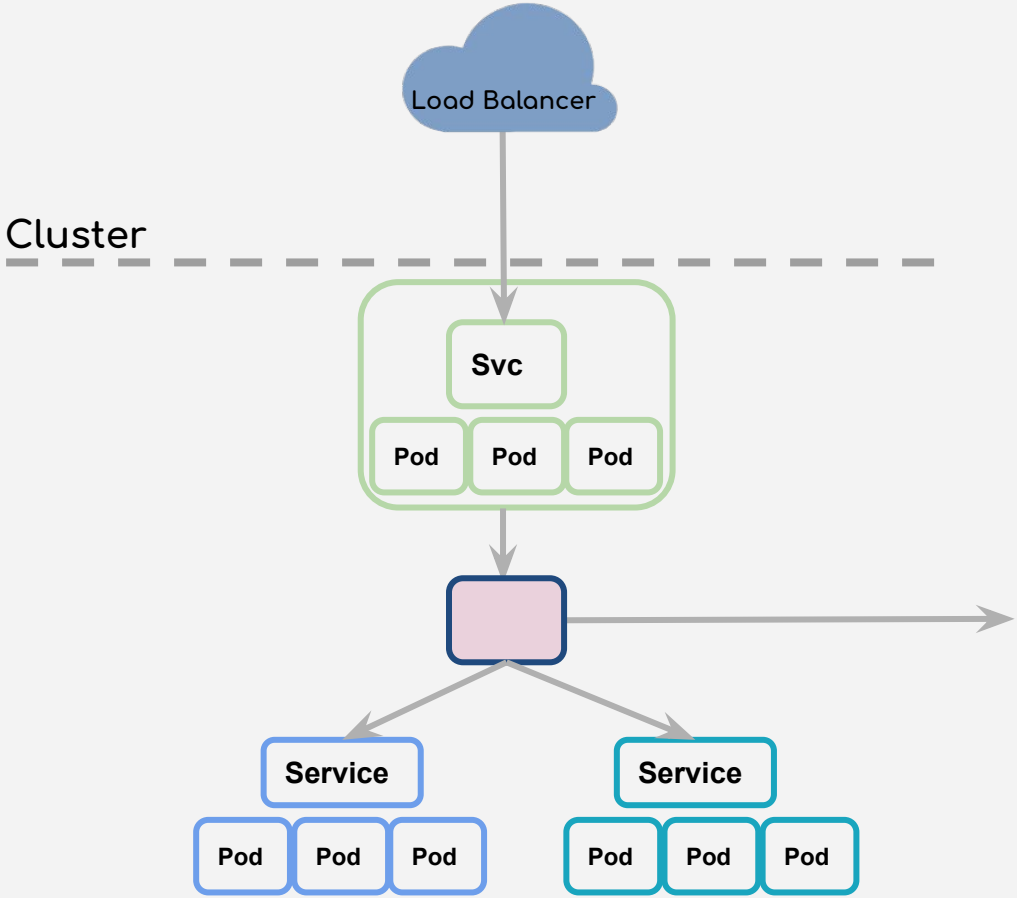

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: catalog
spec:
  ingressClassName: nginx
  rules:
  - http:
      paths:
      - path: /catalog
        pathType: Prefix
        backend:
          service:
            name: catalog
            port:
              number: 80
```

Ingress



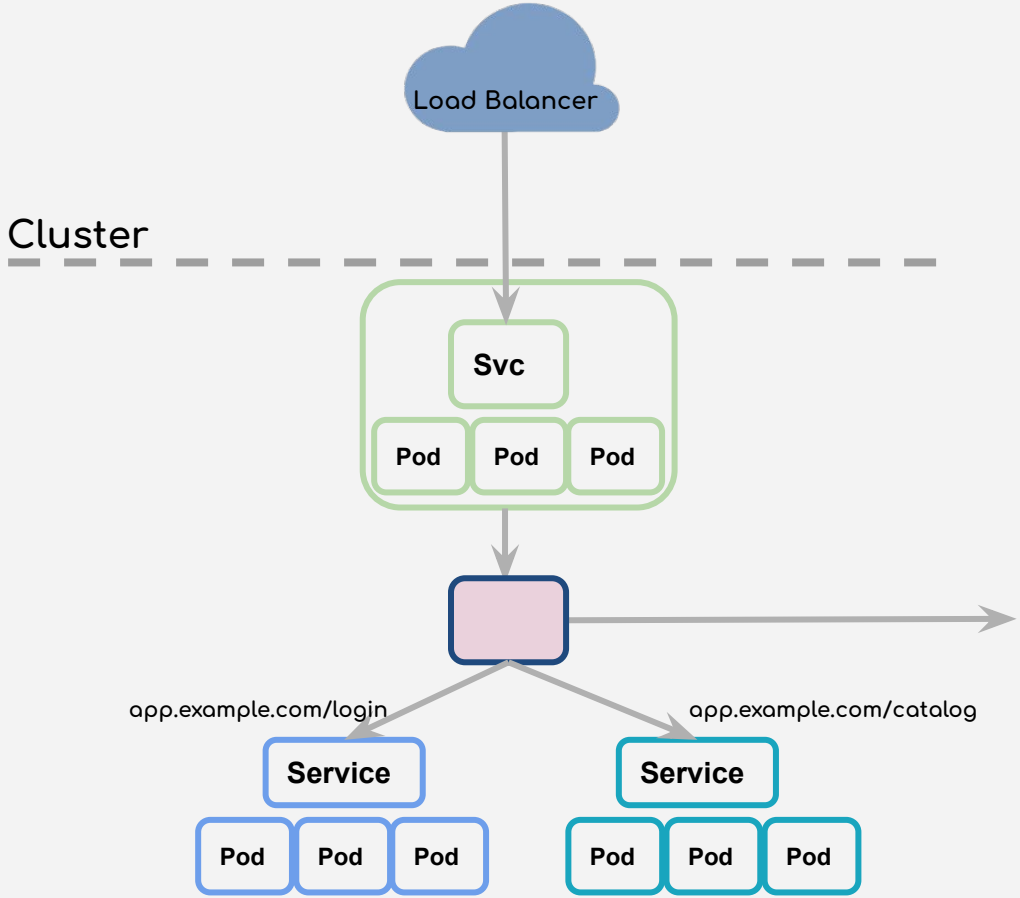

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: login
spec:
  ingressClassName: nginx
  rules:
  - http:
      paths:
      - path: /login
        pathType: Prefix
        backend:
          service:
            name: login
            port:
              number: 80
```

Ingress



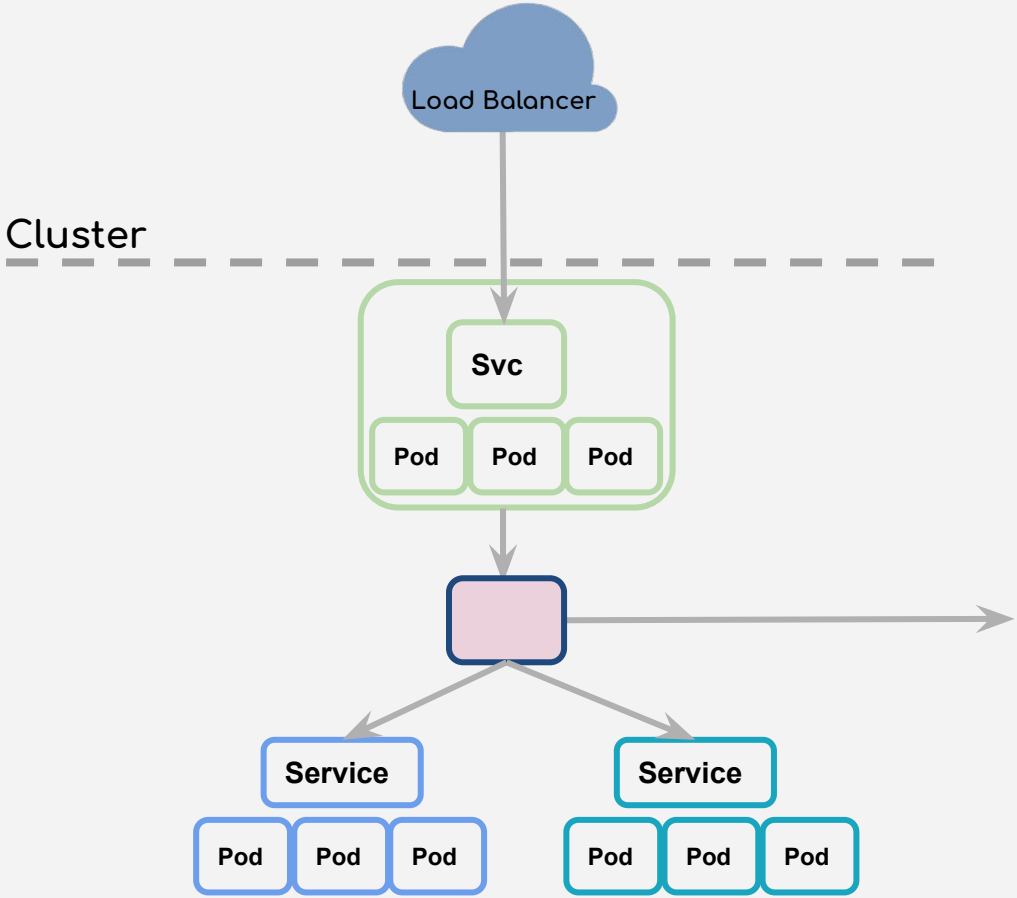

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: example
spec:
  rules:
  - host: app.example.com
    http:
      paths:
      - path: /login
        pathType: Prefix
        backend:
          service:
            name: login
            port:
              number: 4200
    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

Ingress



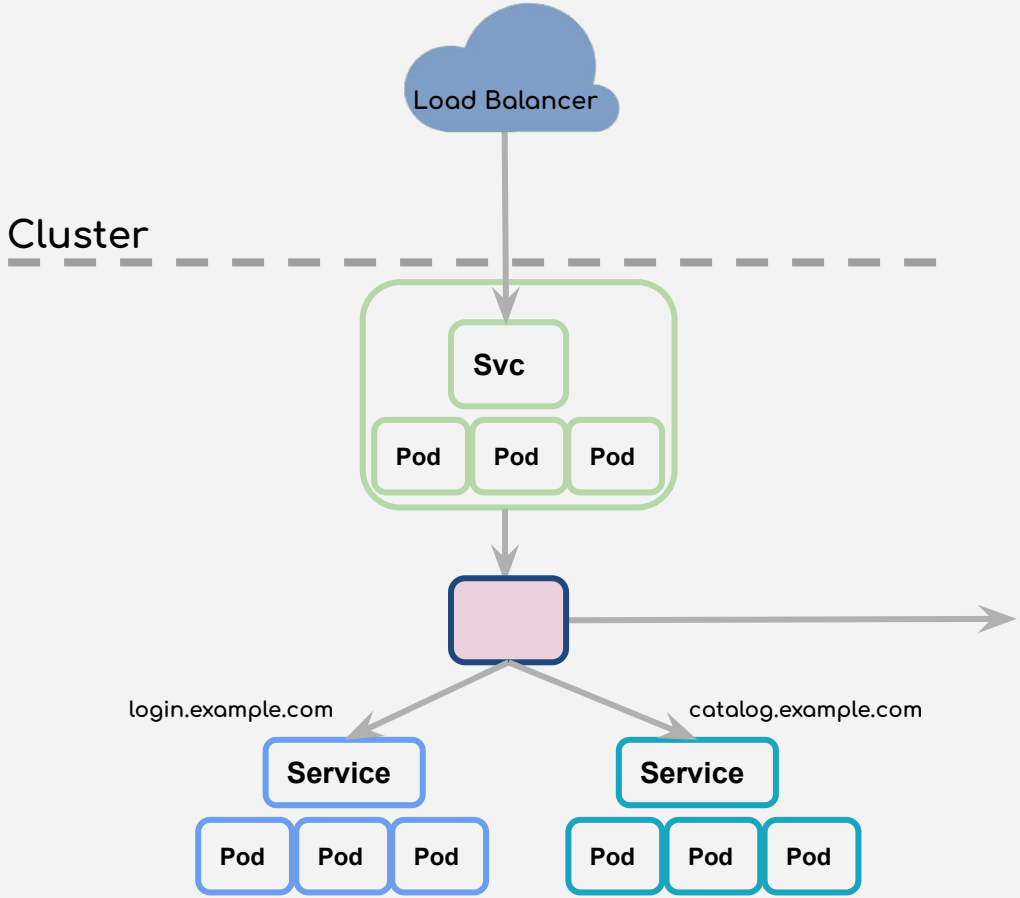

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: example
spec:
  rules:
  - host: app.example.com
    http:
      paths:
      - path: /login
        pathType: Prefix
        backend:
          service:
            name: login
            port:
              number: 4200
    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

Ingress



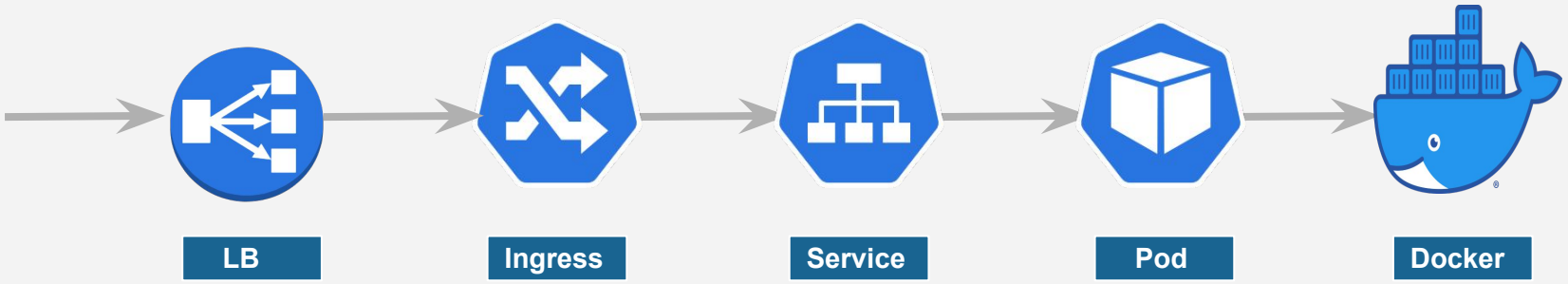

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: example
spec:
  rules:
  - host: login.example.com
    http:
      paths:
      - pathType: Prefix
        path: "/"
        backend:
          service:
            name: login
            port:
              number: 80
  - host: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

Ingress




```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




DEMO
Ahmed ElBakry