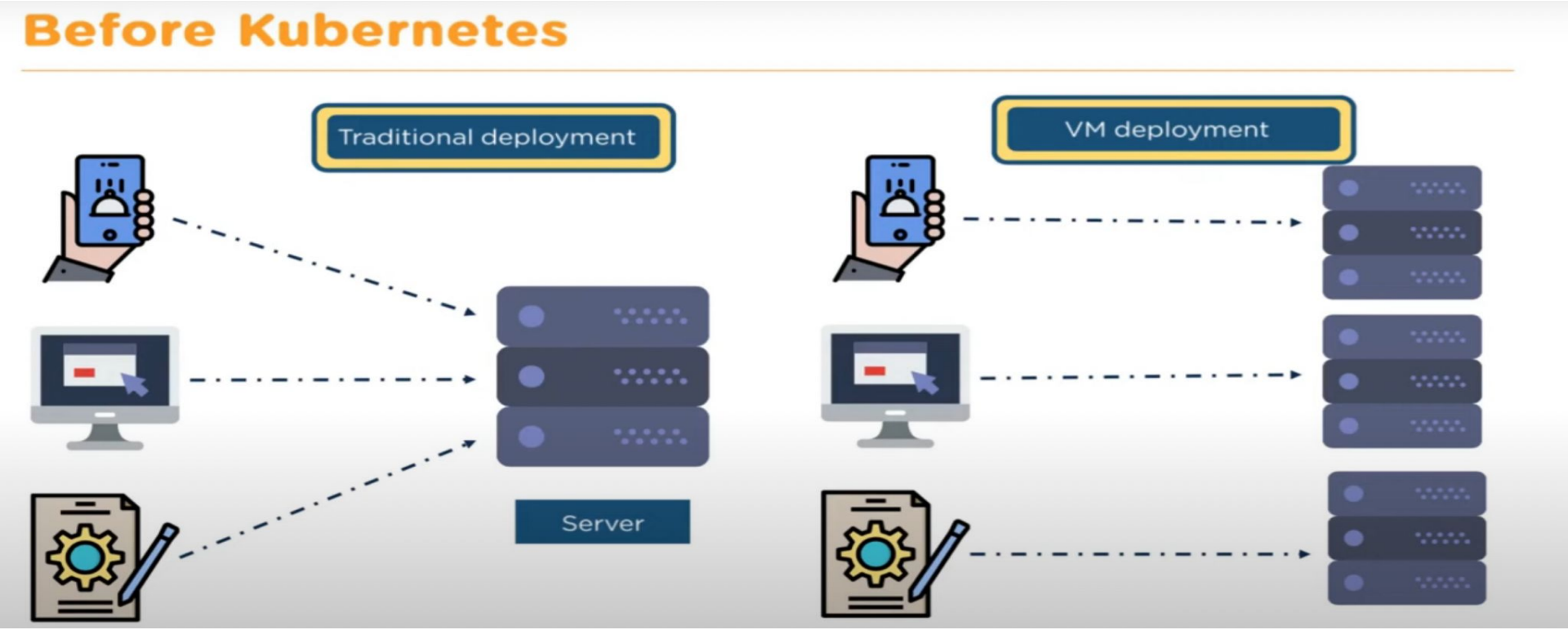
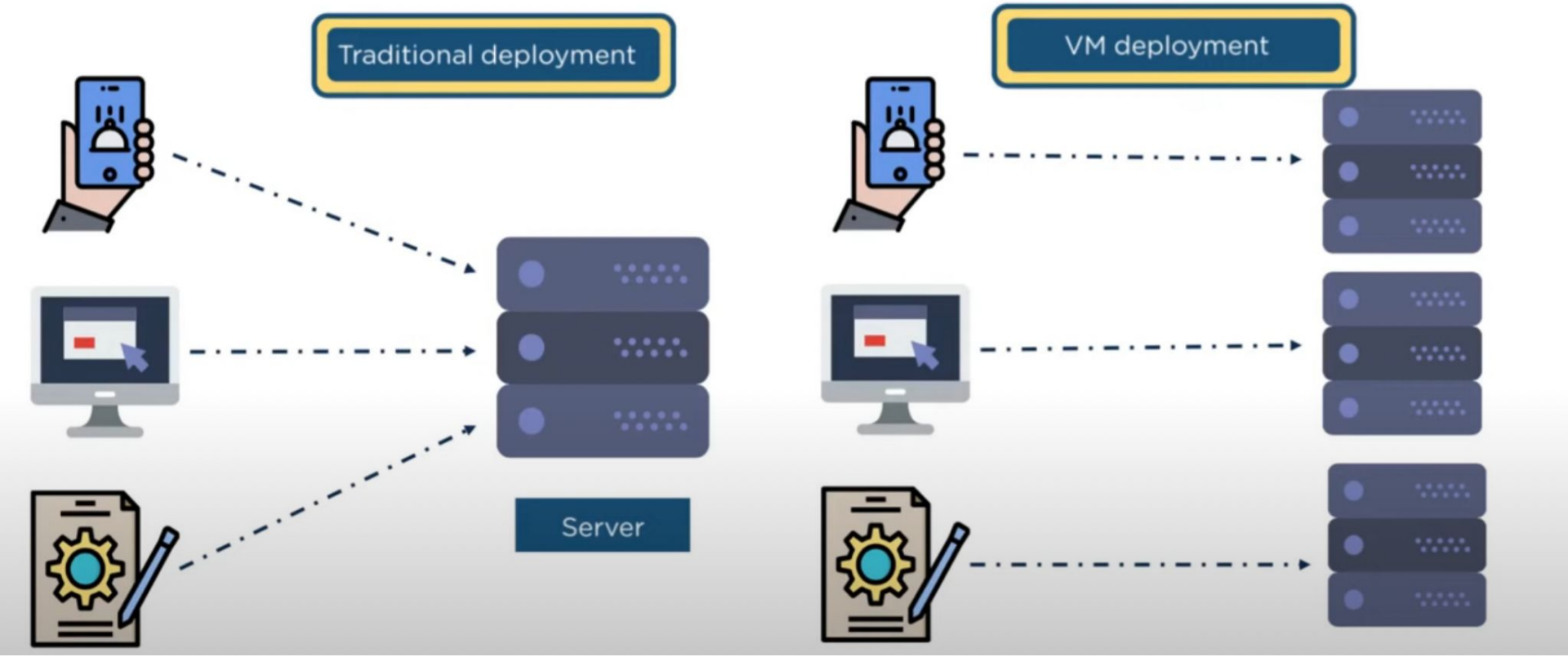
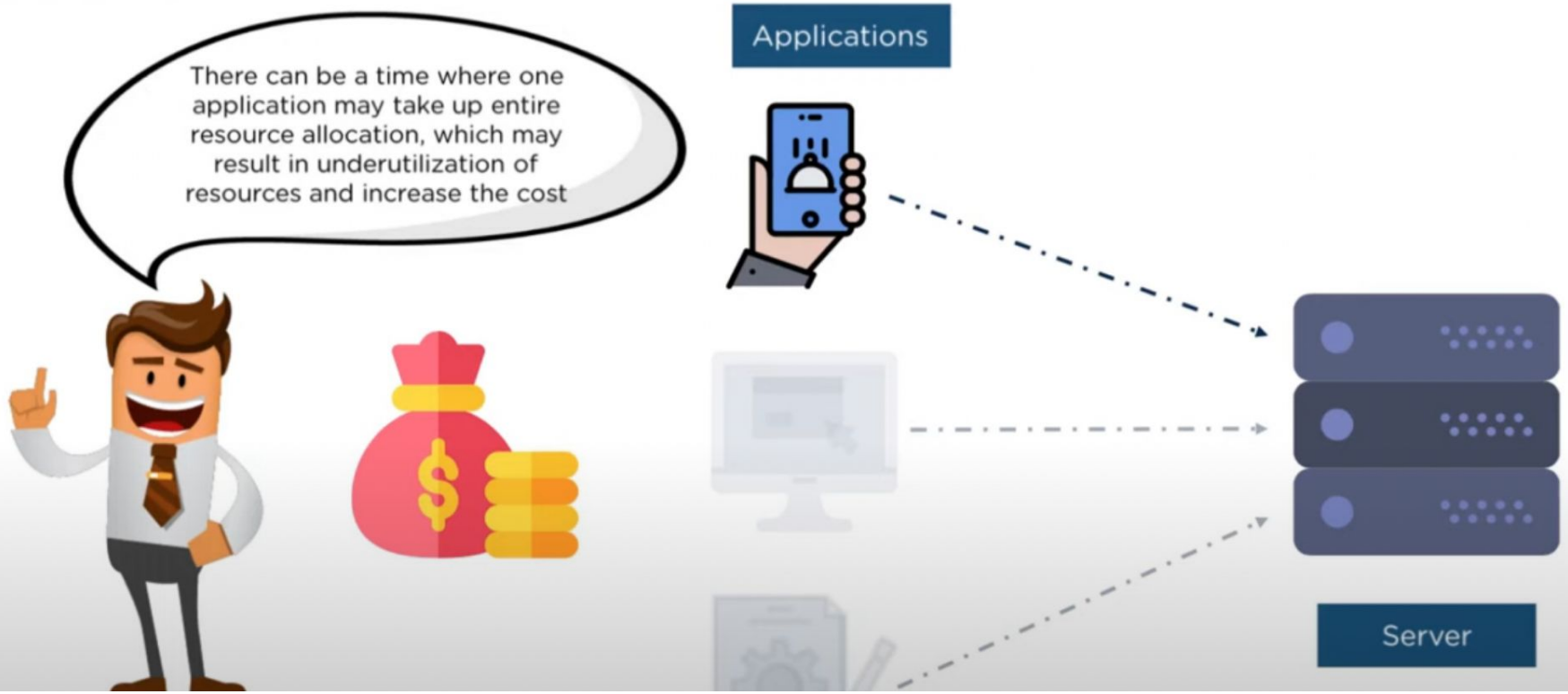
Kubernetes Introduction



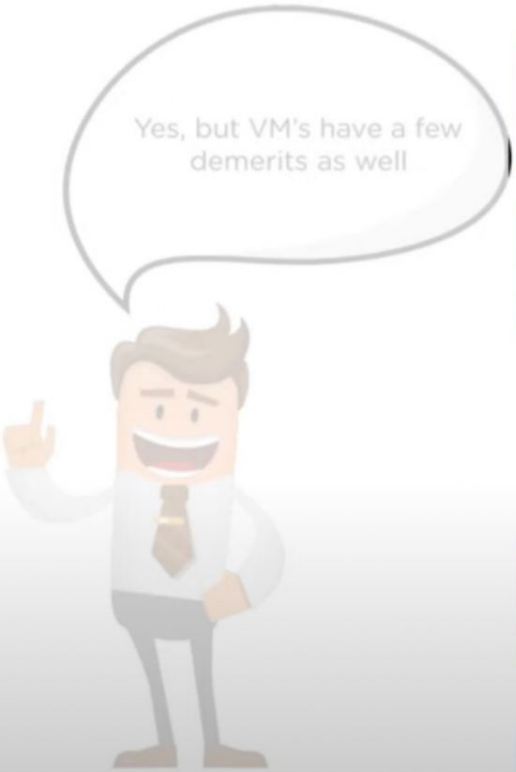
Before Kubernetes



Before Kubernetes – Traditional Deployment

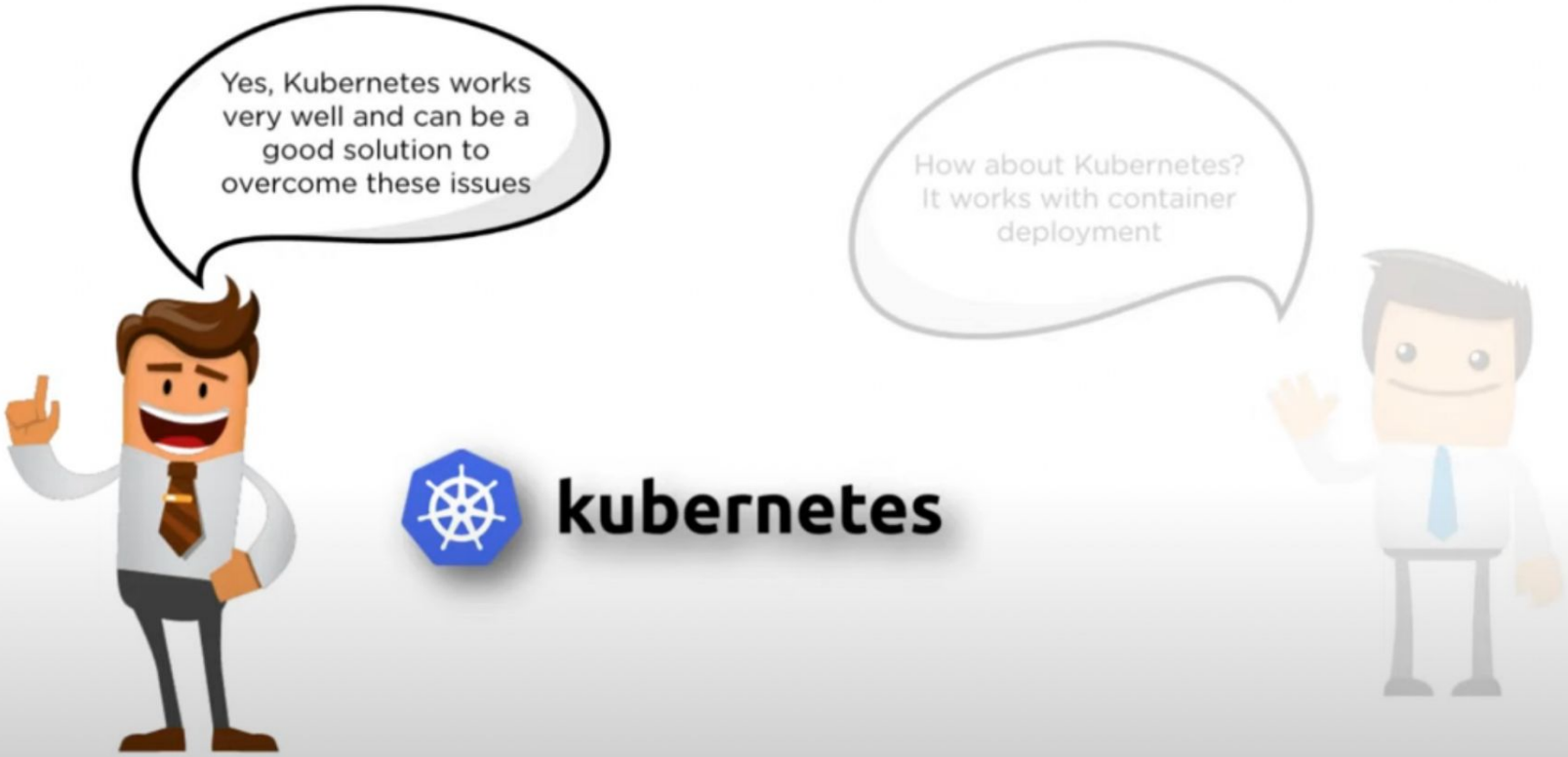


Before Kubernetes – Virtualization Deployment



- **High cost of implementation** — Hardware and software components are required by the VM's that increases the cost of implementation
- **Security risk** — The chances of data breach while using virtualization is common
- **Availability issue** — Assets are unavailable most of the time because the resources are being used by other applications
- **Limitations and restrictions** — Not every application or server can work in a virtualization environment. Some may require hybrid machines to work with
- **Time-consuming** — It is a very time-consuming process. This is because extra steps are followed to generate the desired results

After Kubernetes

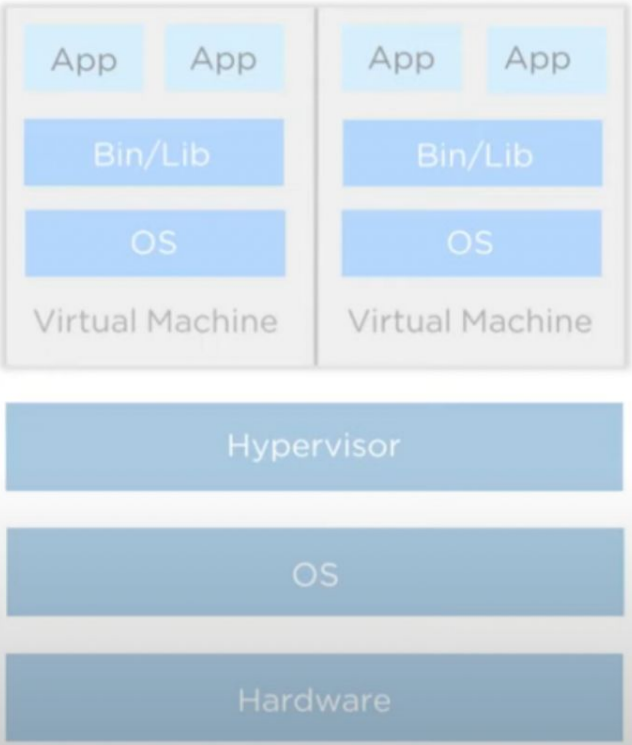


Virtual Machine vs Kubernetes

Major differences are:

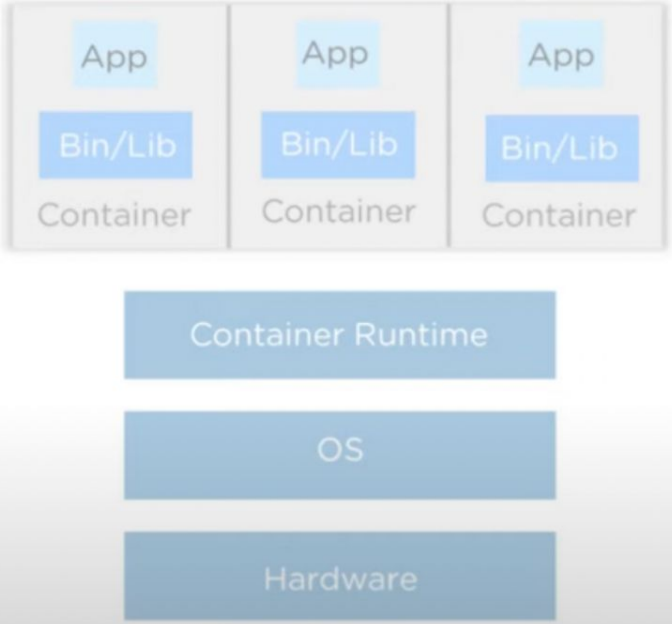
Not secured	Security Risk	Secured
Not easily portable	Portability	Portable
Time is more	Time Consuming	Time is less

Virtual Machine vs Kubernetes

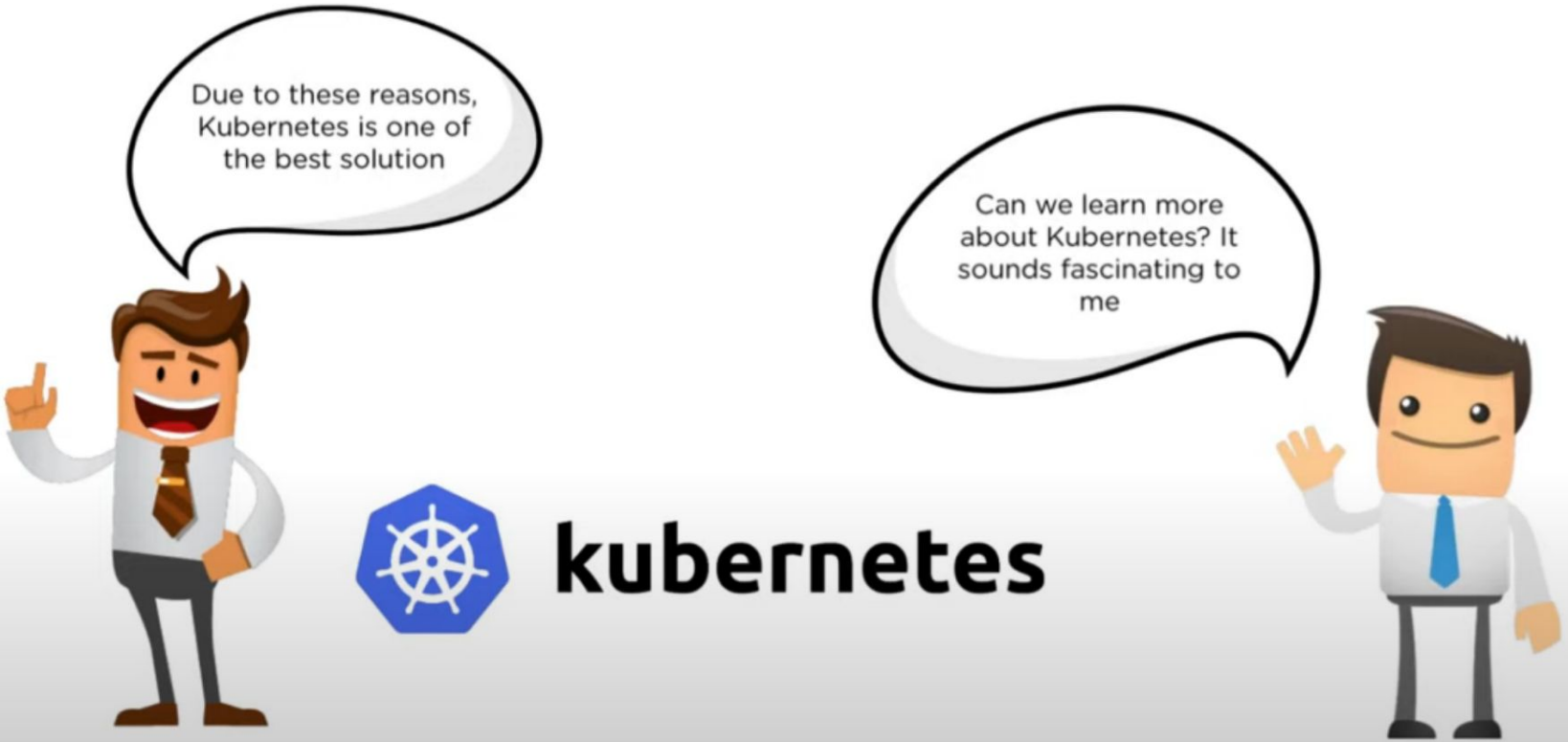


VM – They have less isolation when compared to Kubernetes, and hence, security risks are higher

Kubernetes - They have better isolation properties to share the OS among various applications

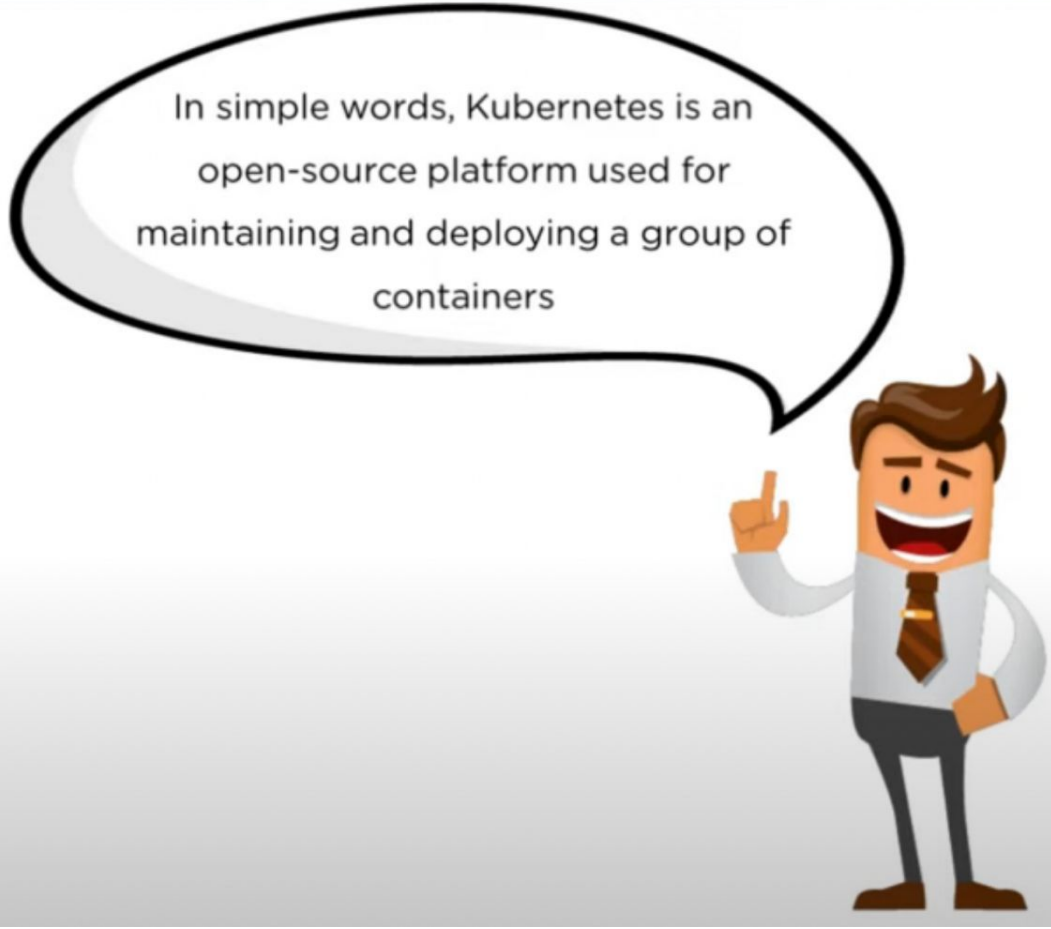


Kubernetes Era



What is Kubernetes?

kubernetes



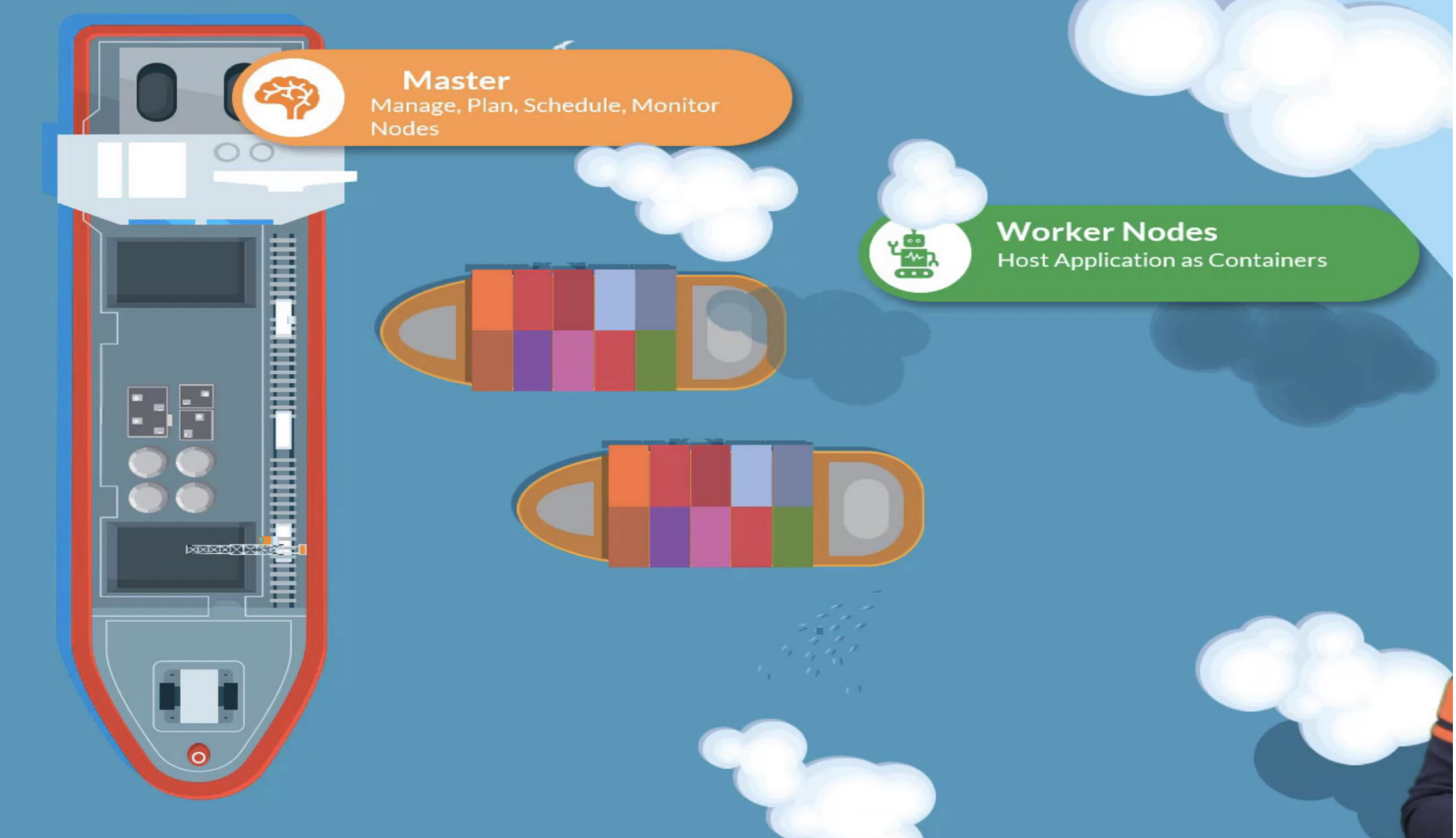
Benefits of Kubernetes

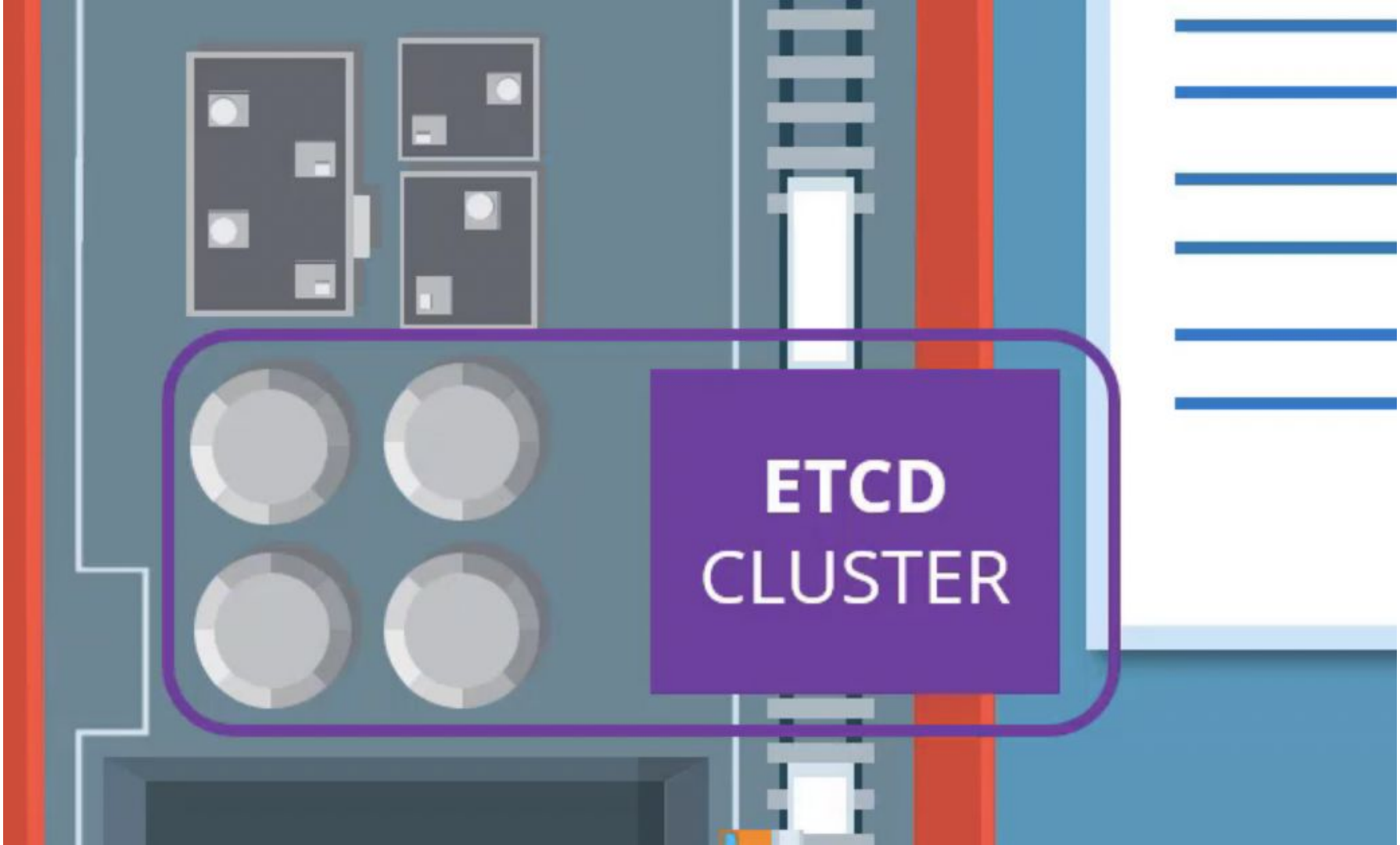
- Portable and 100% open-source
- Workload scalability
- High availability
- Designed for deployment
- Service discovery and load balancing

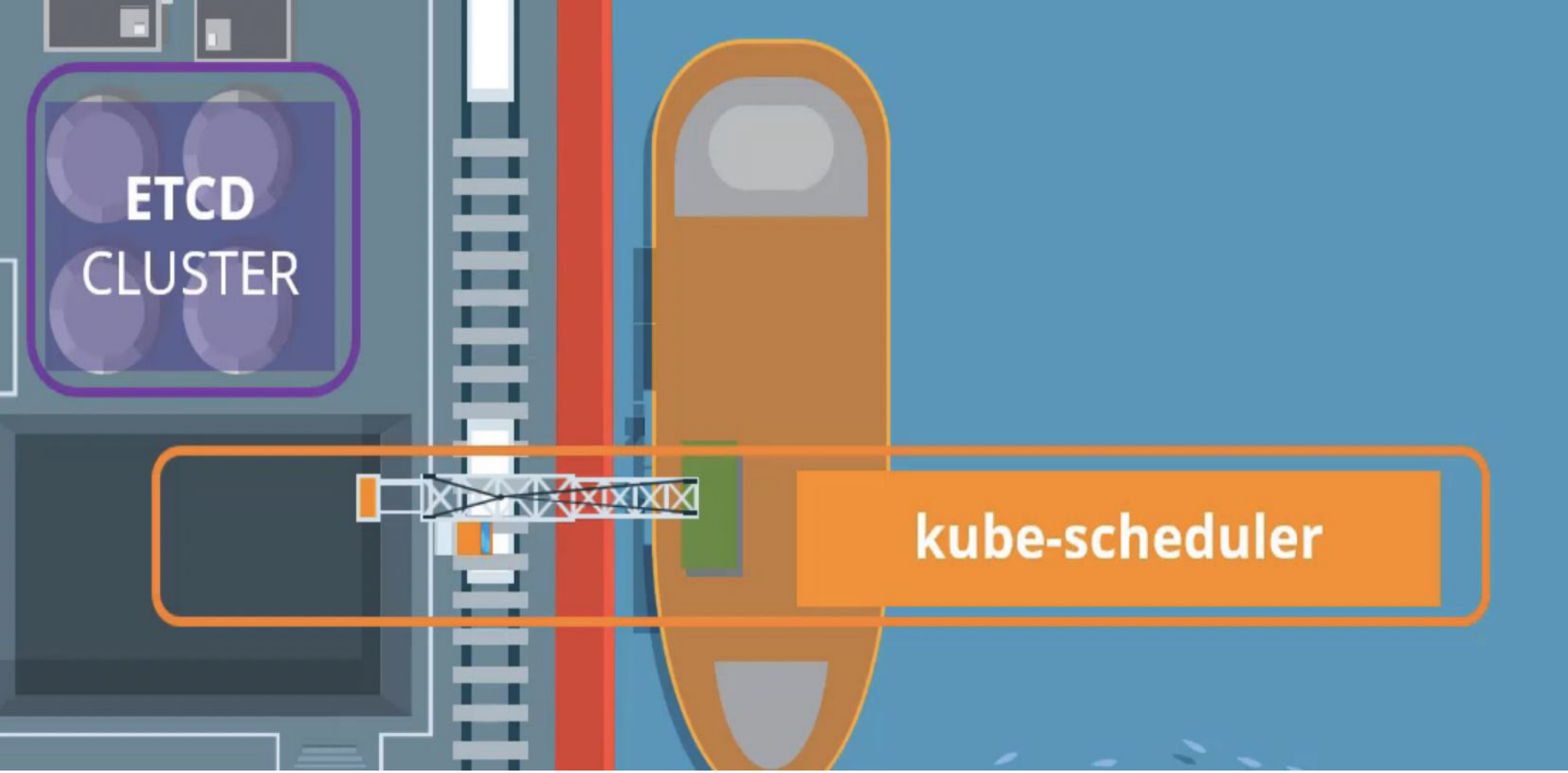
Kubernetes specifies how much CPU and RAM each container needs. Once resources are defined, managing the resources and making decisions for them becomes easy

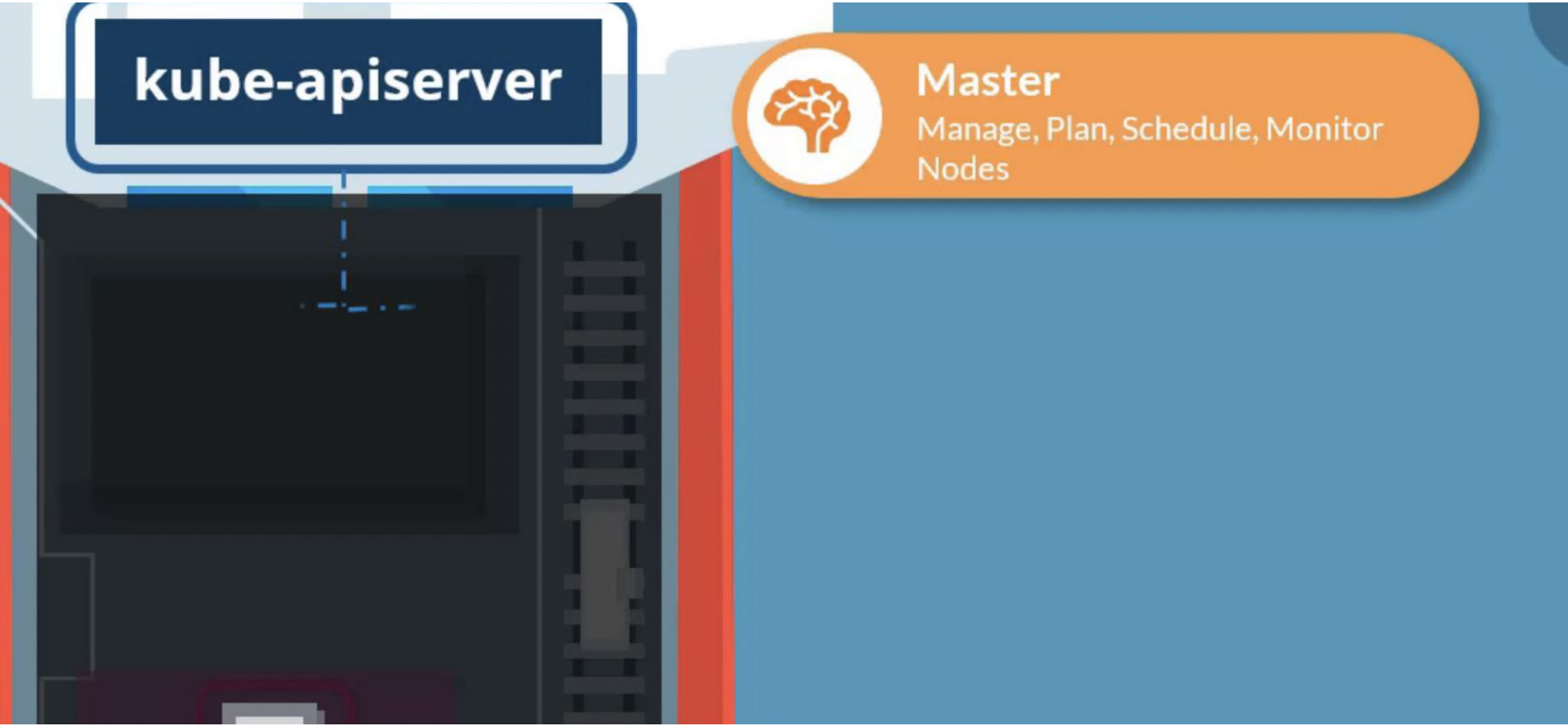
- Storage orchestration
- Self healing
- Automated rollouts and rollbacks
- Automatic bin packing

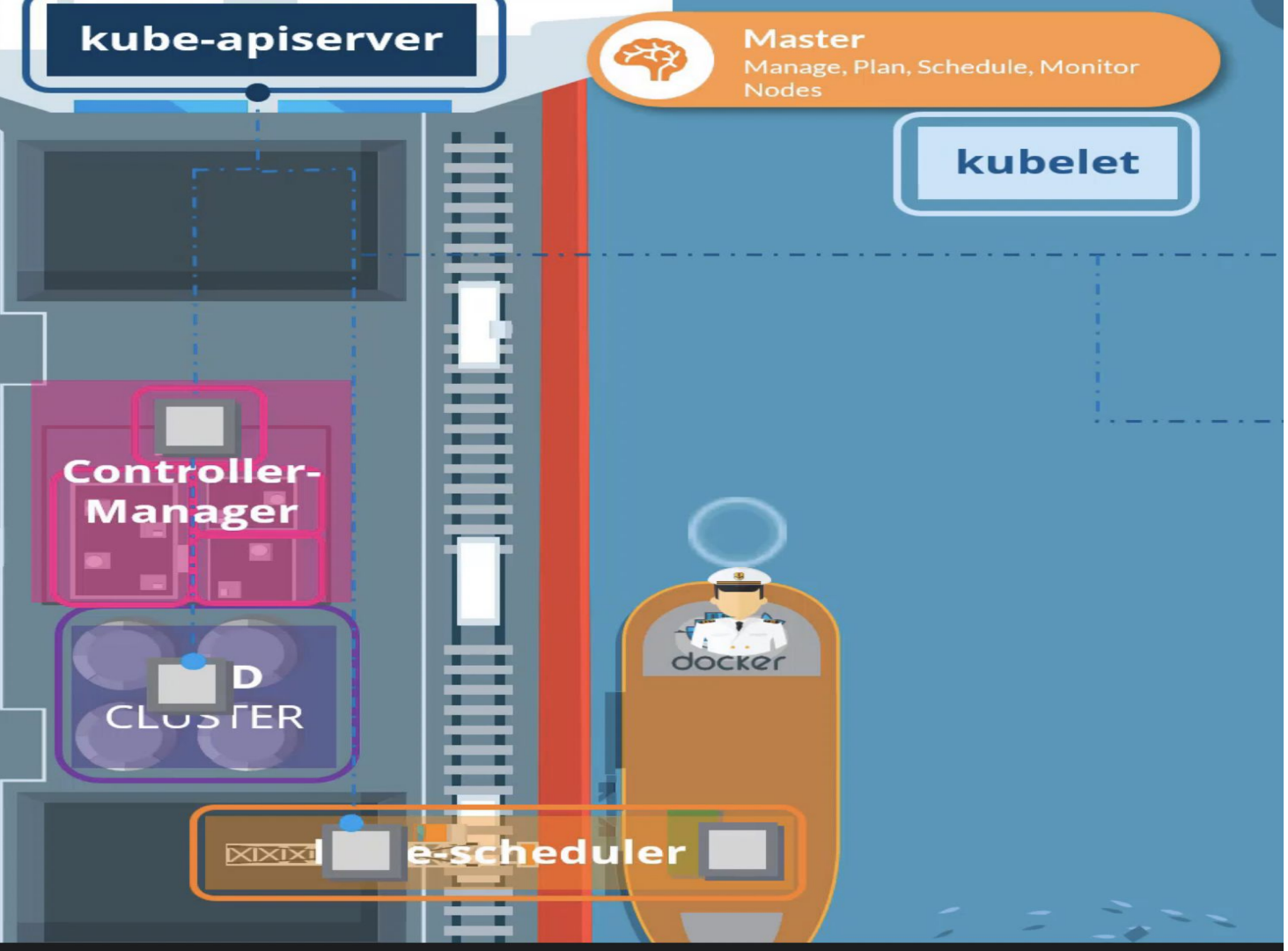

Worker Nodes
Host Application as Containers


Master
Manage, Plan, Schedule, Monitor Nodes
Worker Nodes
Host Application as Containers


ETCD
CLUSTER


ETCD
CLUSTER
kube-scheduler


kube-apiserver
Master
Manage, Plan, Schedule, Monitor
Nodes


kube-apiserver
Master
Manage, Plan, Schedule, Monitor Nodes
kubelet
Controller-Manager
docker

Kubernetes Architecture

Master
Manage, Plan, Schedule, Monitor Nodes

Worker Nodes
Host Application as Containers

ETCD CLUSTER

kube-apiserver

Kube Controller Manager

kube-scheduler

kubelet

Kube-proxy

Container Runtime Engine
Run containers

docker rkt

kubelet

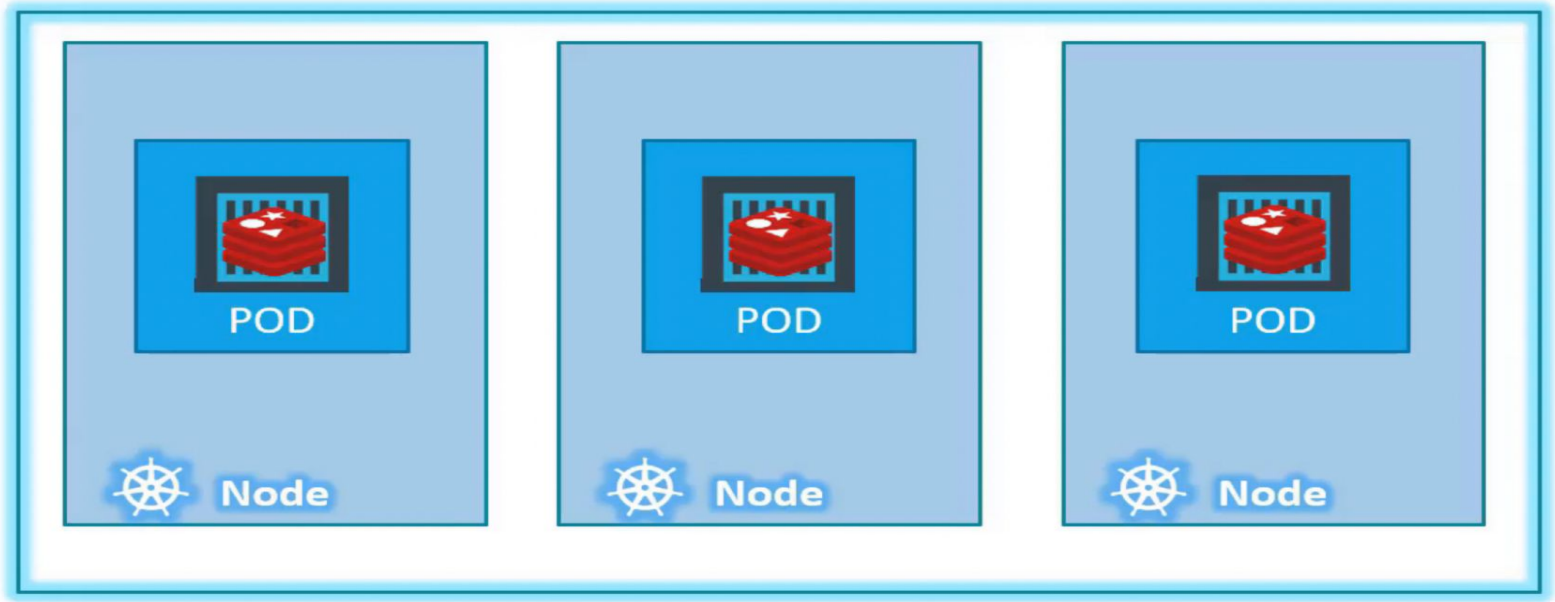
Kube-proxy

Container Runtime Engine
Run containers

docker rkt

Pods

POD



replicaset-definition.yml

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: myapp-replicaset
  labels:
      app: myapp
      type: front-end
spec:
  template:
    metadata:
      name: myapp-pod
      labels:
        app: myapp
        type: front-end
    spec:
      containers:
      - name: nginx-container
        image: nginx

  replicas: 3
  selector:
    matchLabels:
      type: front-end
```

pod-definition.yml

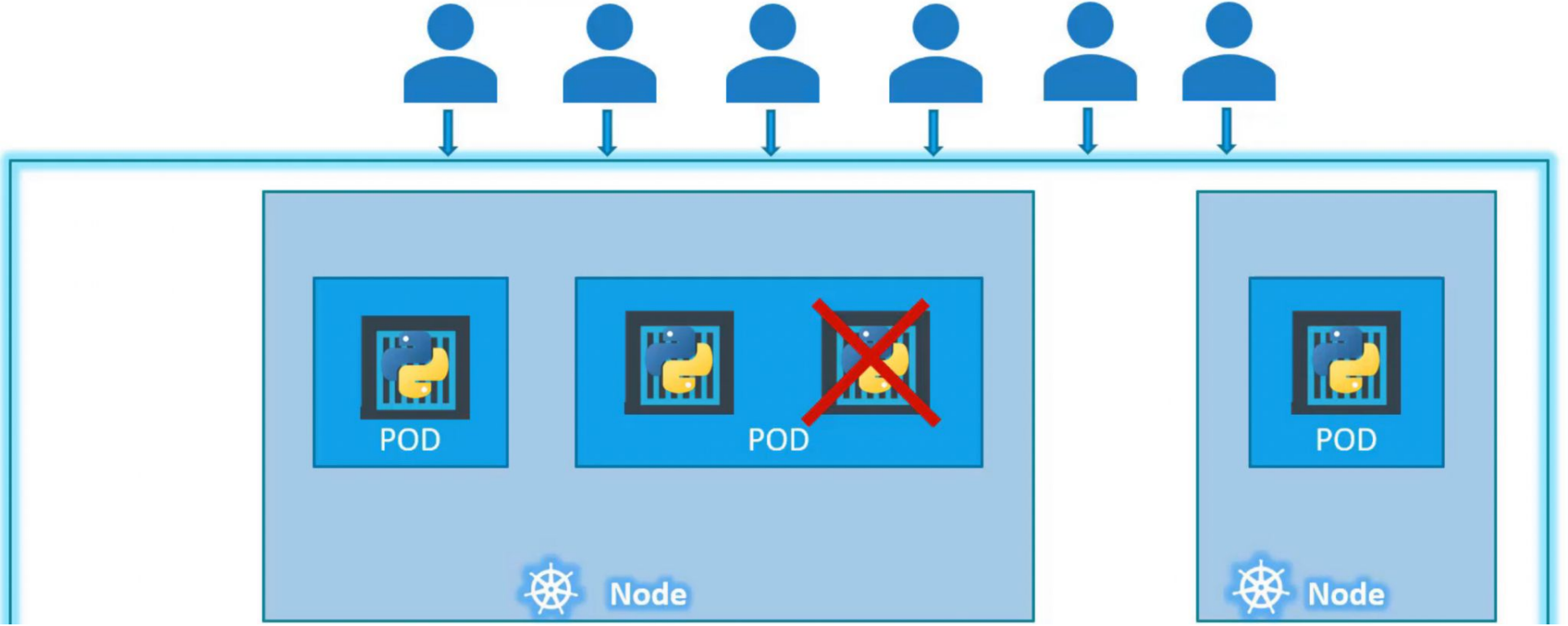
```yaml
apiVersion: v1
kind: Pod
```

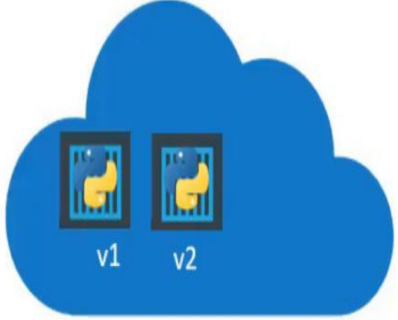
```
> kubectl create -f replicaset-definition.yml
replicaset "myapp-replicaset" created
```

```
> kubectl get replicaset
NAME              DESIRED   CURRENT   READY     AGE
myapp-replicaset  3         3         3         19s
```

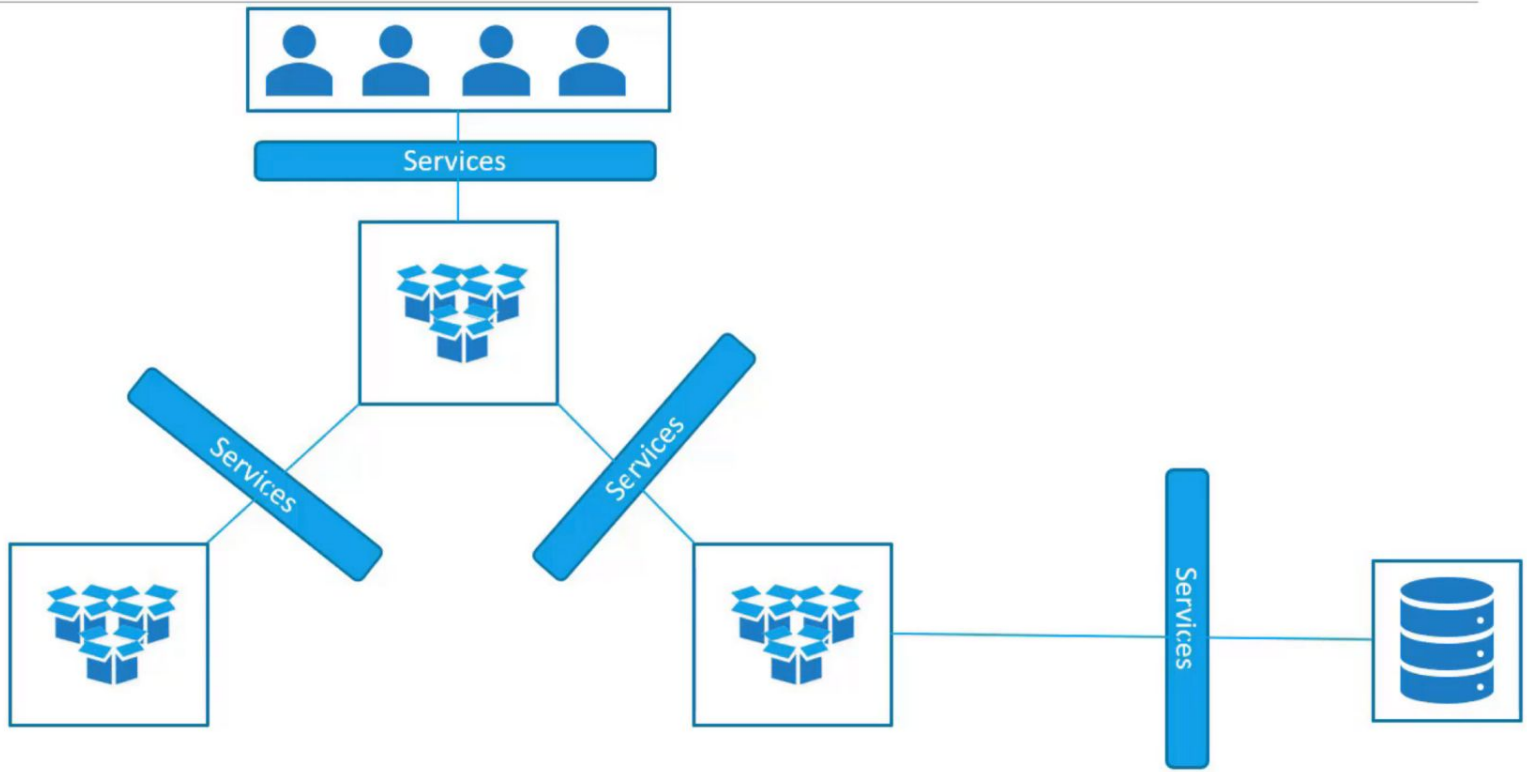
Labels and Selectors

POD

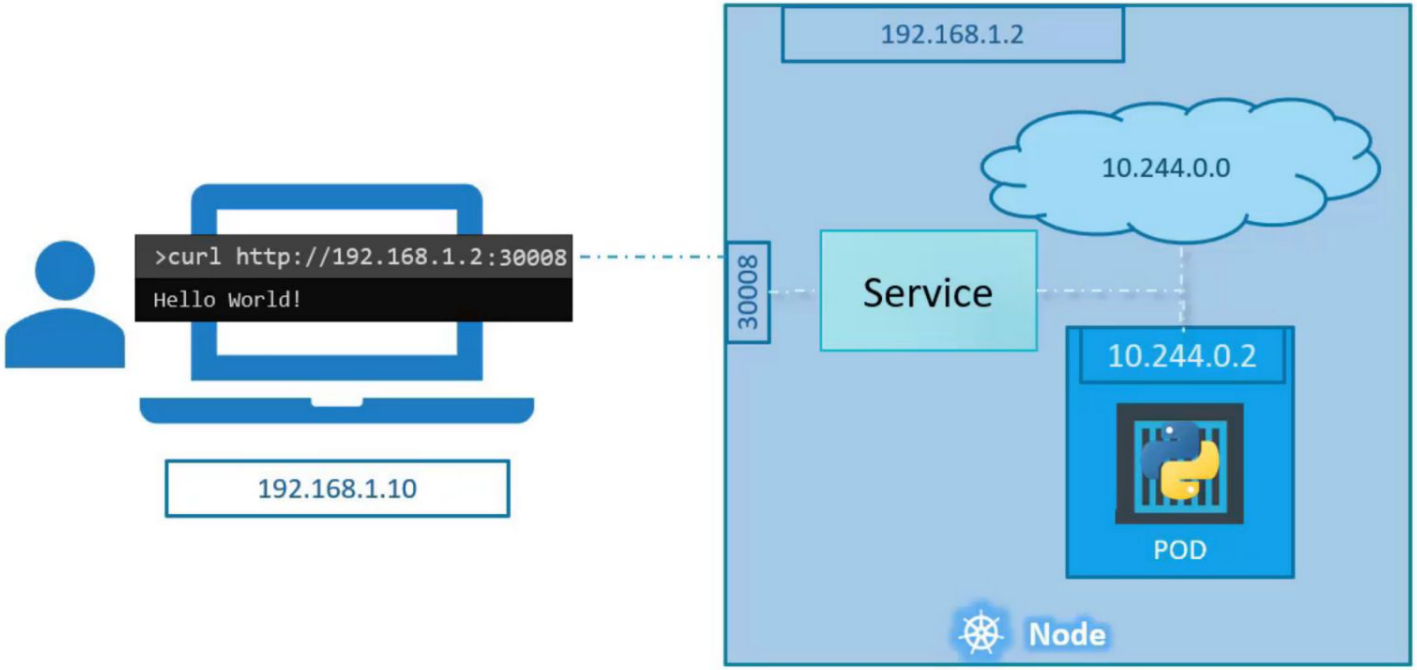



v1
v2

Services

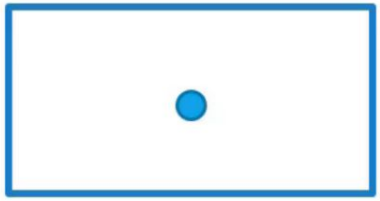


Service

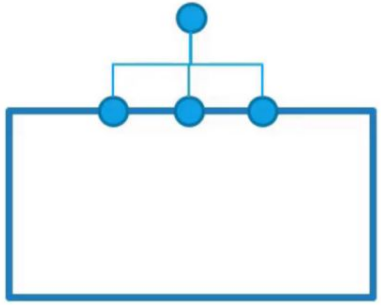

>curl http://192.168.1.2:30008
Hello World!
192.168.1.10
192.168.1.2
10.244.0.0
30008
Service
10.244.0.2
POD
Node

Services Types

NodePort

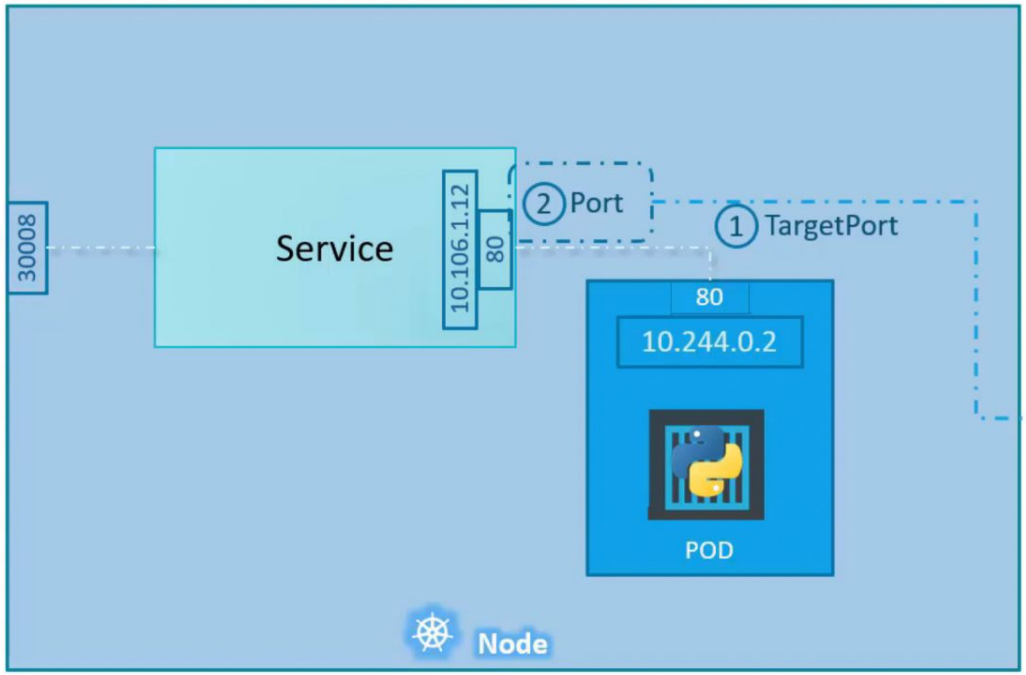


ClusterIP



LoadBalancer

Service - NodePort

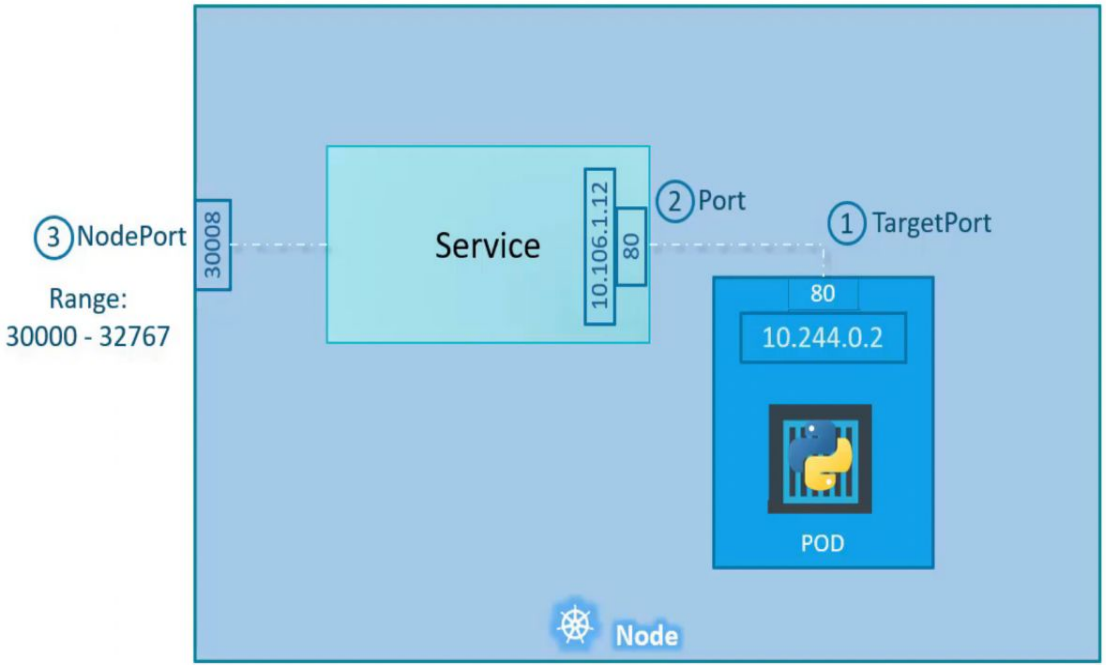


service-definition.yml

```yaml
apiVersion: v1
kind: Service
metadata:
    name: myapp-service

spec:
    type: NodePort
    ports:
     - targetPort: 80
       port: 80
       nodePort: 30008
```

Service - NodePort

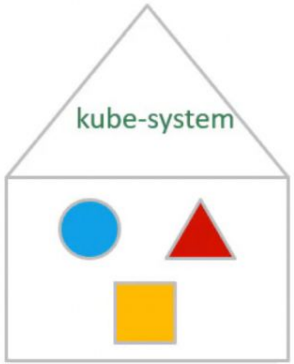


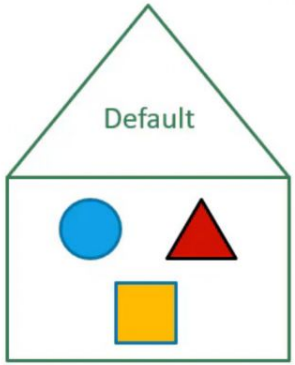
service-definition.yml

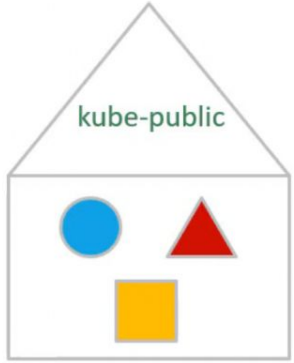
```yaml
apiVersion: v1
kind: Service
metadata:
    name: myapp-service

spec:
    type: NodePort
    ports:
     - targetPort: 80
      *port: 80
       nodePort: 30008
```

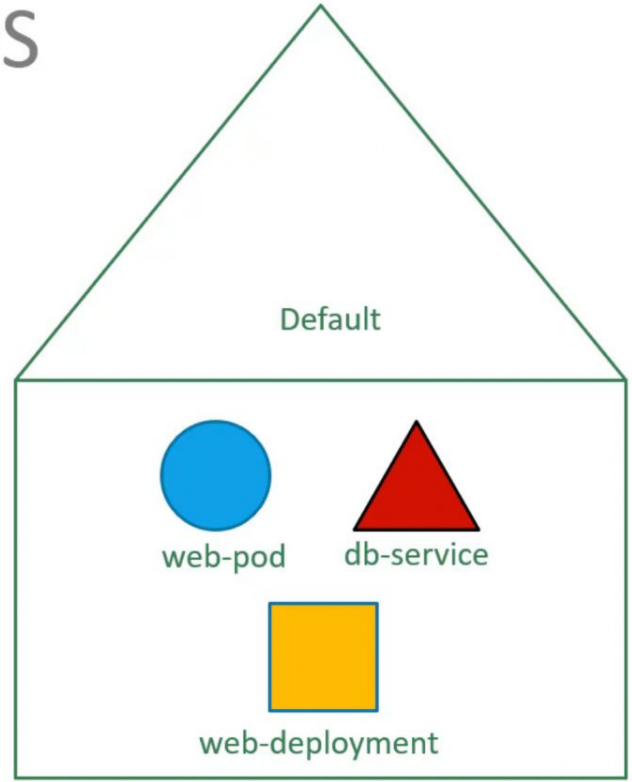
Namespace - Isolation

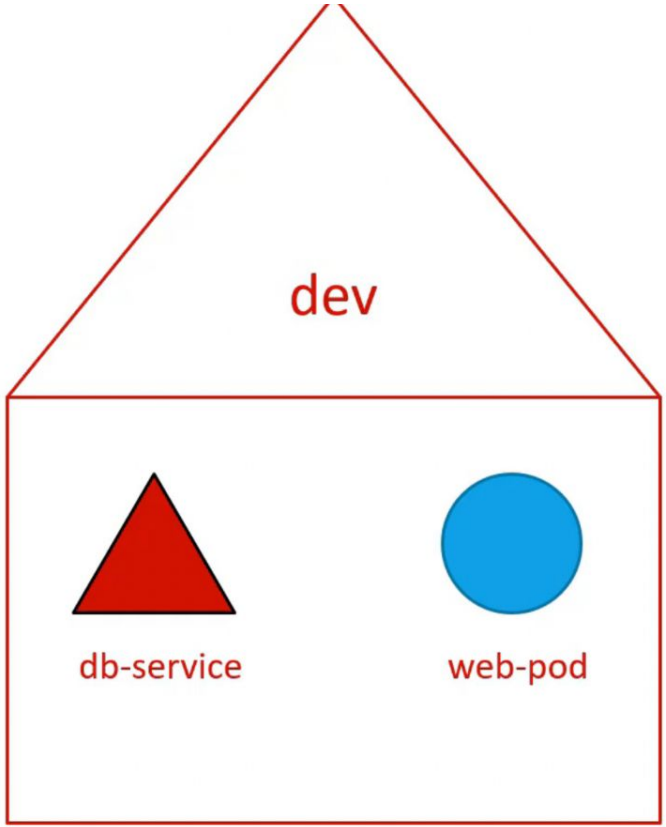






DNS

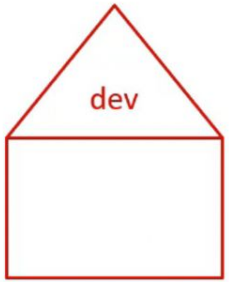


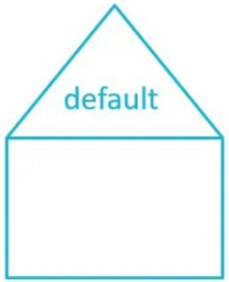


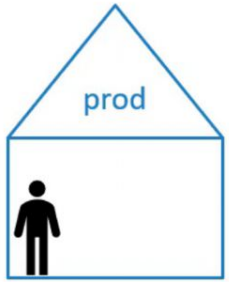

```
mysql.connect("db-service")
```

```
mysql.connect("db-serv
```

Switch








```
> kubectl get pods --namespace=dev
```

```
> kubectl get pods
```

```
> kubectl get pods --namespace=prod
```

```
> kubectl config set-context $(kubectl config current-context) --namespace=dev
```

```
> kubectl get pods
```

```
> kubectl get pods --namespace=default
```

```
> kubectl get pods --namespace=prod
```

```
> kubectl config set-context $(kubectl config current-context) --namespace=prod
```

```
> kubectl get pods --namespace=dev
```

```
> kubectl get pods --namespace=default
```

```
> kubectl get pods
```

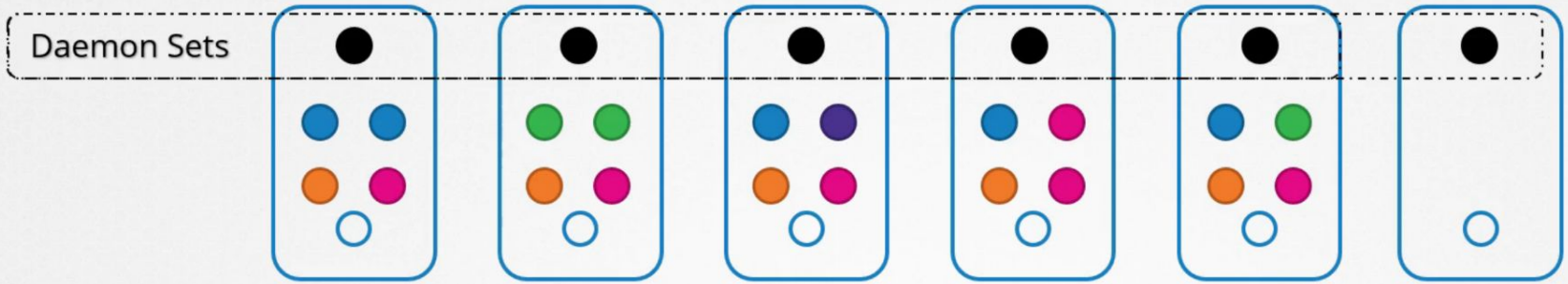
```
> kubectl get pods --all-namespaces
```

https://minikube.sigs.k8s.io/docs/start/

Daemon Sets – UseCase

Monitoring Solution

Logs Viewer



Step 1
Configure cluster

Step 2
Specify networking

Step 3
Configure logging

Step 4
Review and create

Configure cluster

Cluster configuration Info

Name - *Not editable after creation.*
Enter a unique name for this cluster.

javahome-eks

Kubernetes version Info
Select the Kubernetes version for this cluster.

1.20

Cluster Service Role Info - *Not editable after creation.*
Select the IAM Role to allow the Kubernetes control plane to manage AWS resources on your behalf.
To create a new role, go to the IAM console.

Select role

Secrets encryption Info

Once enabled, secrets encryption cannot be modified or removed.

Enable envelope encryption of Kubernetes secrets using KMS
Enable envelope encryption to provide an additional layer of encryption for your Kubernetes secrets.

Tags (0) Info

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Common use cases

- EC2
  Allows EC2 instances to call AWS services on your behalf.
- Lambda
  Allows Lambda functions to call AWS services on your behalf.

Use cases for other AWS services:

EKS

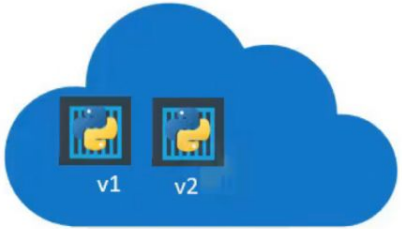
- EKS
  Allows EKS to manage clusters on your behalf.
- **EKS - Cluster**
  Allows access to other AWS service resources that are required to operate clusters managed by EKS.
- EKS - Nodegroup
  Allow EKS to manage nodegroups on your behalf.
- EKS - Fargate pod
  Allows access to other AWS service resources that are required to run Amazon EKS pods on AWS Fargate.
- EKS - Fargate profile
  Allows EKS to run Fargate tasks.
- EKS - Connector
  Allows access to other AWS service resources that are required to connect to external clusters
- EKS Local - Outpost
  Allows Amazon EKS Local to call AWS services on your behalf.

Deployment

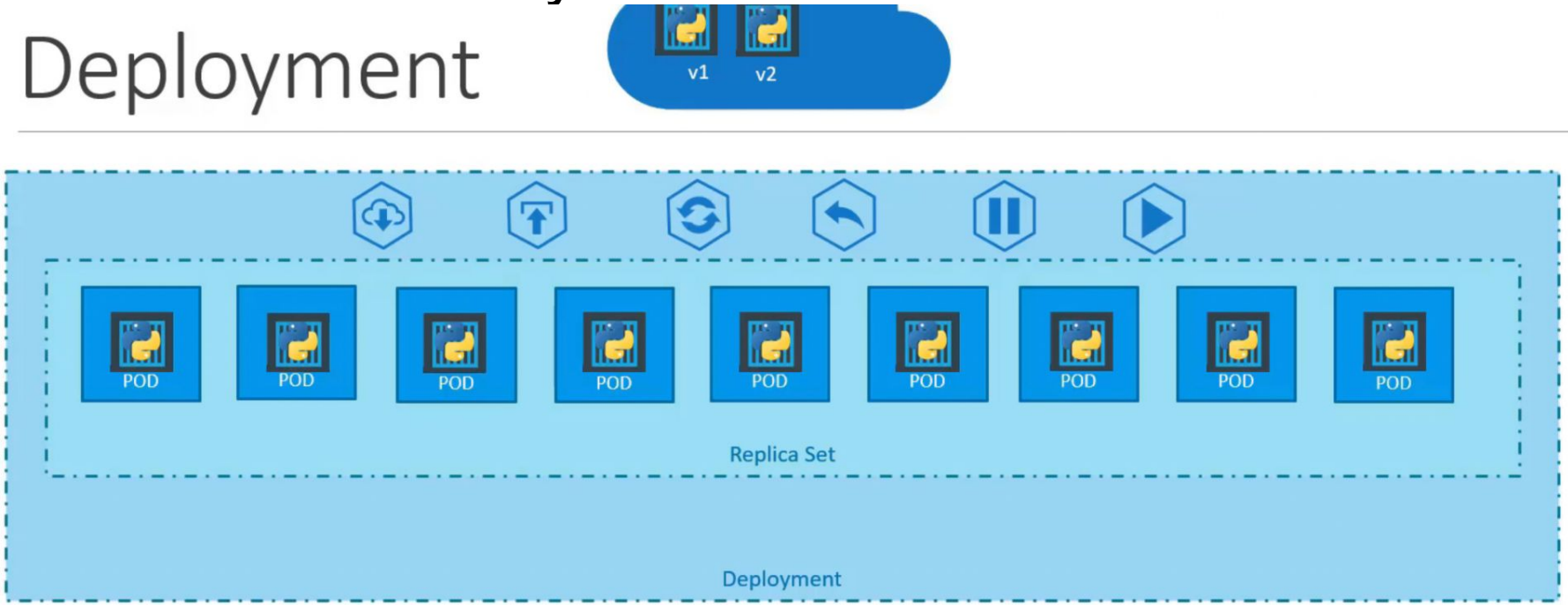
When we need to update instance we need update docker instance seemlesly

Rolling update

We need to update the cluster with rolling update one after another we should be



Rolling update we should be able to undo changes which are recently carried out



This will help in the doing upgrade seamlessly and undo changes resume changes as required

https://docs.aws.amazon.com/eks/latest/userguide/sample-deployment.html

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