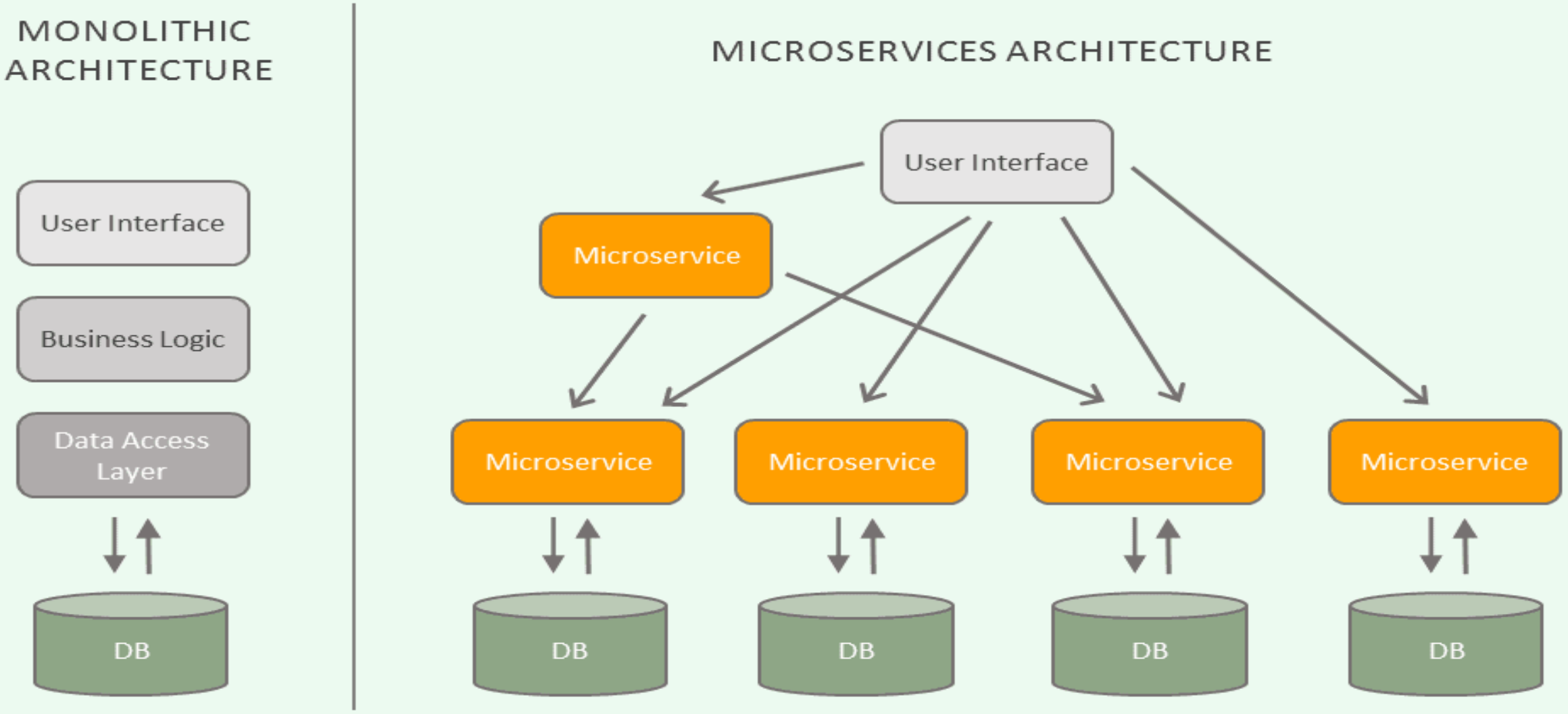
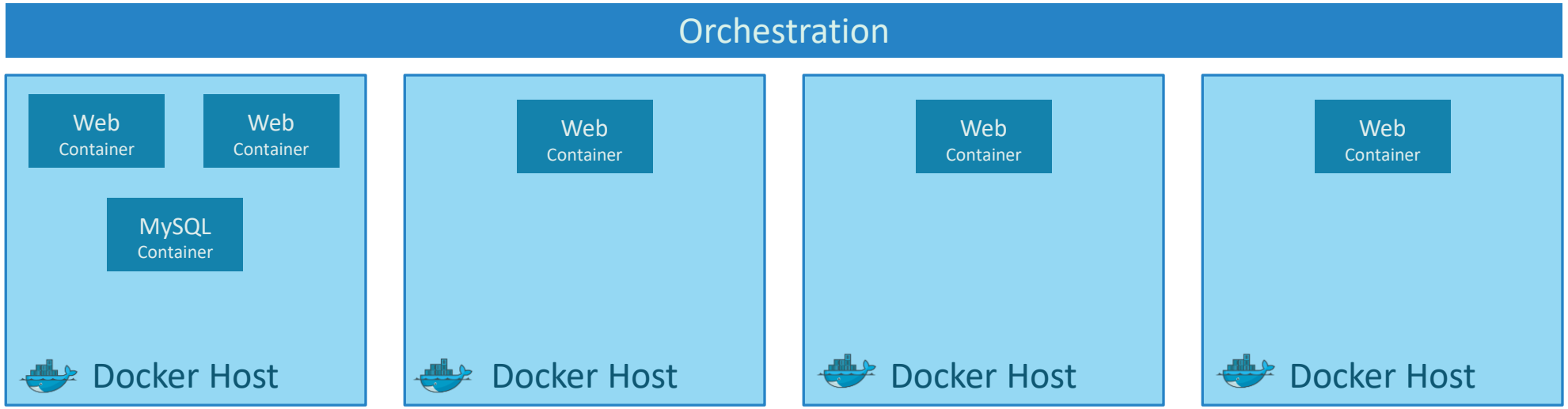

MONOLITHIC ARCHITECTURE
User Interface
Business Logic
Data Access Layer
DB
MICROSERVICES ARCHITECTURE
User Interface
Microservice
Microservice
Microservice
Microservice
Microservice
DB
DB
DB
DB

Container Orchestration

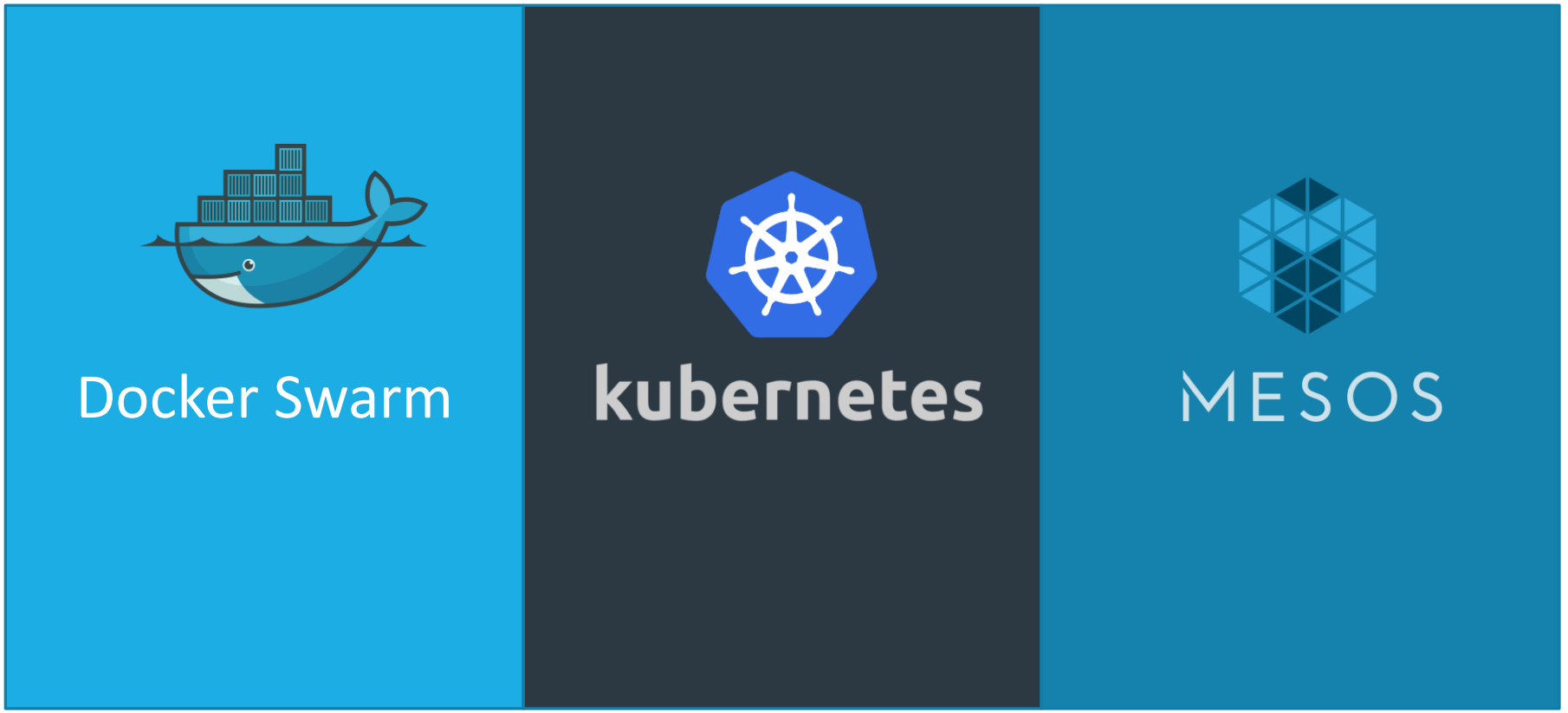
Containers Limitation?

High Availability?
Overlay Network?
Versioning of Application – Rollout, Rollback?
Scaling?
Autoscaling?
Monitoring?
Dependency between containers?

Container orchestration



Orchestration Technologies



What is Kubernetes?

The Kubernetes project was started by Google in 2014.

Kubernetes builds upon a decade and a half of experience that Google has with running production workloads at scale.

Kubernetes can run on a range of platforms, from your laptop, to VMs on a cloud provider, to rack of bare metal servers.

Kubernetes is an open-source platform for automating deployment, scaling, and operations of application containers across clusters of hosts, providing container-centric infrastructure.

portable: with all public, private, hybrid, community cloud

self-healing: auto-placement, auto-restart, auto-replication, auto-scaling

COPYRIGHT © TECHLANDERS SOLUTIONS

Why Kubernetes

Kubernetes can schedule and run application containers on clusters of physical or virtual machines.

host-centric infrastructure to a **container-centric** infrastructure.

Orchestrator

Load balancing

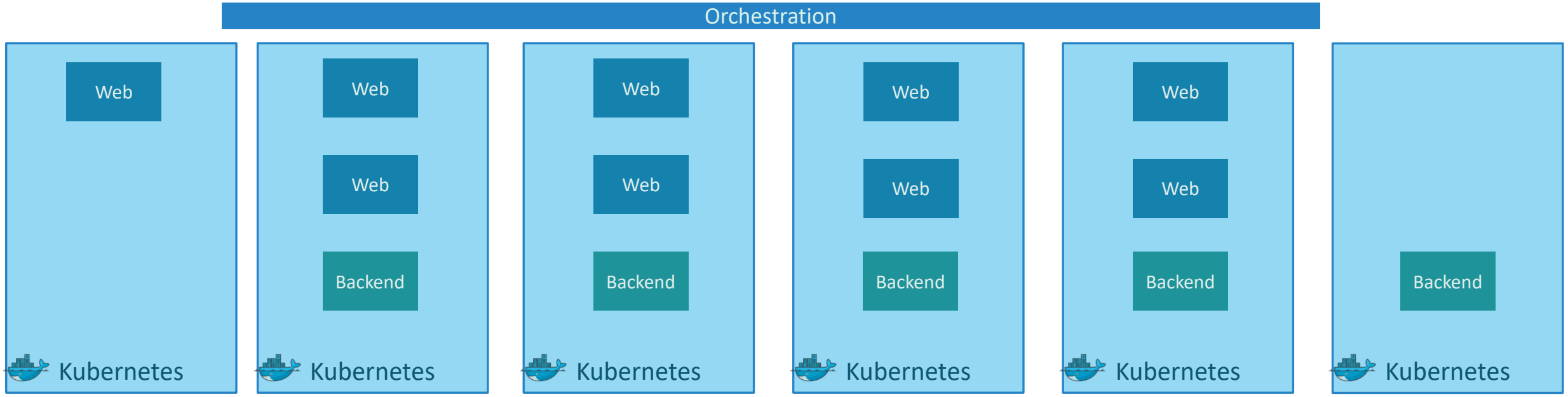
Auto Scaling

Application Health checks

Rolling updates

COPYRIGHT © TECHLANDERS SOLUTIONS

Kubernetes Advantage



TechLanders
Delivering the future

And that is kubernetes..

Setup

Minikube

Kubeadm

Google Cloud Platform

Amazon Web Services

play-with-k8s.com

Setup Kubernetes

Setup - kubeadm

Kubernetes Cluster

A Kubernetes cluster consists of two types of resources:

Master: Which coordinates with the cluster

The Master is responsible for managing the cluster. The master coordinates all activities in your cluster, such as scheduling applications, maintaining applications' desired state, scaling applications, and rolling out new updates.

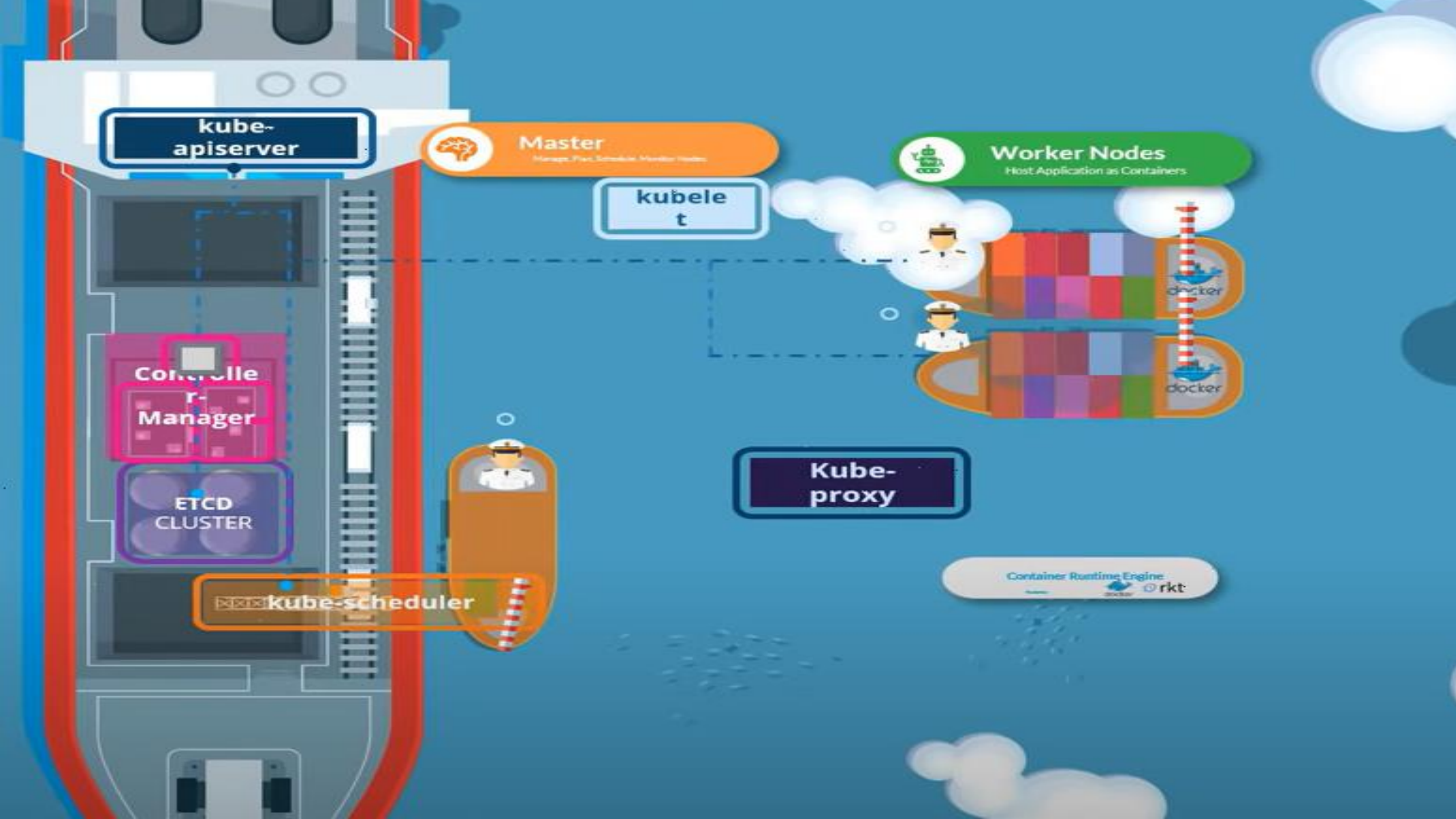
Nodes: Are the workers that run application

A node is a VM or a physical computer that serves as a worker machine in a Kubernetes cluster.

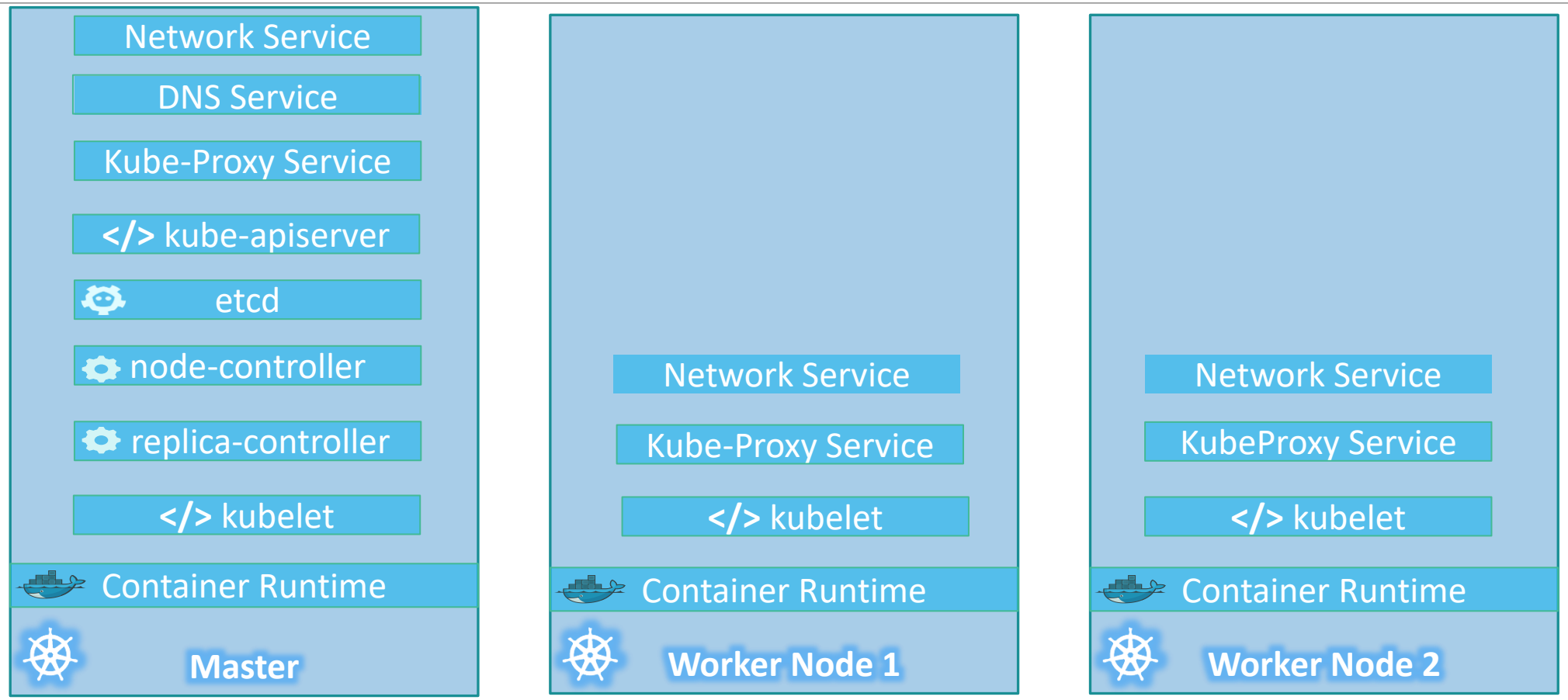
Masters manage the cluster and the nodes are used to host the running applications.

The nodes communicate with the master using the Kubernetes API, which the master exposes.

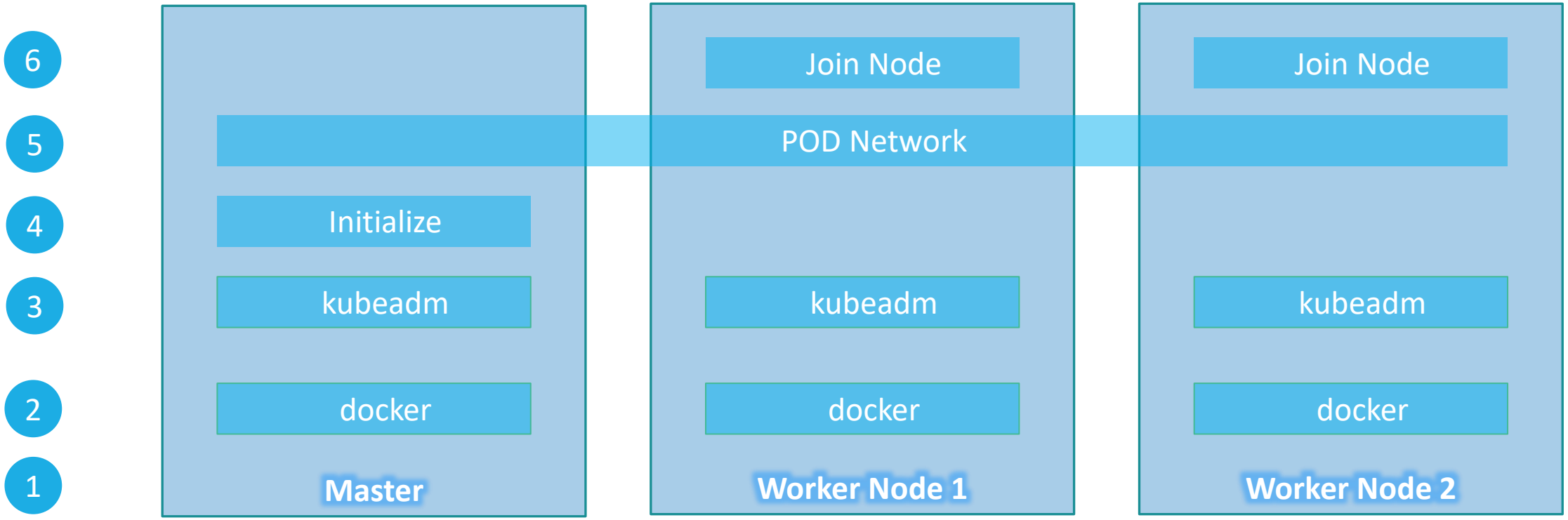
COPYRIGHT © TECHLANDERS SOLUTIONS


kube-
apiserver
Master
kubele
t
Worker Nodes
Host Application as Containers
docker
docker
Controlle
r-
Manager
ETCD
CLUSTER
kube-scheduler
Kube-
proxy
Container Runtime Engine
rkt

kubeadm


Network Service
DNS Service
Kube-Proxy Service
</> kube-apiserver
etcd
node-controller
replica-controller
</> kubelet
Container Runtime
Master
Network Service
Kube-Proxy Service
</> kubelet
Container Runtime
Worker Node 1
Network Service
KubeProxy Service
</> kubelet
Container Runtime
Worker Node 2

Steps


6
5
4
3
2
1
Join Node
Join Node
POD Network
Initialize
kubeadm
kubeadm
kubeadm
docker
docker
docker
Master
Worker Node 1
Worker Node 2

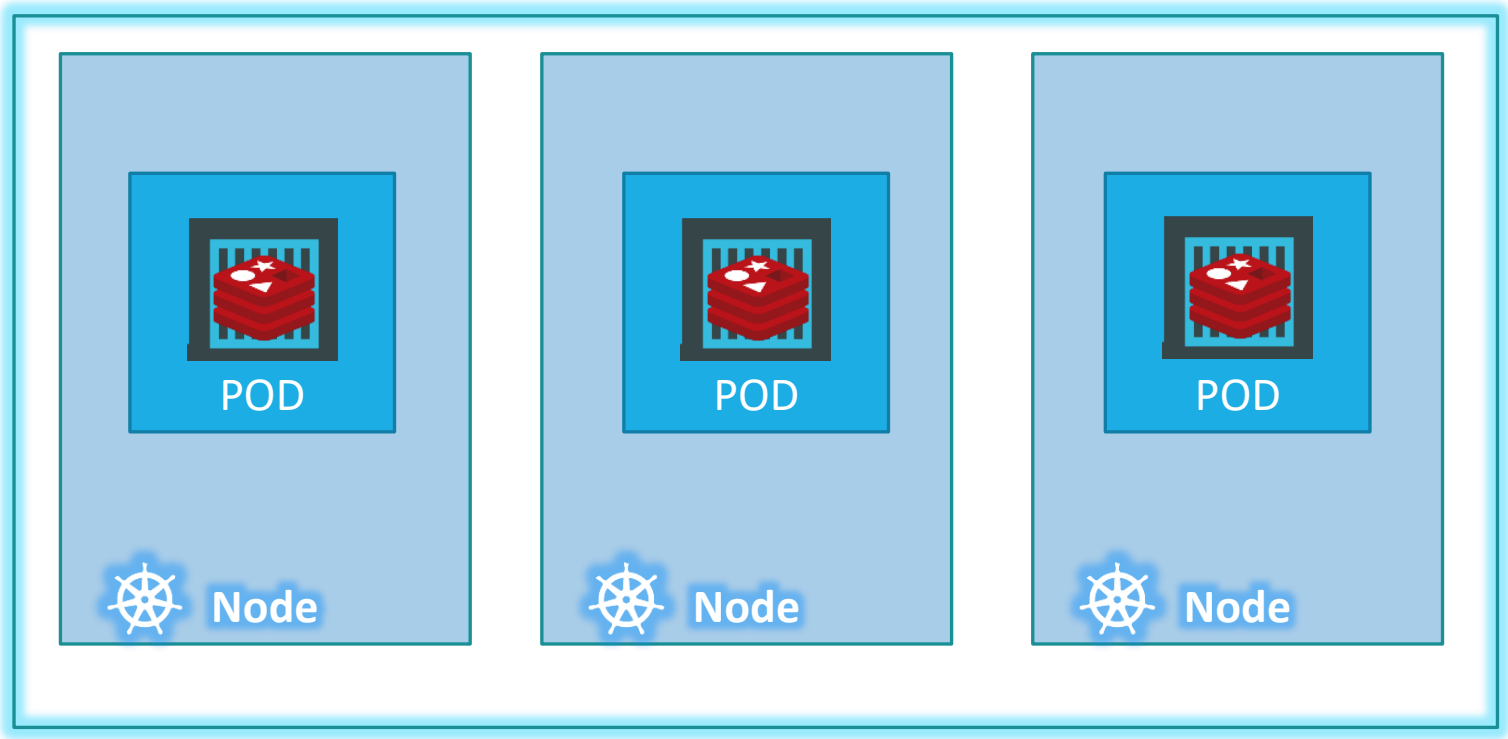
POD

Assumptions

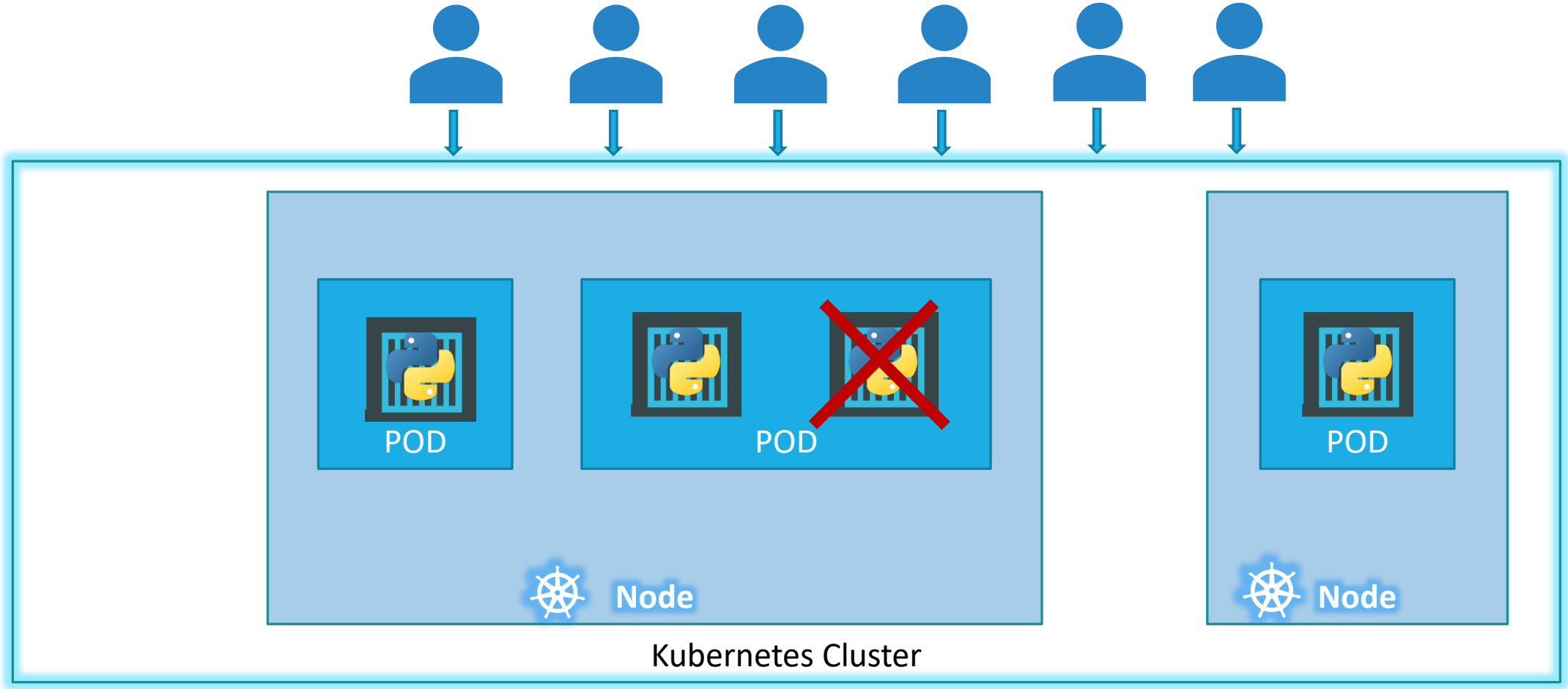
Docker Image

Kubernetes Cluster

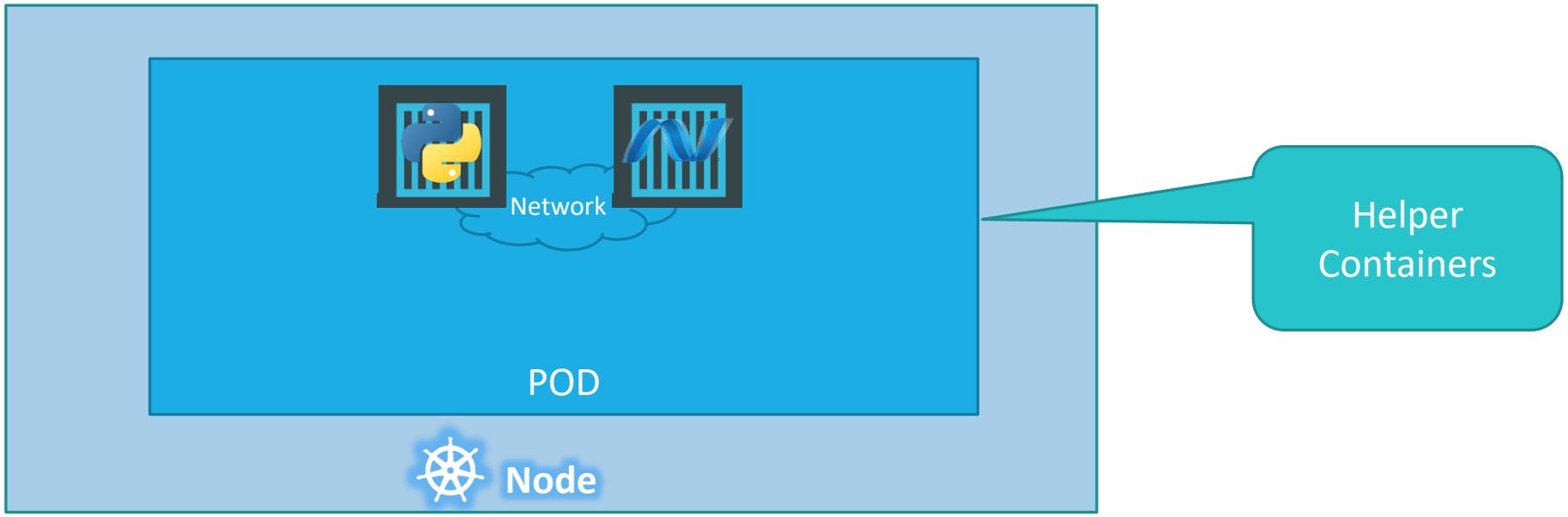
POD



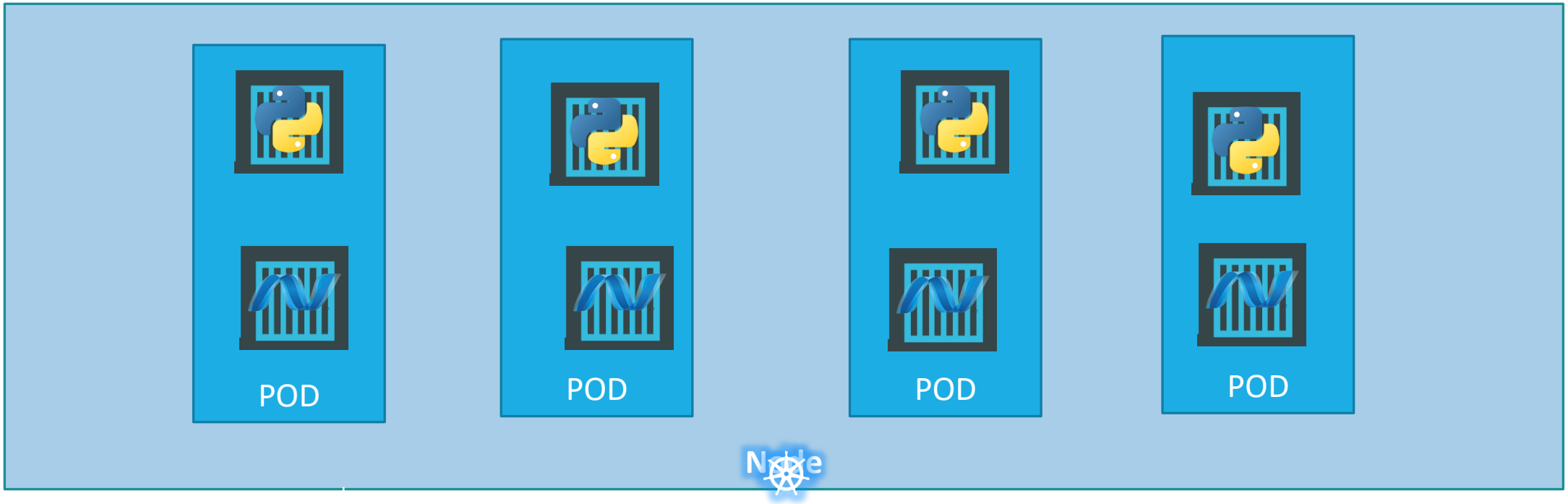
POD



Multi-Container PODs



PODs Again!



Note: I am avoiding networking and load balancing details to keep explanation simple.

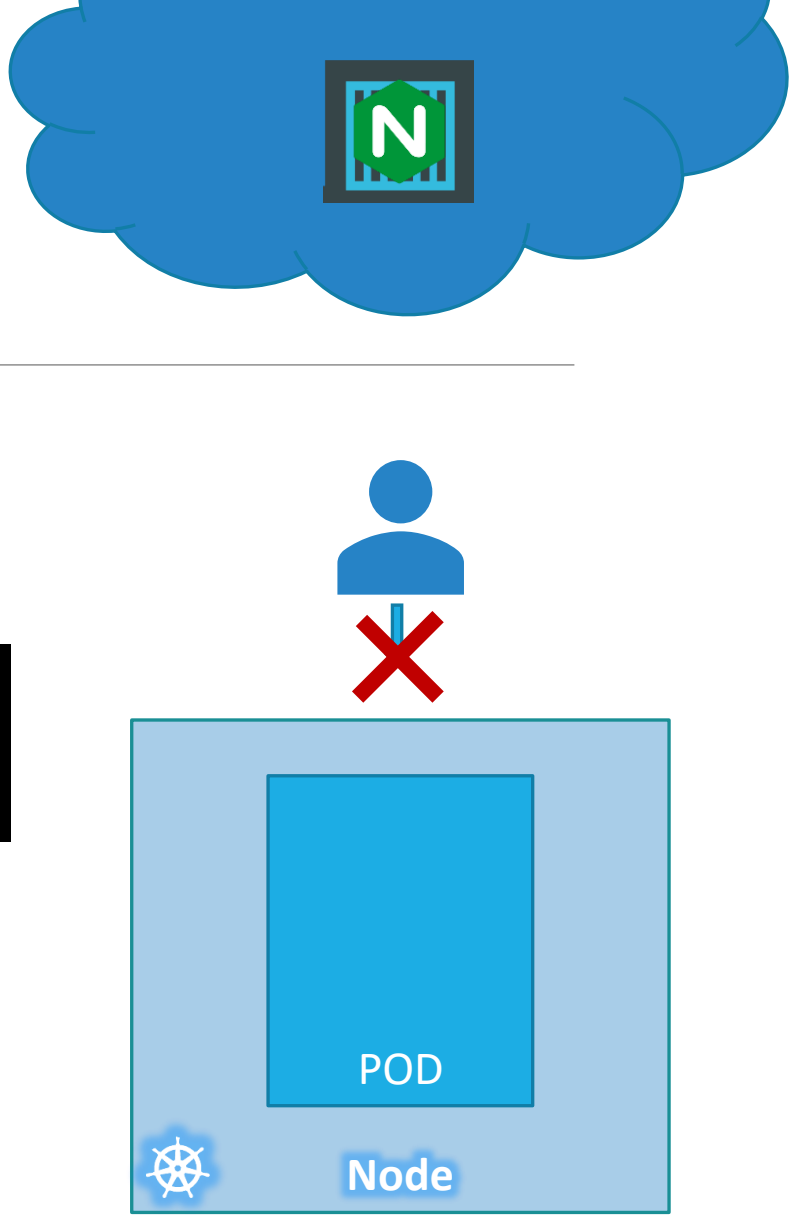
kubectl

```
kubectl run nginx --image nginx
```

```
kubectl get pods
```

```
C:\Kubernetes>kubectl get pods
NAME                     READY     STATUS              RESTARTS   AGE
nginx-8586cf59-whssr     0/1       ContainerCreating   0          3s
```

```
C:\Kubernetes>kubectl get pods
NAME                     READY     STATUS    RESTARTS   AGE
nginx-8586cf59-whssr     1/1       Running   0          8s
```



TechLanders
Delivering the future

YAML Introduction

TechLanders
Delivering the future

POD

With YAML

YAML in Kubernetes

pod-definition.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
    - name: nginx-container
      image: nginx
```

String
String
List/Array
1st Item in List

Kind	Version
POD	v1
Service	v1
ReplicaSet	apps/v1
Deployment	apps/v1

```
kubectl create -f pod-definition.yml
```

Commands

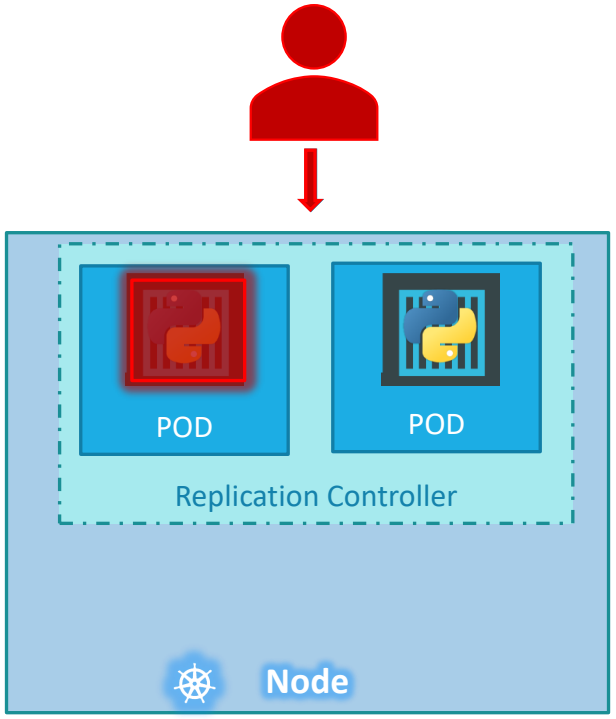
TechLanders
Delivering the future

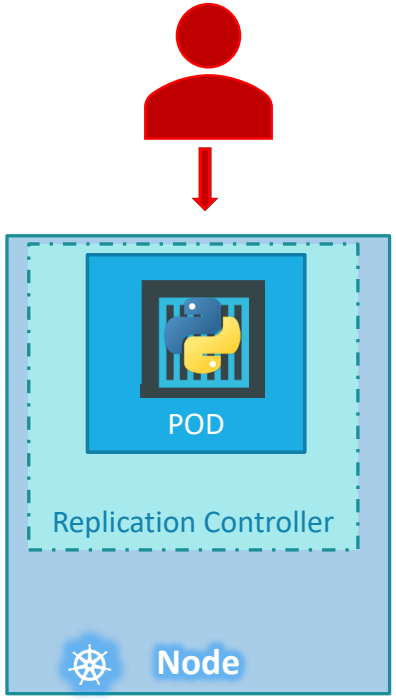
```
> kubectl get pods
NAME            READY     STATUS    RESTARTS   AGE
myapp-pod       1/1       Running   0          20s
```

```
> kubectl describe pod myapp-pod
Name:         myapp-pod
Namespace:    default
Node:         minikube/192.168.99.100
Start Time:   Sat, 03 Mar 2018 14:26:14 +0800
Labels:       app=myapp
              name=myapp-pod
Annotations:  <none>
Status:       Running
IP:           10.244.0.24
Containers:
  nginx:
    Container ID:   docker://830bb56c8c42a86b4bb70e9c1488fae1bc38663e4918b6c2f5a783e7688b8c9d
    Image:          nginx
    Image ID:       docker-pullable://nginx@sha256:4771d09578c7c6a65299e110b3ee1c0a2592f5ea2618d23e4ffe7a4cab1ce5de
    Port:           <none>
    State:          Running
      Started:      Sat, 03 Mar 2018 14:26:21 +0800
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-x95w7 (ro)
Conditions:
  Type           Status
  Initialized    True
  Ready          True
  PodScheduled   True
Events:
  Type    Reason                 Age   From               Message
  ----    ------                 ----  ----               -------
  Normal  Scheduled              34s   default-scheduler  Successfully assigned myapp-pod to minikube
  Normal  SuccessfulMountVolume  33s   kubelet, minikube  MountVolume.SetUp succeeded for volume "default-token-x95w7"
  Normal  Pulling                33s   kubelet, minikube  pulling image "nginx"
  Normal  Pulled                 27s   kubelet, minikube  Successfully pulled image "nginx"
  Normal  Created                27s   kubelet, minikube  Created container
  Normal  Started                27s   kubelet, minikube  Started container
```

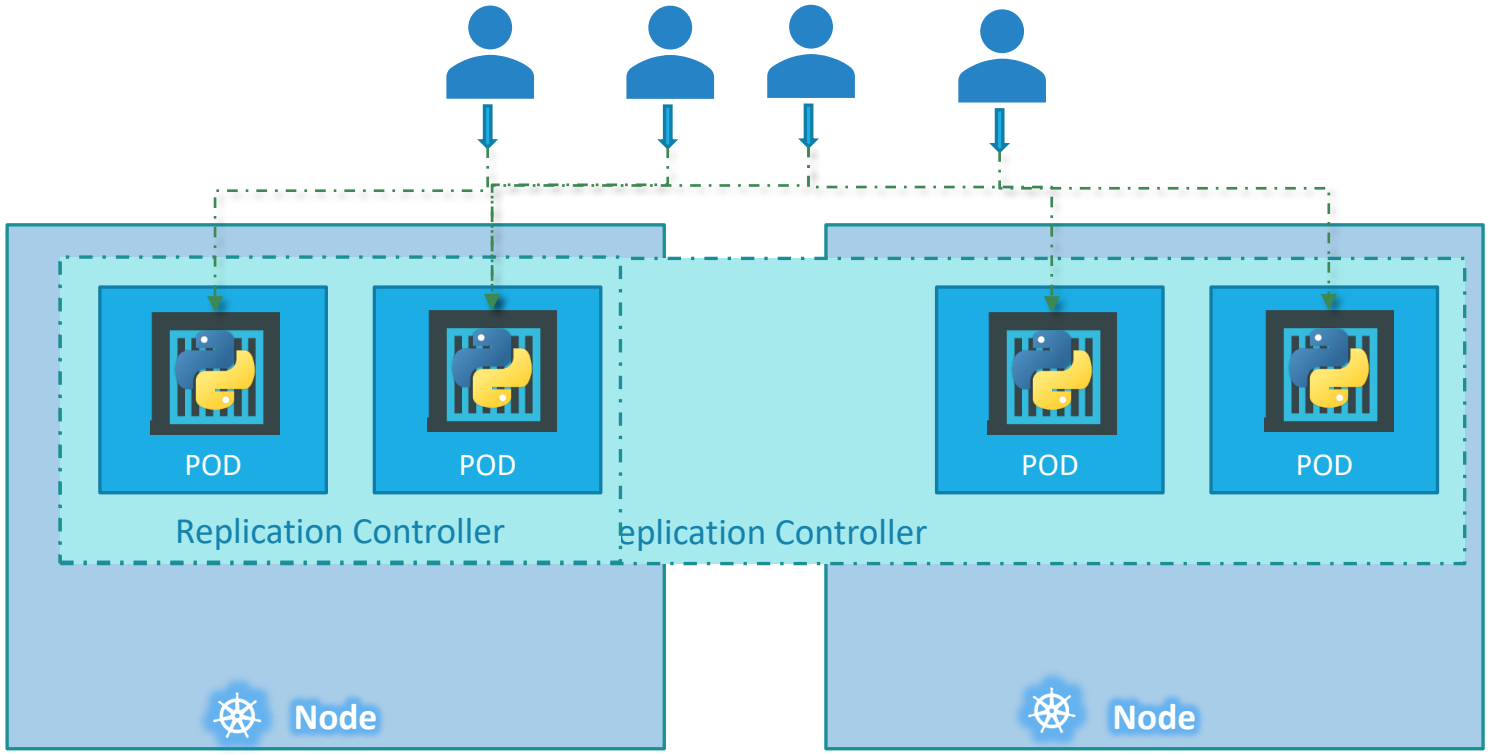
Replication Controller

High Availability





Load Balancing & Scaling



Replication Controller

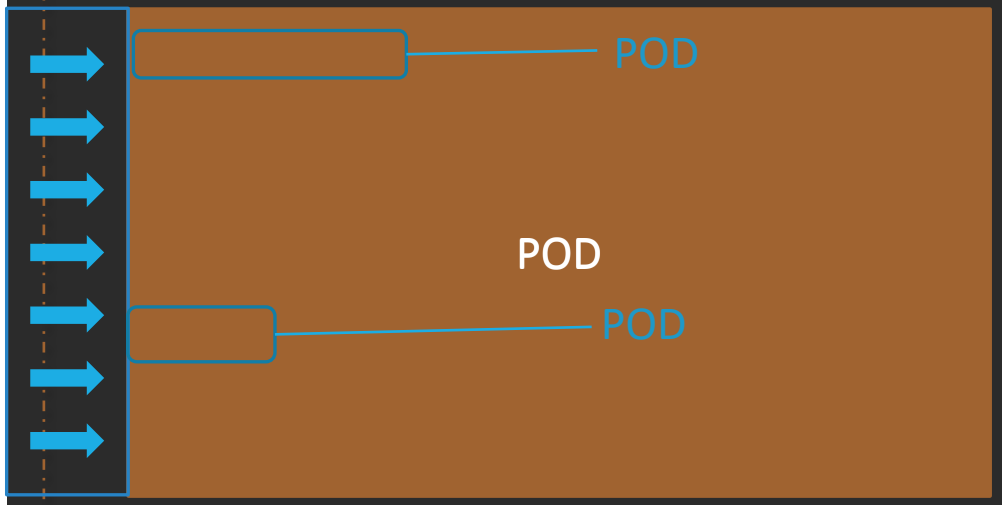
Replica Set

rc-definition.yml

```yaml
apiVersion: v1
kind: ReplicationController
metadata:
  name: myapp-rc
  labels:
      app: myapp
      type: front-end
spec:
  template:
    replicas: 3
```

Replication Controller

Replication Controller



pod-definition.yml

```yaml
apiVersion: v1
kind: Pod

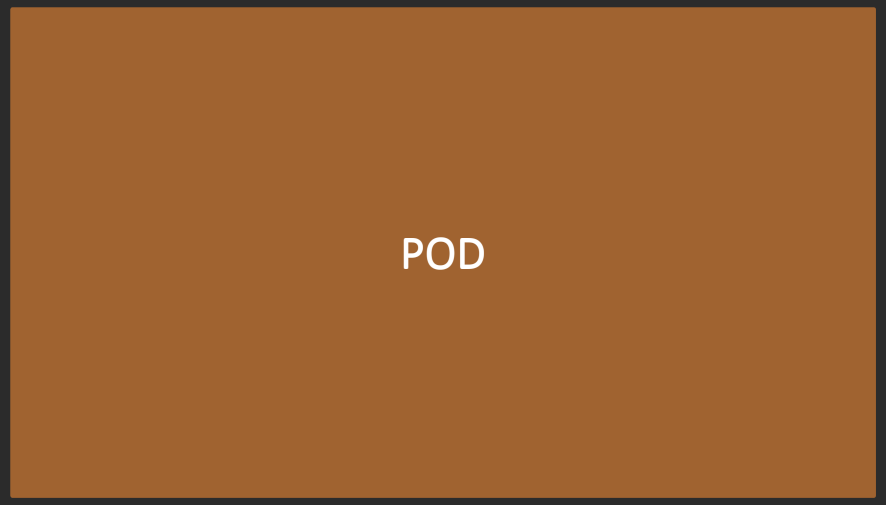
metadata:
  name: myapp-pod
  labels:
     app: myapp
     type: front-end
spec:
  containers:
  - name: nginx-container
    image: nginx
```

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

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

```
> kubectl get pods
NAME             READY     STATUS    RESTARTS   AGE
myapp-rc-4lvk9   1/1       Running   0          20s
myapp-rc-mc2mf   1/1       Running   0          20s
myapp-rc-px9pz   1/1       Running   0          20s
```

replicaset-definition.yml

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: myapp-repl
  labels:
      app: myapp
      type: front-end
spec:
  template:
      
      
  replicas: 3
  selector:
      matchLabels:
          type: front-end
```

```
error: unable to recognize "replicaset-
definition.yml": no matches for /, Kind=ReplicaSet
```

pod-definition.yml

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
      app: myapp
      type: front-end
spec:
  containers:
  - name: nginx-container
    image: nginx
```

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

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

```
> kubectl get pods
NAME                    READY     STATUS    RESTARTS   AGE
myapp-replicaset-9ddl9  1/1       Running   0          45s
myapp-replicaset-9jtpx  1/1       Running   0          45s
myapp-replicaset-hq84m  1/1       Running   0          45s
```

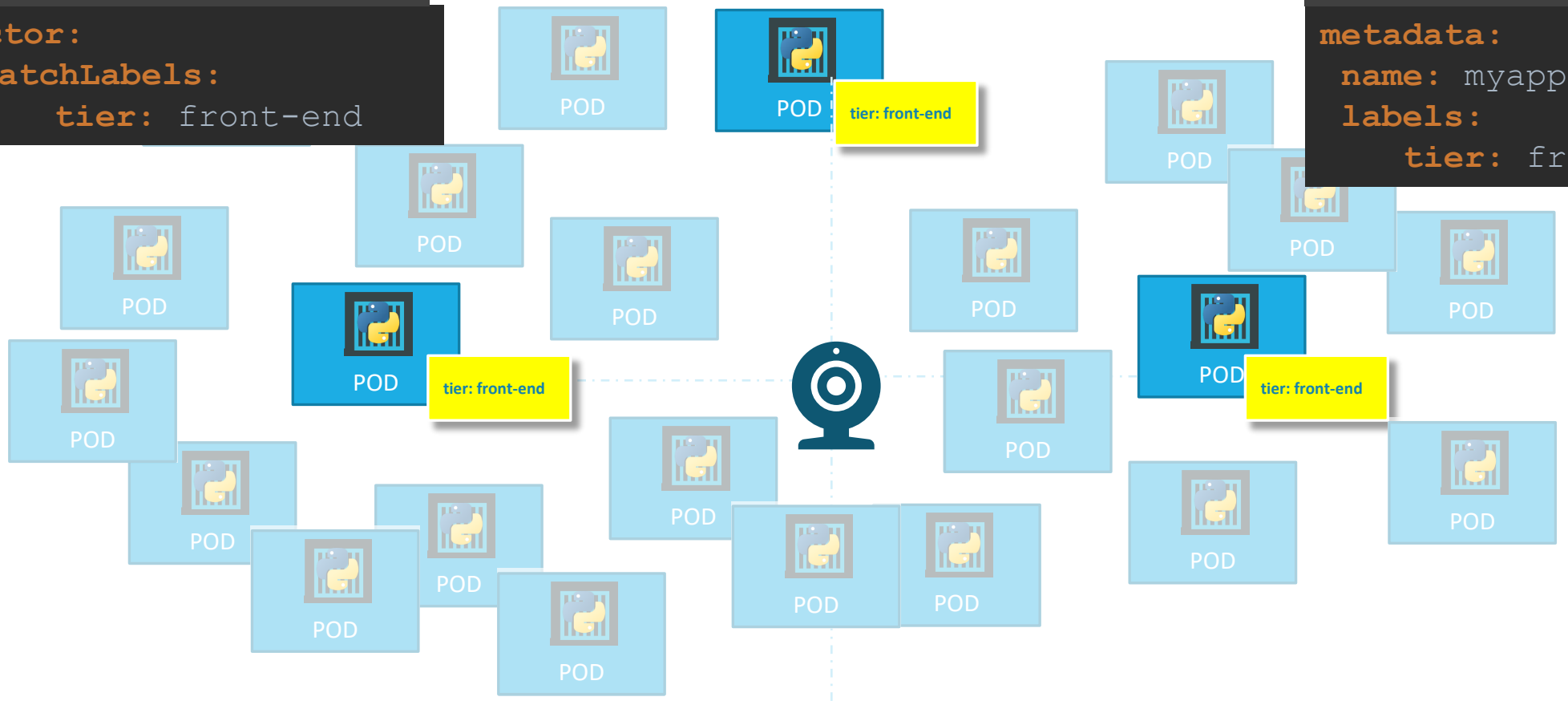
Labels and Selectors

replicaset-definition.yml

```
selector:
    matchLabels:
        tier: front-end
```

pod-definition.yml

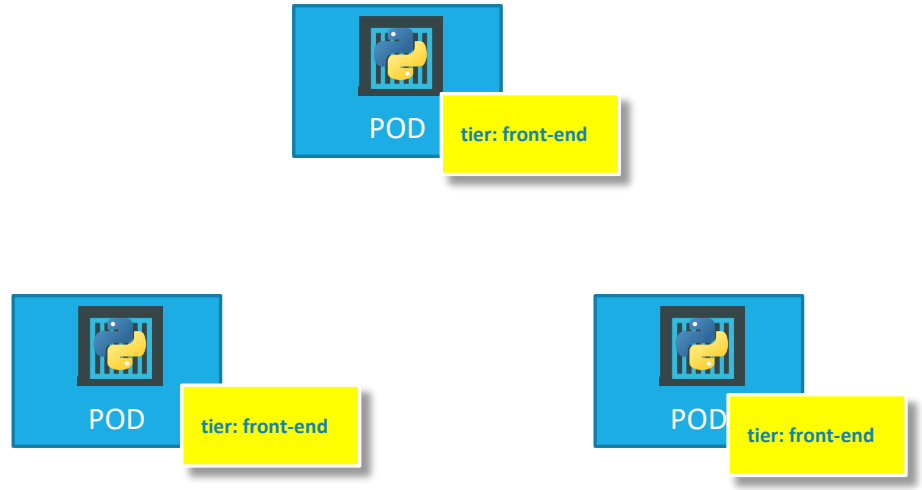
```
metadata:
  name: myapp-pod
  labels:
    tier: front-end
```



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
```

Template



Scale

```
> kubectl replace -f replicaset-definition.yml
```

```
> kubectl scale --replicas=6 -f replicaset-definition.yml
```

```
> kubectl scale --replicas=6 replicaset myapp-replicaset
```

TYPE NAME

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: 6
  selector:
    matchLabels:
      type: front-end
```

TechLanders

commands

```
> kubectl create –f replicaset-definition.yml
```

```
> kubectl get replicaset
```

```
> kubectl delete replicaset myapp-replicaset
```

*Also deletes all underlying PODs

```
> kubectl replace -f replicaset-definition.yml
```

```
> kubectl scale –replicas=6 -f replicaset-definition.yml
```

Demo

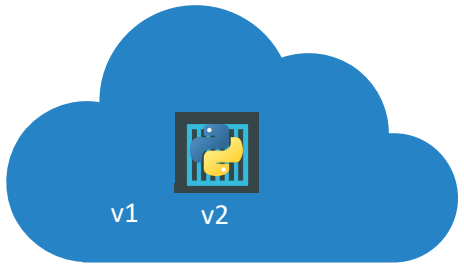
ReplicaSet

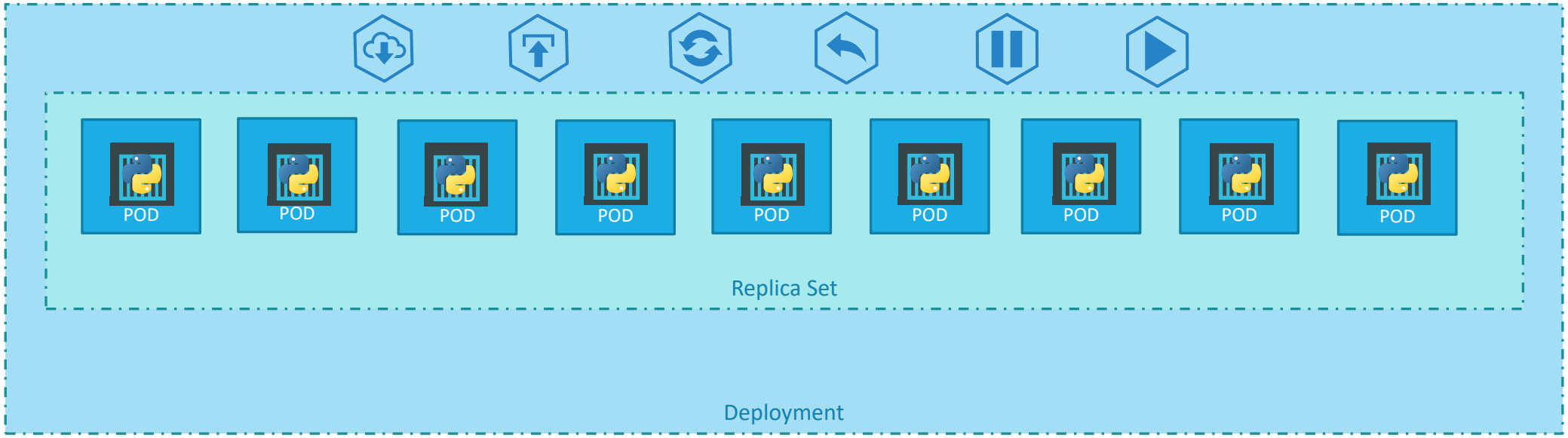
ReplicaSet as an Horizontal Pod Autoscaler Target

https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/#replicaset-as-an-horizontal-pod-autoscaler-target

Deployment

Deployment


v1
v2


POD
POD
POD
POD
POD
POD
POD
POD
POD
Replica Set
Deployment

Definition

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

```
> kubectl get deployments
NAME              DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
myapp-deployment  3         3         3            3           21s
```

```
> kubectl get replicaset
NAME                         DESIRED   CURRENT   READY     AGE
myapp-deployment-6795844b58  3         3         3         2m
```

```
> kubectl get pods
NAME                               READY     STATUS    RESTARTS   AGE
myapp-deployment-6795844b58-5rbjl  1/1       Running   0          2m
myapp-deployment-6795844b58-h4w55  1/1       Running   0          2m
myapp-deployment-6795844b58-lfjhv  1/1       Running   0          2m
```

deployment-definition.yml

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: myapp-deployment
  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
```

TechLanders
Delivering the future

commands

```
> kubectl get all
NAME                     DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
deploy/myapp-deployment   3         3         3            3           9h

NAME                               DESIRED   CURRENT   READY     AGE
rs/myapp-deployment-6795844b58   3         3         3         9h

NAME                                   READY     STATUS    RESTARTS   AGE
po/myapp-deployment-6795844b58-5rbjl   1/1       Running   0          9h
po/myapp-deployment-6795844b58-h4w55   1/1       Running   0          9h
po/myapp-deployment-6795844b58-lfjhv   1/1       Running   0          9h
```

Demo

Deployment

Demo

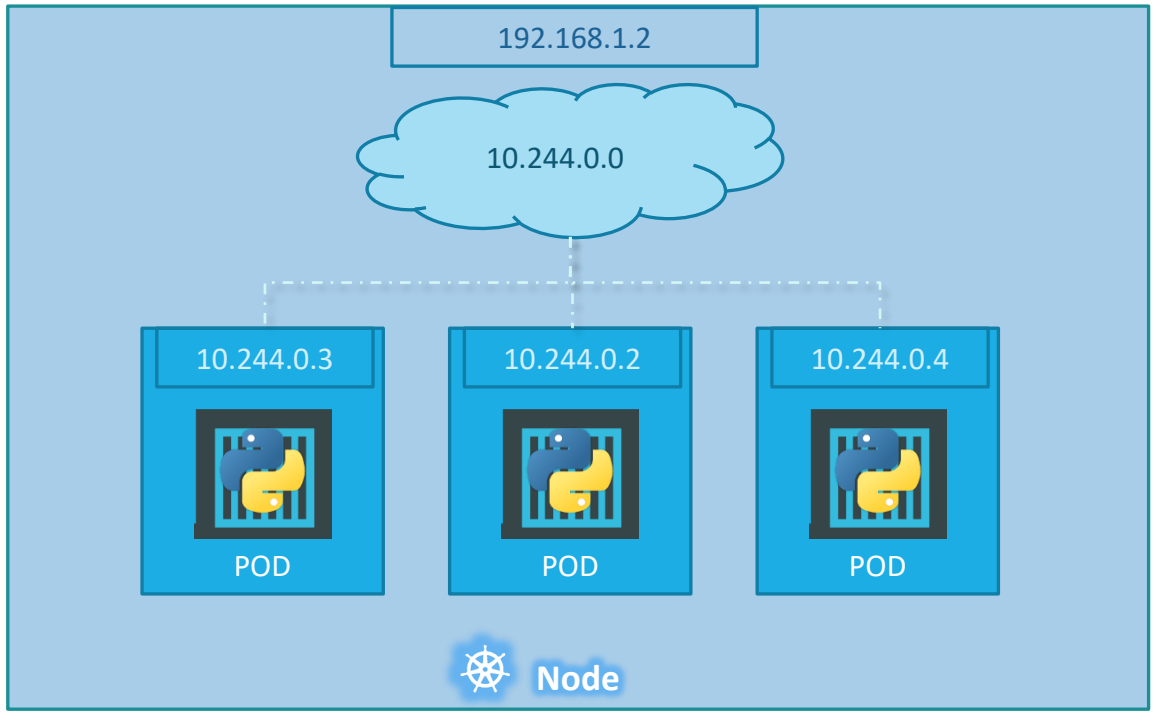
Deployment

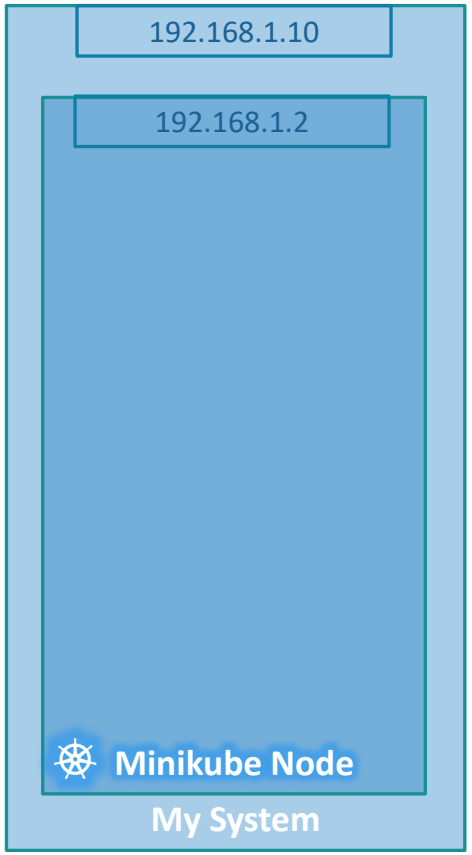
Networking 101

Raman Khanna

Kubernetes Networking - 101

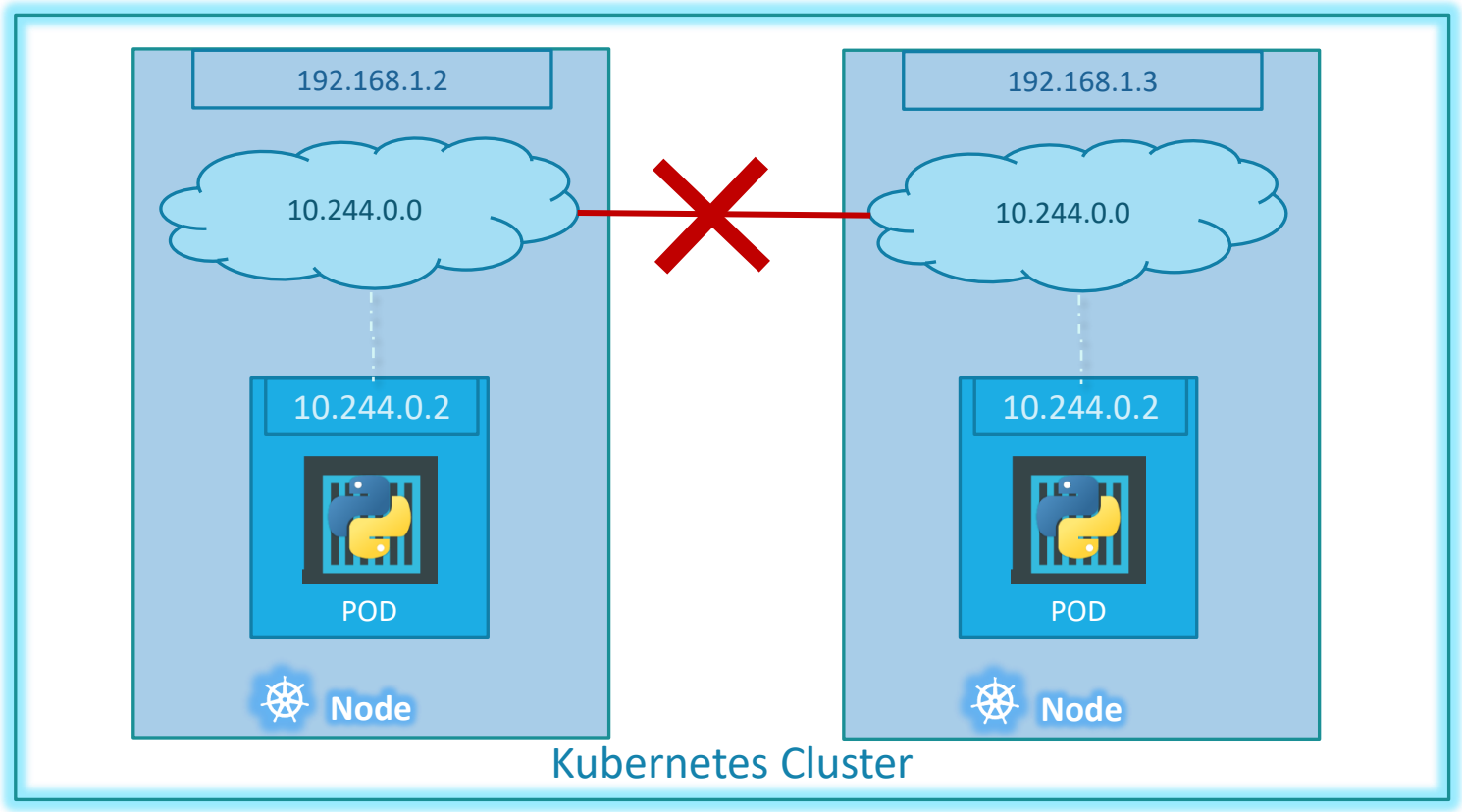
- IP Address is assigned to a POD





Cluster Networking

- All containers/PODs can communicate to one another without NAT
- All nodes can communicate with all containers and vice-versa without NAT

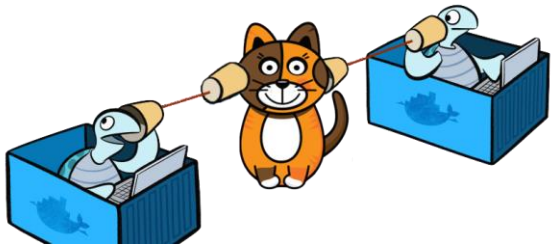


CISCO

cilium

flannel

vmware
NSX



Cluster Networking Setup

(3/4) Installing a pod network

You **MUST** install a pod network add-on so that your pods can communicate with each other.

The network must be deployed before any applications. Also, kube-dns, an internal helper service, will not start up before a network is installed. kubeadm only supports Container Network Interface (CNI) based networks (and does not support kubenet).

Several projects provide Kubernetes pod networks using CNI, some of which also support Network Policy. See the add-ons page for a complete list of available network add-ons. IPv6 support was added in CNI v0.6.0. CNI bridge and local-ipam are the only supported IPv6 network plugins in 1.9.

Note: kubeadm sets up a more secure cluster by default and enforces use of RBAC. Please make sure that the network manifest of choice supports RBAC.

You can install a pod network add-on with the following command:

```
kubectl apply -f <add-on.yaml>
```

NOTE: You can install **only one** pod network per cluster.

Choose one... | Calico | Canal | Flannel | Kube-router | Romana | Weave Net

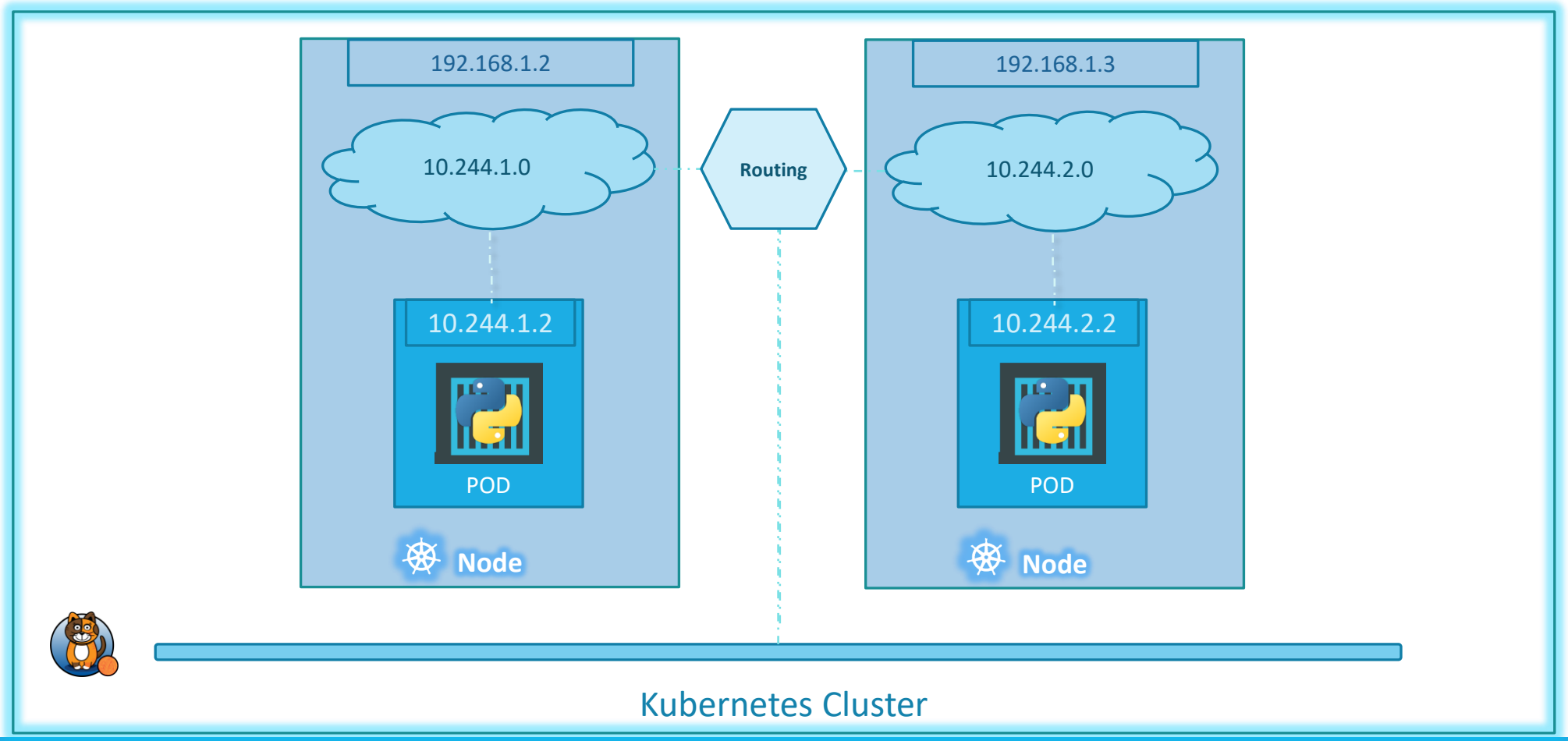
Refer to the Calico documentation for a kubeadm quickstart, a kubeadm installation guide, and other resources.

Note:

- In order for Network Policy to work correctly, you need to pass `--pod-network-cidr=192.168.0.0/16` to `kubeadm init`.
- Calico works on `amd64` only.

```
kubectl apply -f https://docs.projectcalico.org/v3.0/getting-started/kubernetes/installation/hosted/kubeadm/1.7/calico.yaml
```

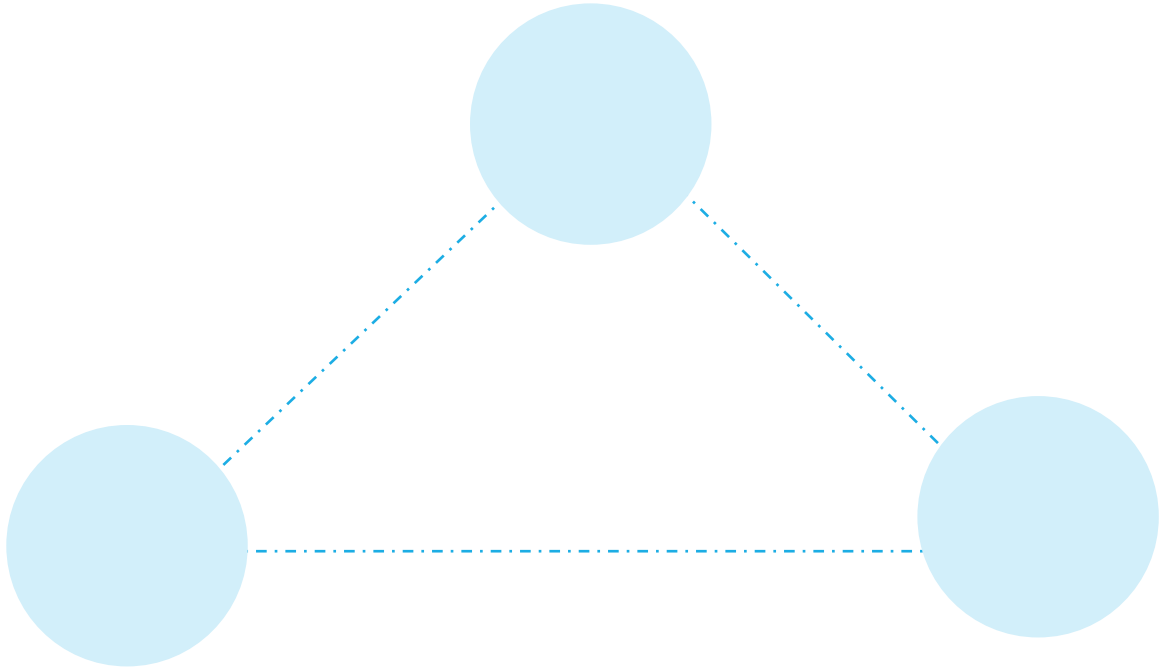
Cluster Networking



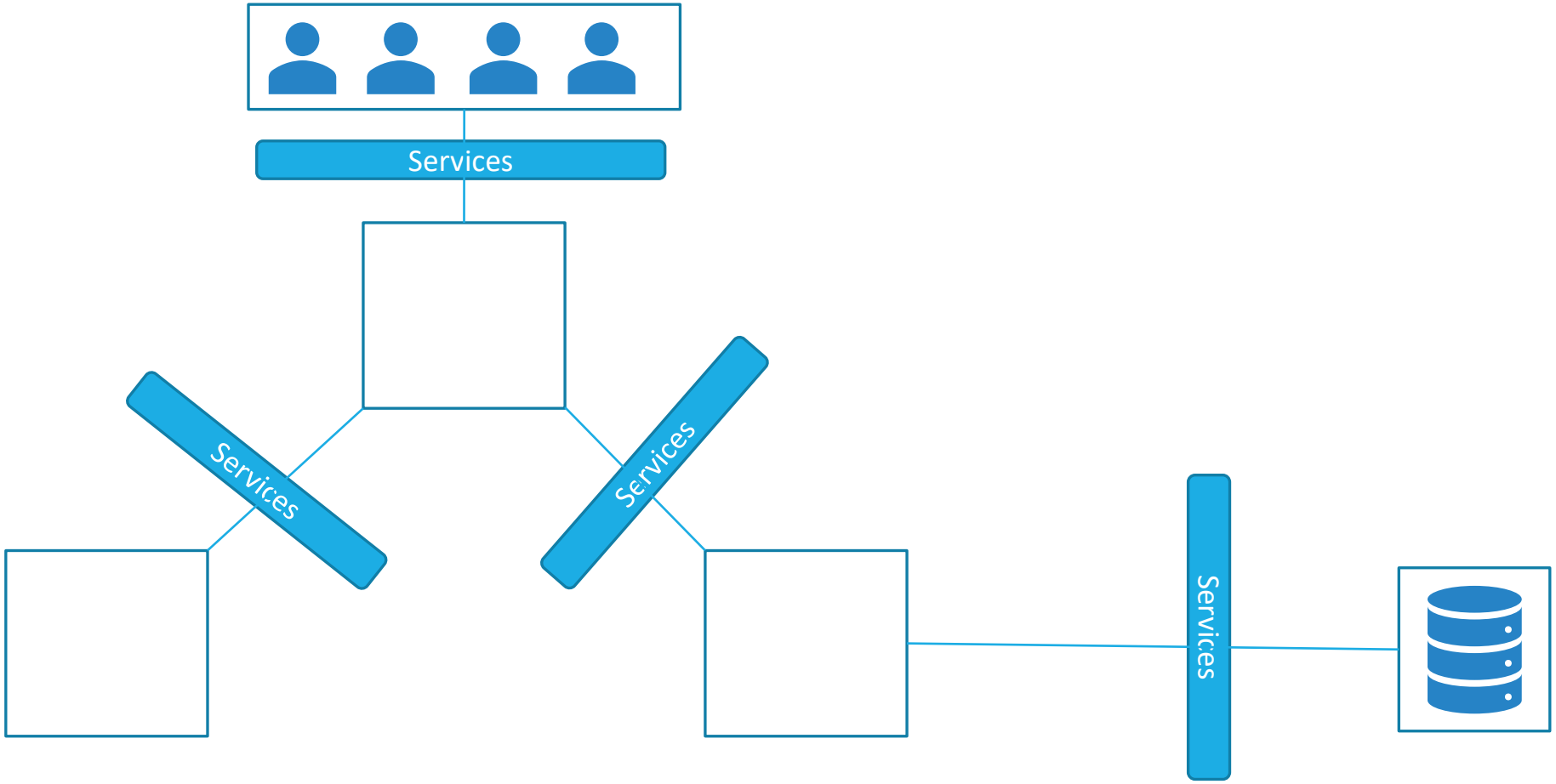
Demo

Networking

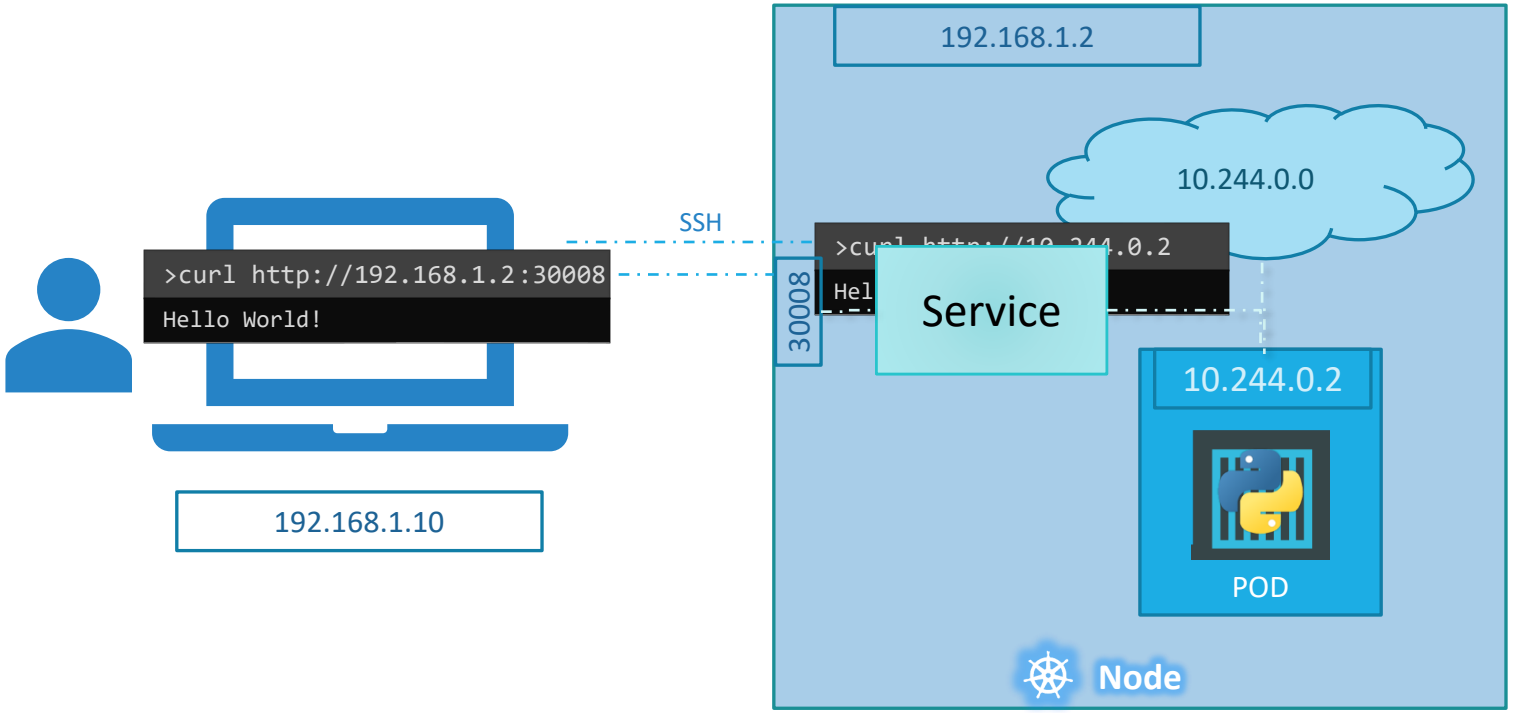
Services



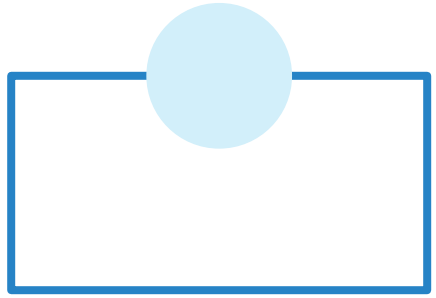
Services



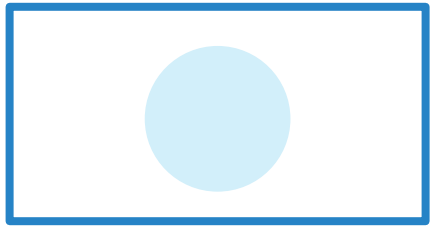
Service


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

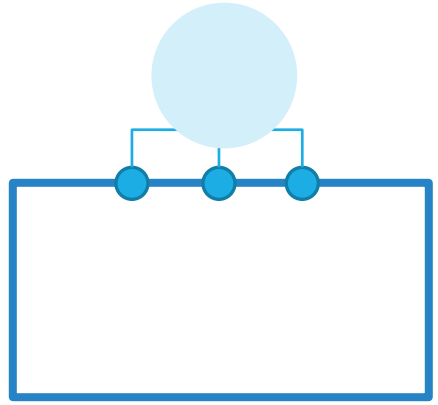
Services Types



NodePort

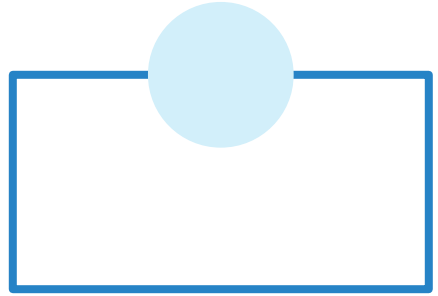


ClusterIP

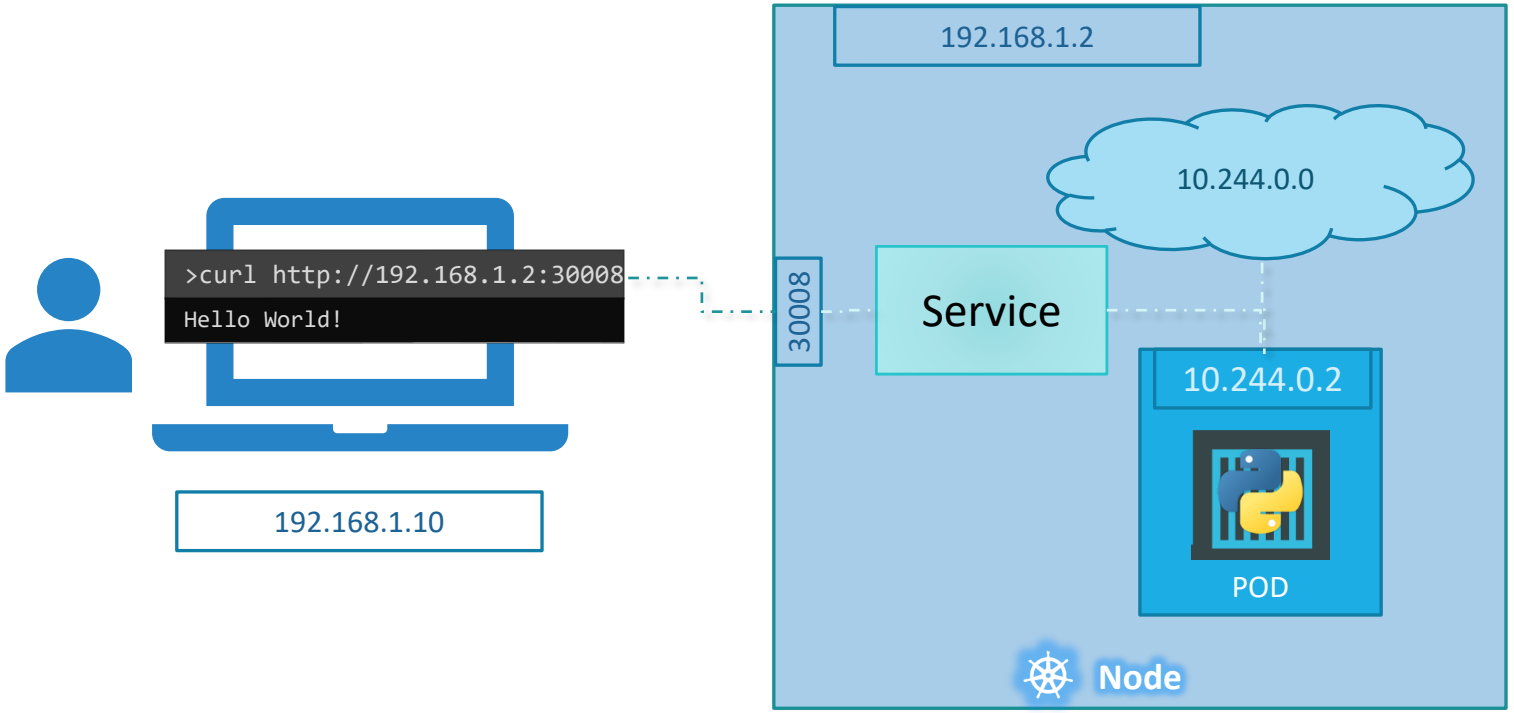


LoadBalancer

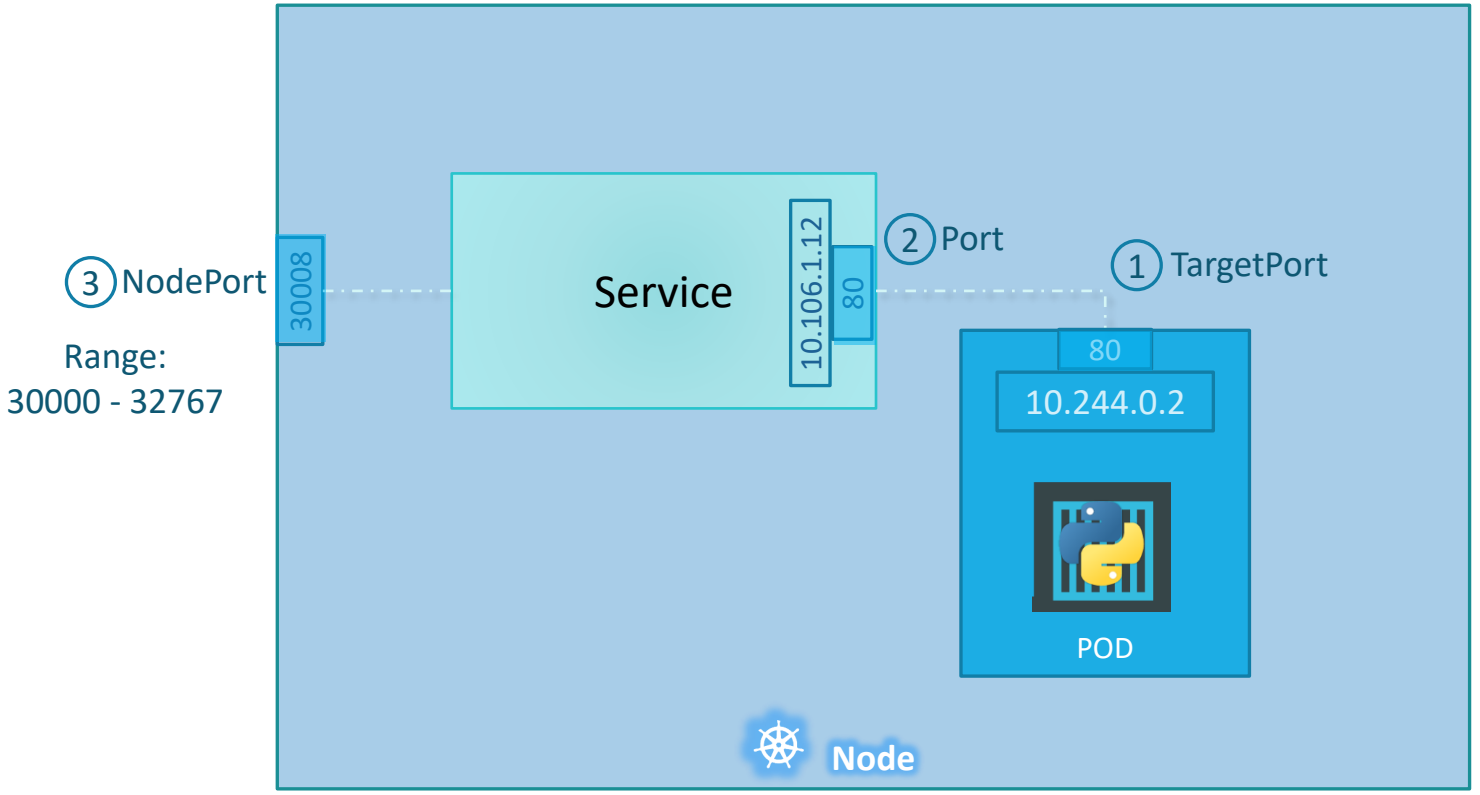
NodePort



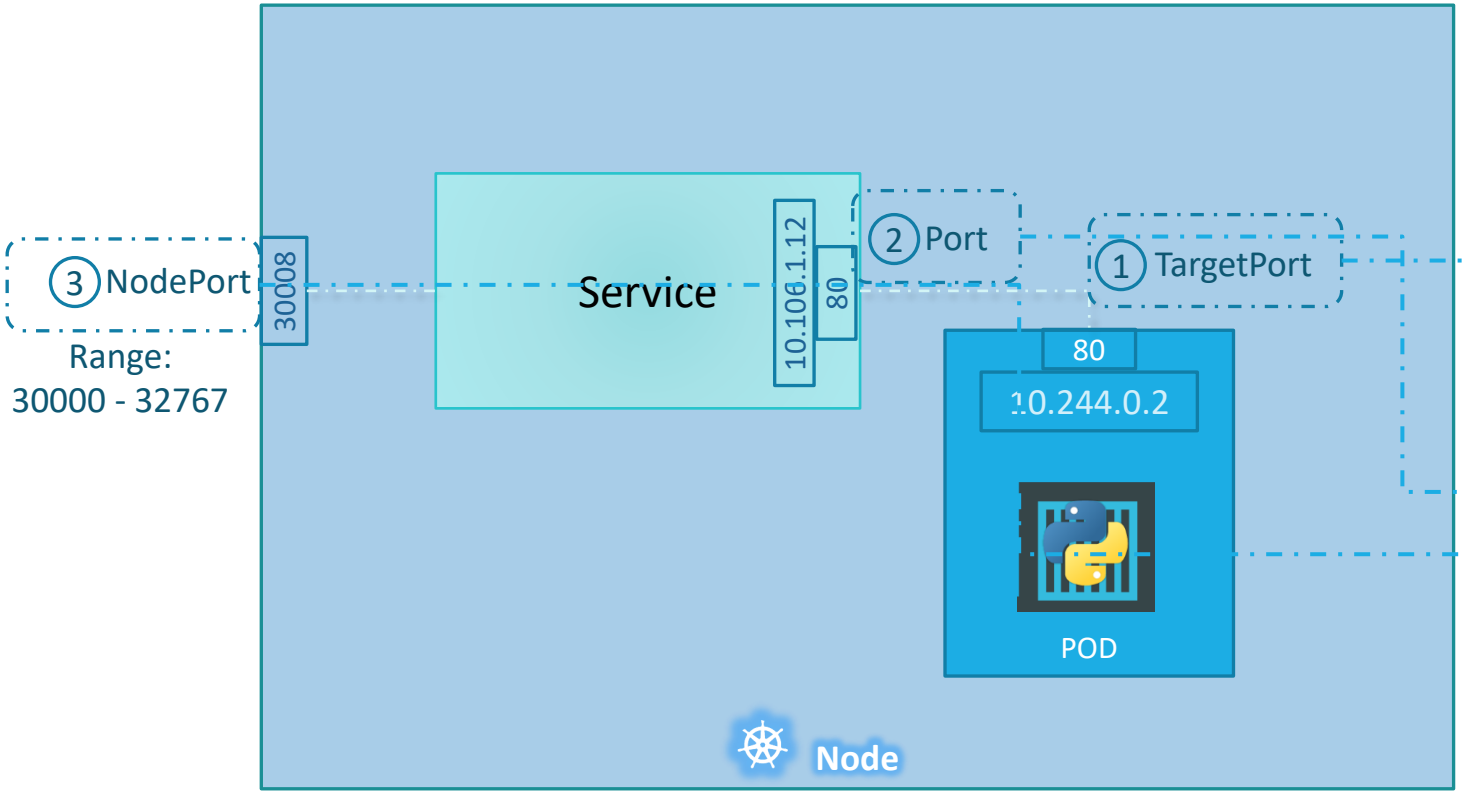
Service - NodePort



Service - NodePort



Service - NodePort



service-definition.yml

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

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

TechLanders
Delivering the future

Service - NodePort

service-definition.yml

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

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

pod-definition.yml

```
        app: myapp
```

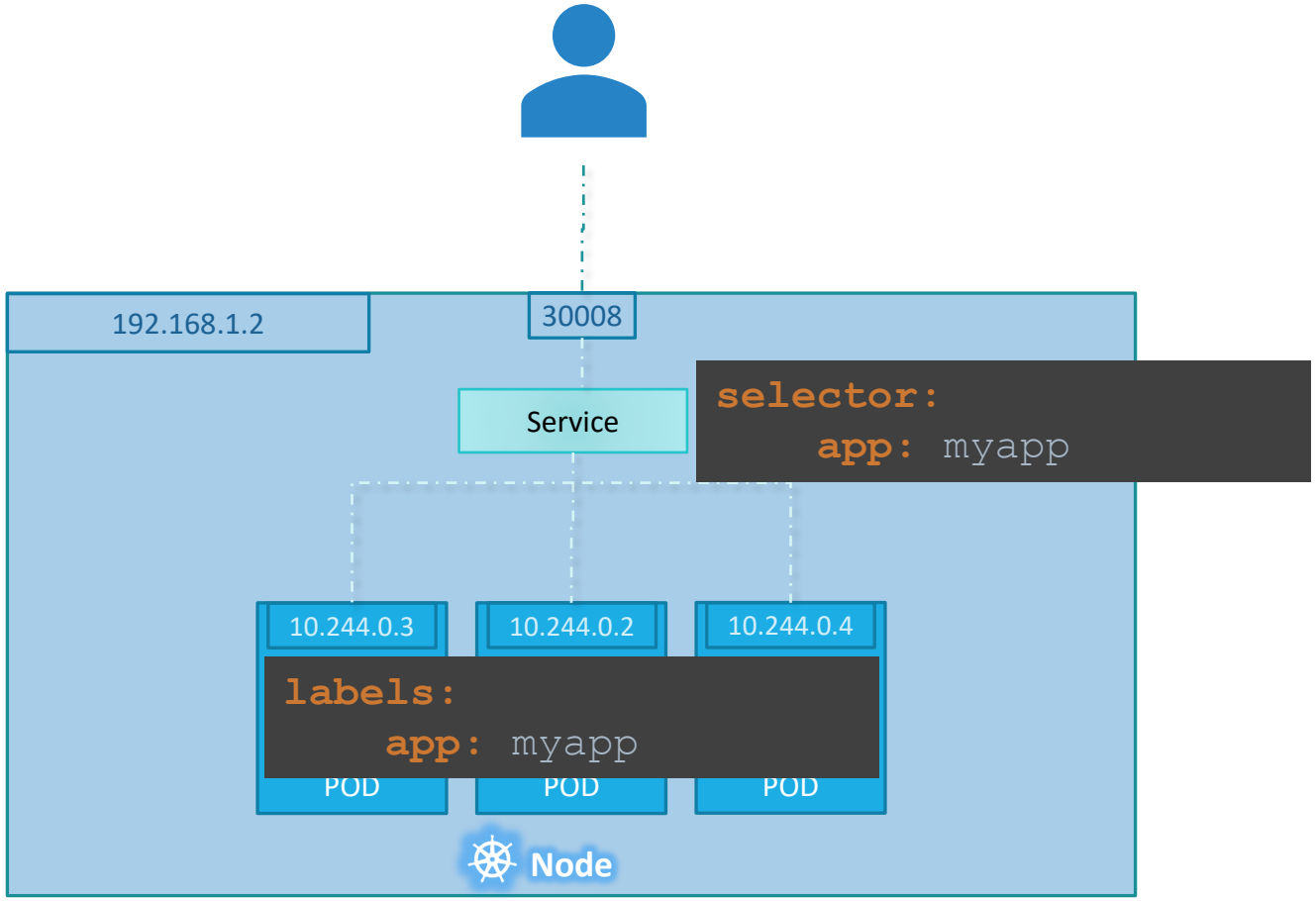
```
> kubectl create -f service-definition.yml
service "myapp-service" created
```

```
> kubectl get services
NAME            TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes      ClusterIP   10.96.0.1        <none>        443/TCP        16d
myapp-service   NodePort    10.106.127.123   <none>        80:30008/TCP   5m
```

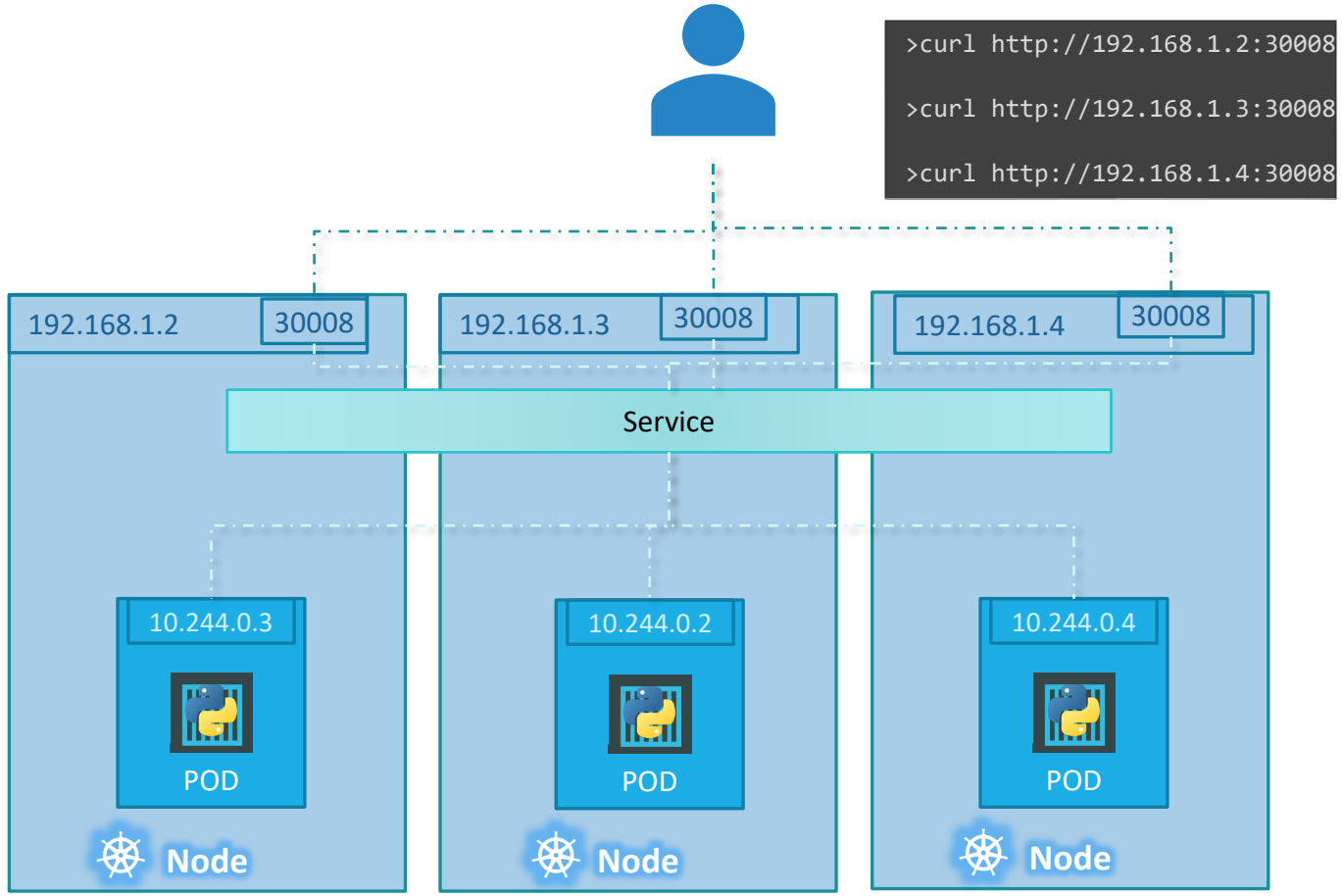
```
> curl http://192.168.1.2:30008
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
```

Service - NodePort

Algorithm: Random
SessionAffinity: Yes



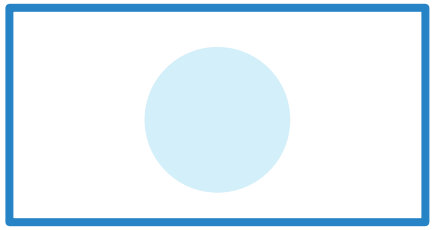
Service - NodePort



Demo

Service - NodePort

ClusterIP

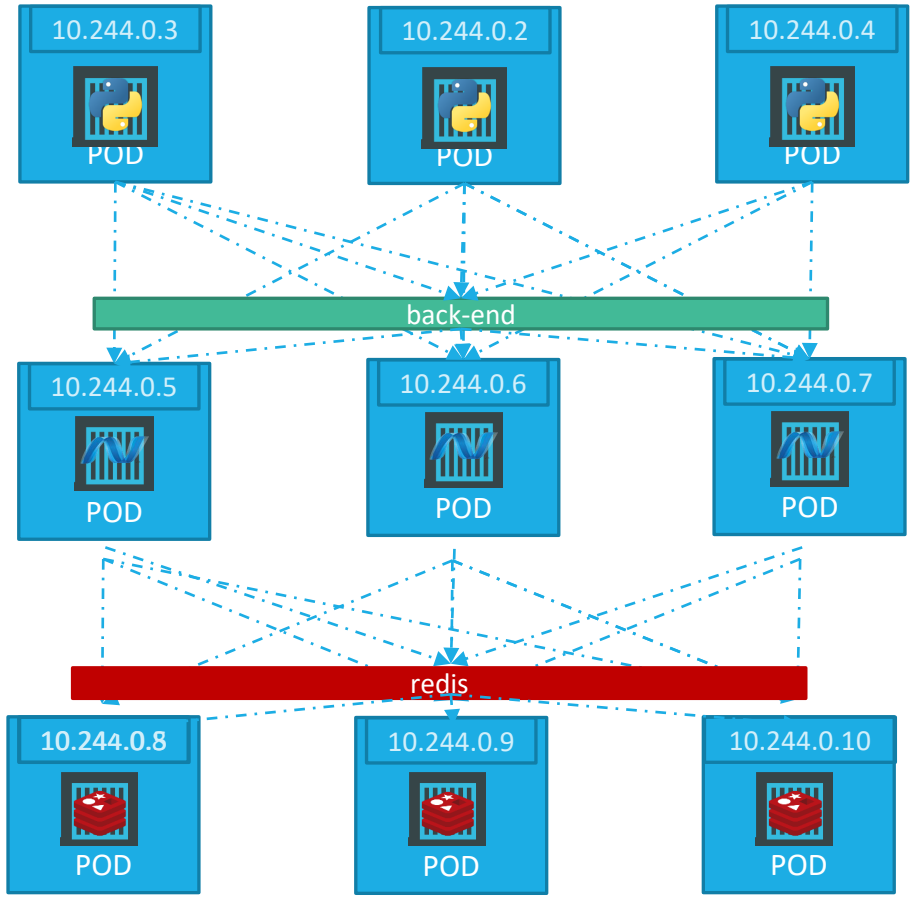


ClusterIP

front-end

back-end

redis



service-definition.yml

```yaml
apiVersion: v1
kind: Service
metadata:
    name: back-end

spec:
    type: ClusterIP
    ports:
     - targetPort: 80
       port: 80

    selector:
```

pod-definition.yml

```yaml
      app: myapp
      type: back-end
spec:
  containers:
  - name: nginx-container
    image: nginx
```

```
> kubectl create -f service-definition.yml
service “back-end" created
```

```
> kubectl get services
NAME          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
kubernetes    ClusterIP   10.96.0.1        <none>        443/TCP    16d
back-end      ClusterIP   10.106.127.123   <none>        80/TCP     2m
```

Demo

Service - NodePort

References

https://kubernetes.io/docs/concepts/services-networking/dns-pod-service/