SOFTWARE DEVELOPMENT TOOLS AND ENVIRONMENTS

minikube

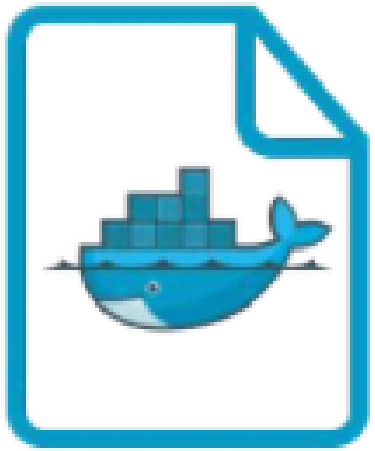
minikube

- Created to learn
- Local Kubernetes Cluster
- Recognition:
  - Cloud Native Foundation
  - The Linux Fondation

minikube

Dockerfile

- Dockerfile is instructions to build Docker Image
  - How to run commands
  - Add files or directories
  - Create environment variables
  - What process to run when launching container
- Result from building Dockerfile is Docker Image



Dockerfile

Sample Dockerfile

```
FROM node:14.15.0-alpine3.12
COPY . /nodejs/.
WORKDIR /nodejs
RUN npm install
ENV VERSION 1.0
EXPOSE 8081
CMD ["node", "/nodejs/main.js"]
```

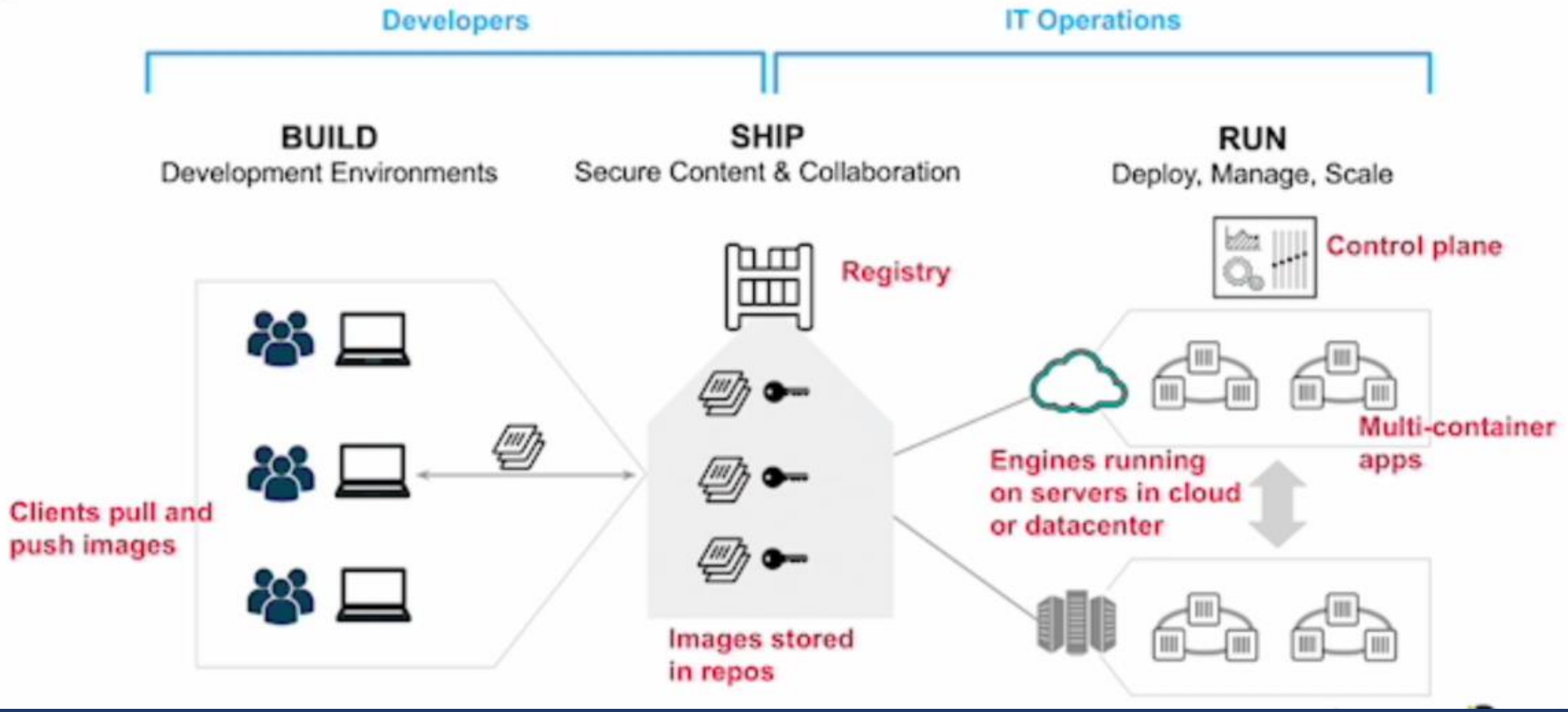
OS + System Packages

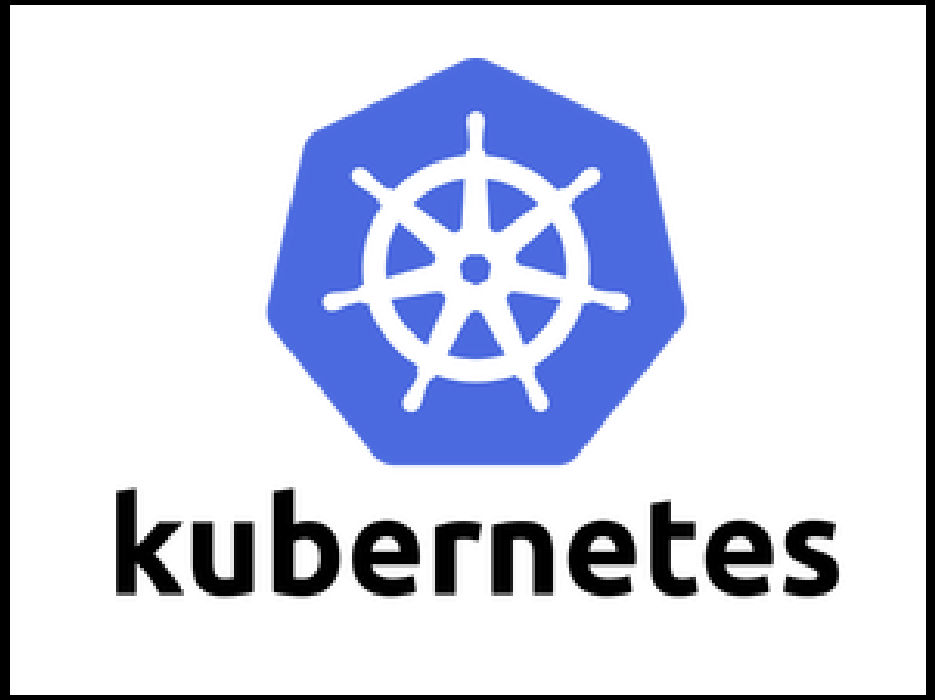
Source Code

Library Dependencies

Configuration

Docker and DevOps





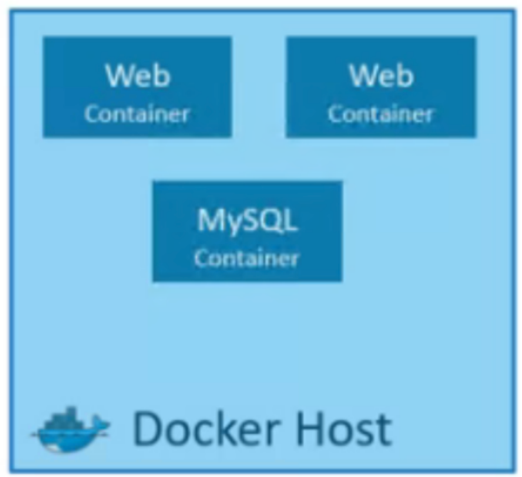
Introduction to Kubernetes

What is Kubernetes?

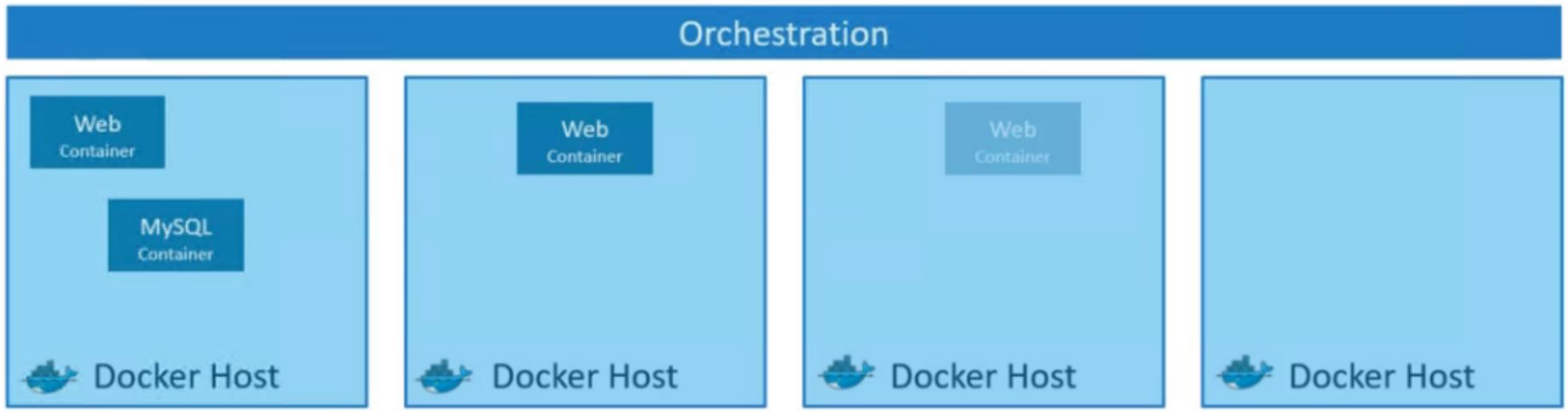
- **Kubernetes**, in Greek, means the Helmsman, or pilot of the ship, pilot of a ship of containers
- Kubernetes is a software written in **Go** for **automating deployment**, **scaling**, and **management** of containerized applications
- Focus on manage **applications**, not machines
- Open source, open API container **orchestrator**
- Supports **multiple cloud** and **bare-metal** environments
- Inspired and informed by 15 years of Google's experiences and internal systems

The Why's and What's of Kubernetes

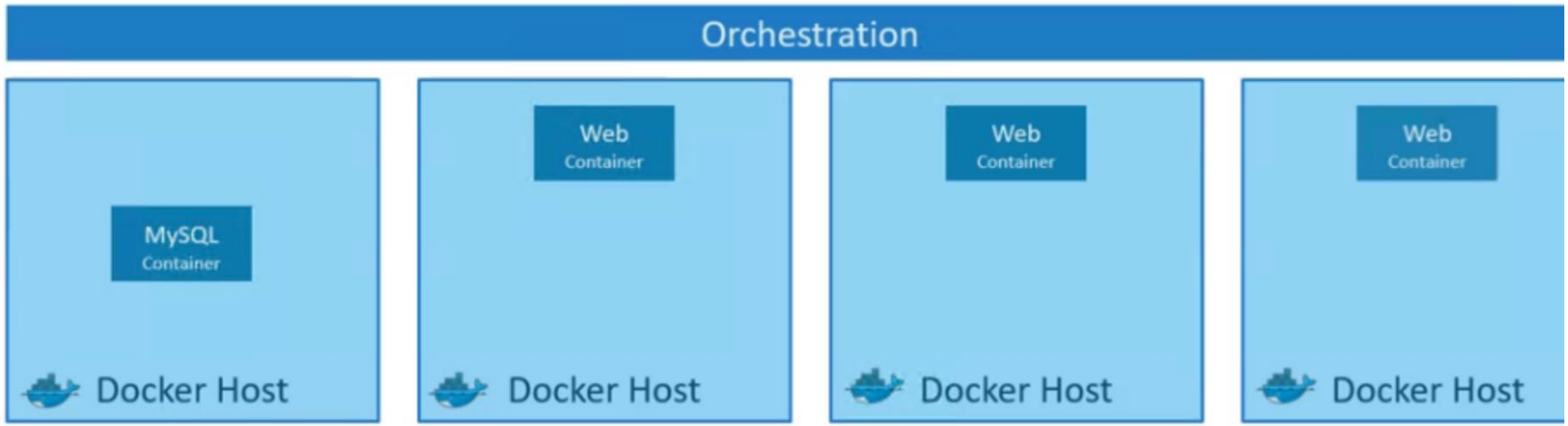
Container orchestration

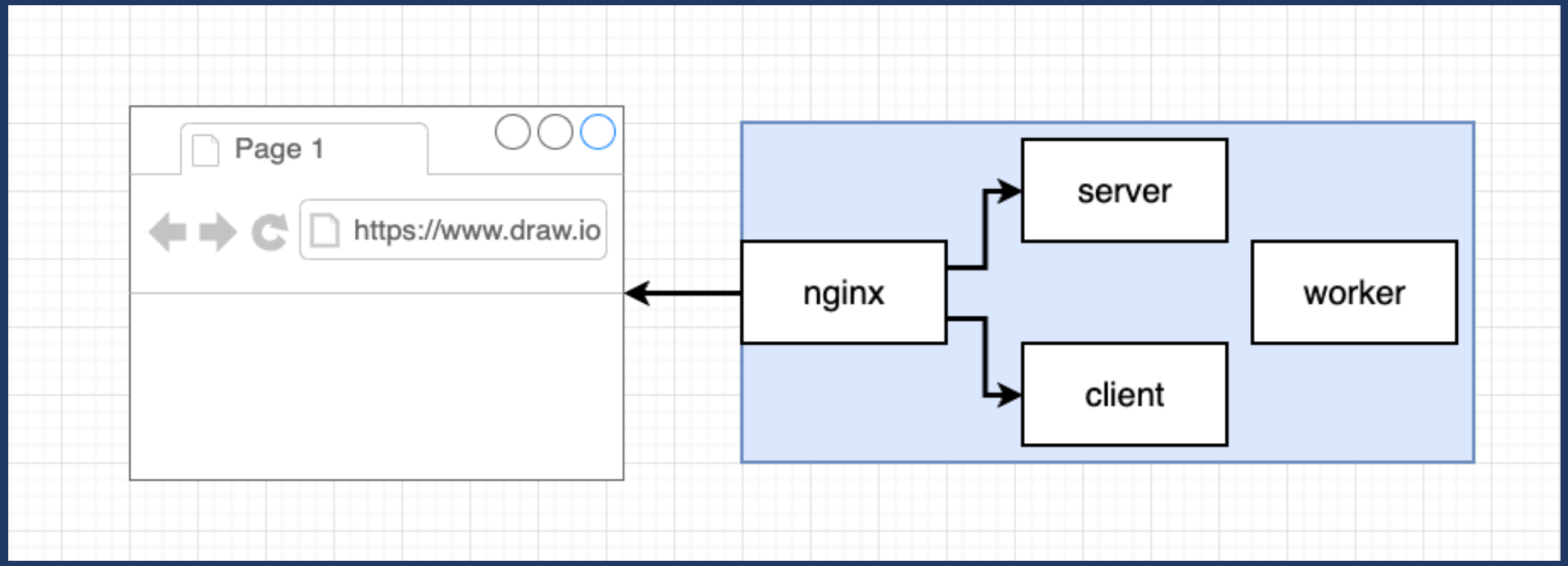


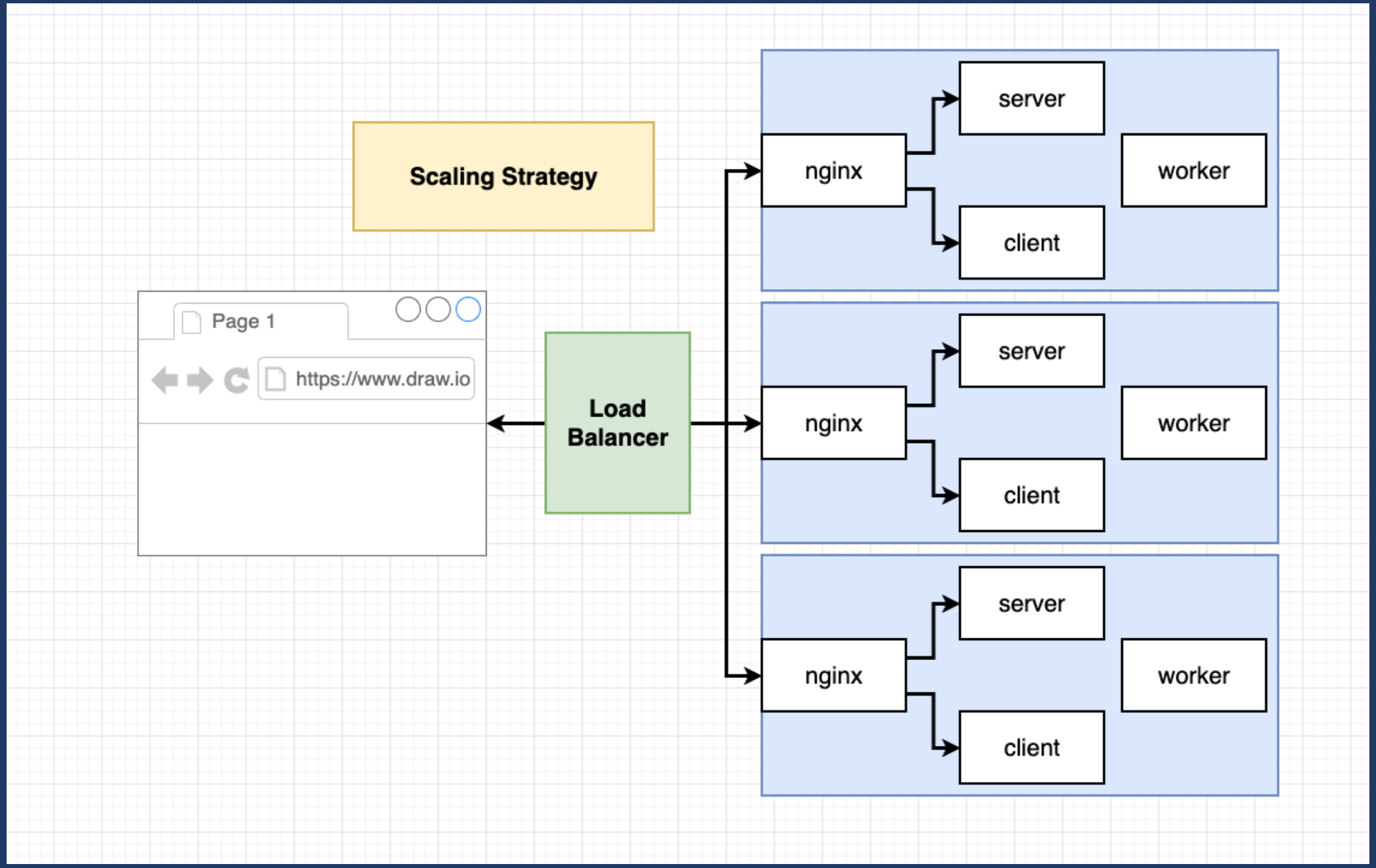
Container orchestration



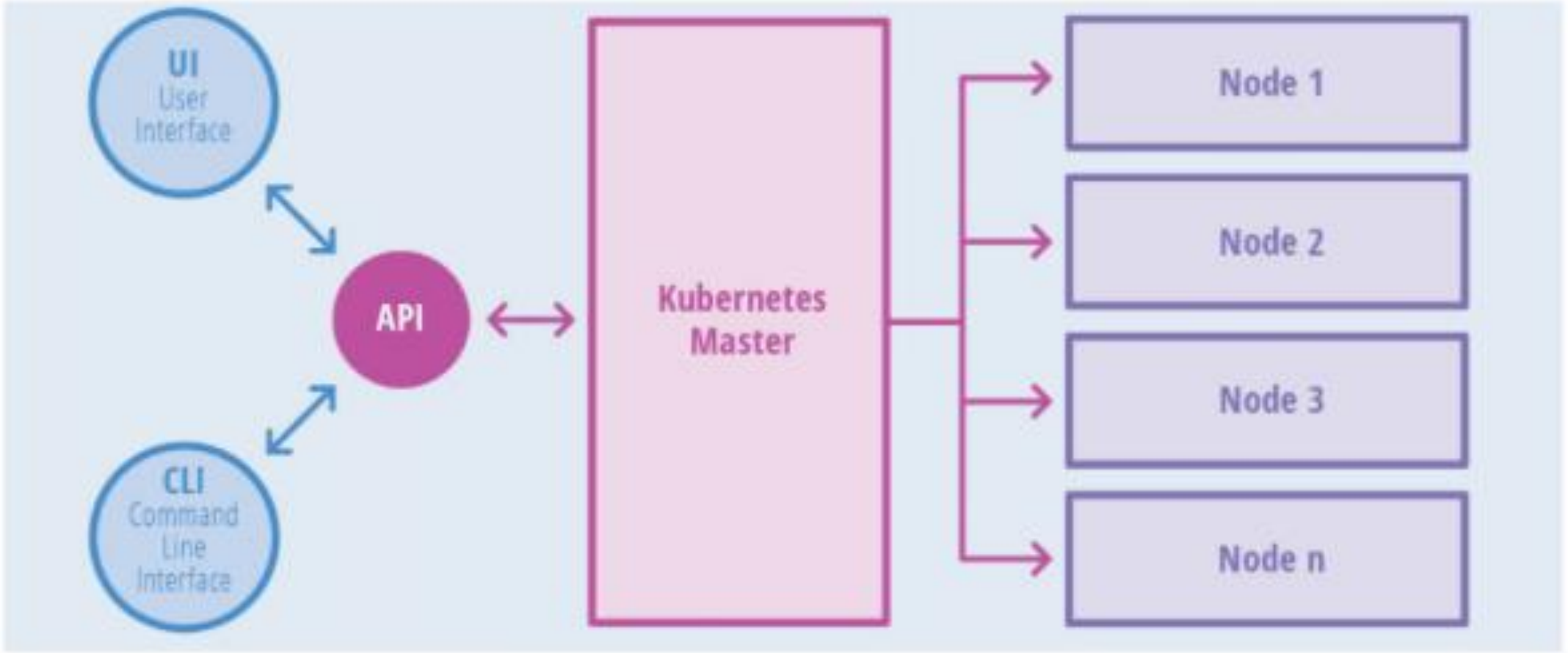
Container orchestration

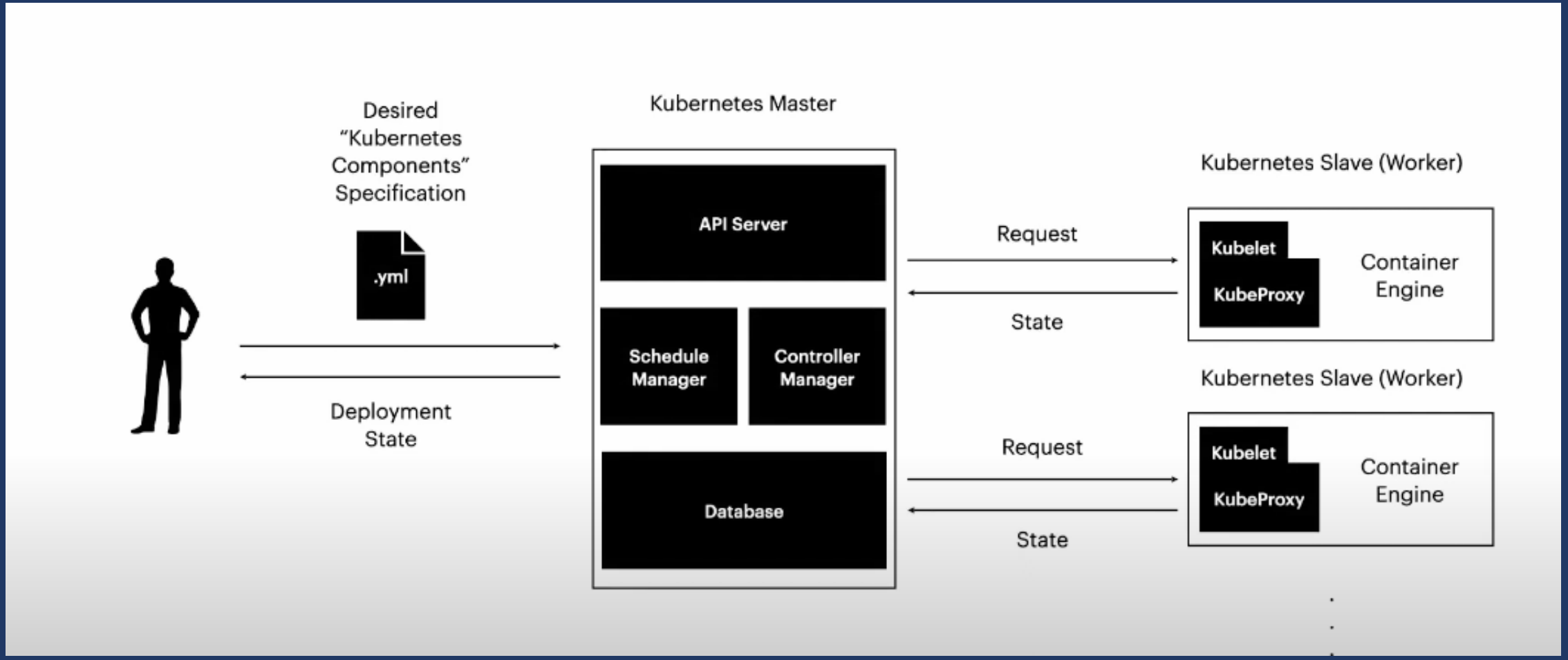


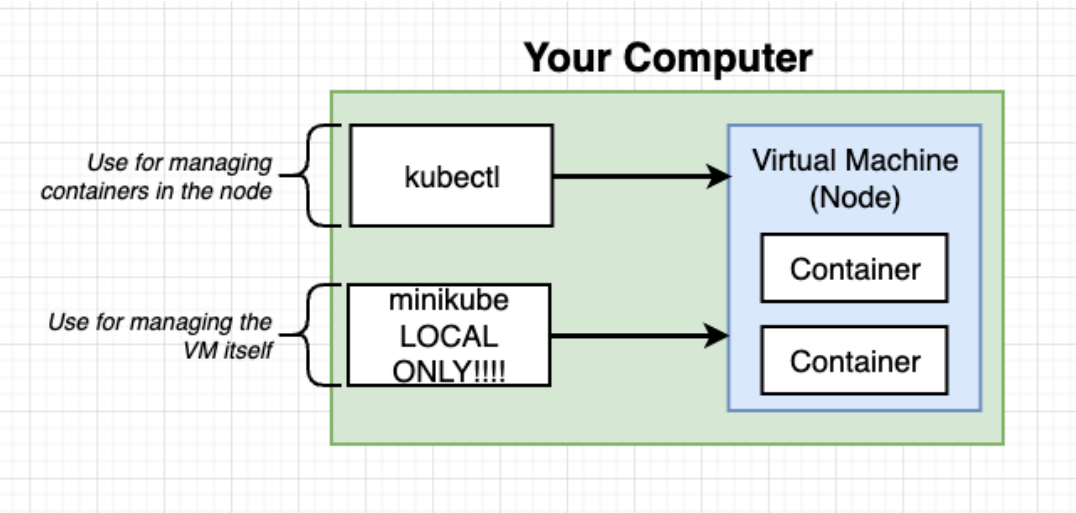

Page 1
https://www.draw.io
nginx
server
worker
client

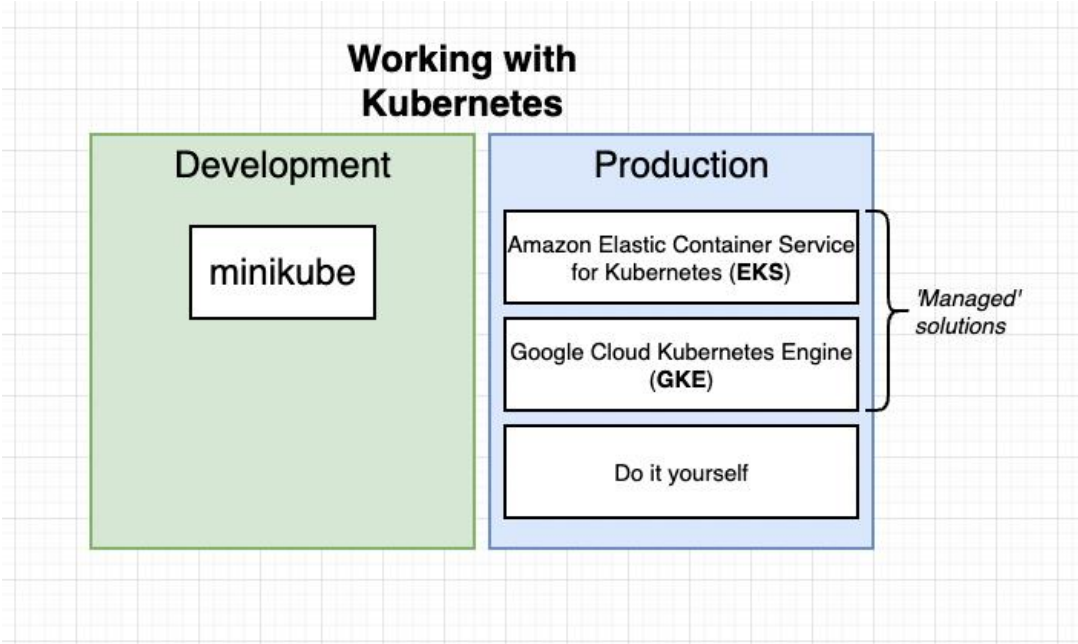

Scaling Strategy
Page 1
https://www.draw.io
Load Balancer
nginx
server
client
worker
nginx
server
client
worker
nginx
server
client
worker

Kubernetes Architecture




Desired
"Kubernetes
Components"
Specification
.yml
Deployment
State
Kubernetes Master
API Server
Schedule
Manager
Controller
Manager
Database
Request
State
Request
State
Kubernetes Slave (Worker)
Kubelet
KubeProxy
Container
Engine
Kubernetes Slave (Worker)
Kubelet
KubeProxy
Container
Engine


Your Computer
Use for managing containers in the node
kubectl
Virtual Machine (Node)
Container
Container
Use for managing the VM itself
minikube LOCAL ONLY!!!!

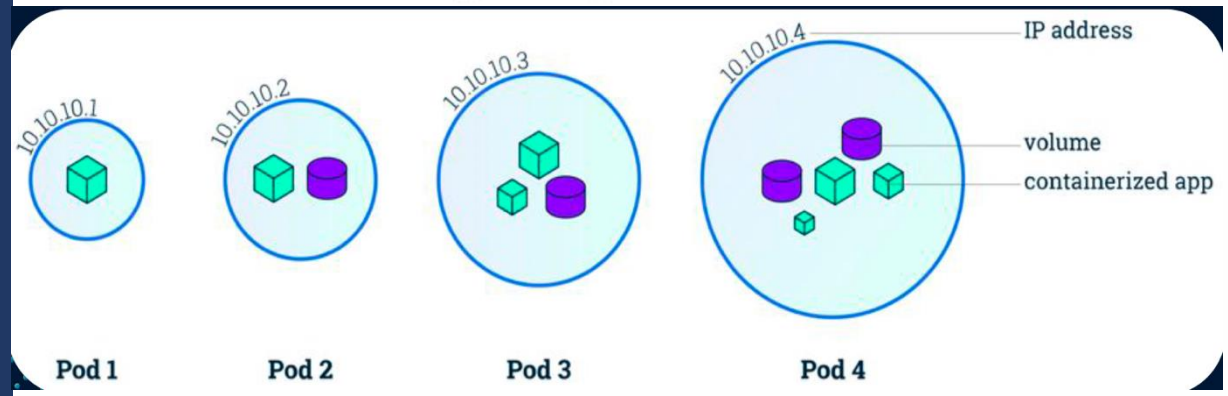

Working with Kubernetes
Development
minikube
Production
Amazon Elastic Container Service for Kubernetes (EKS)
Google Cloud Kubernetes Engine (GKE)
Do it yourself
'Managed' solutions

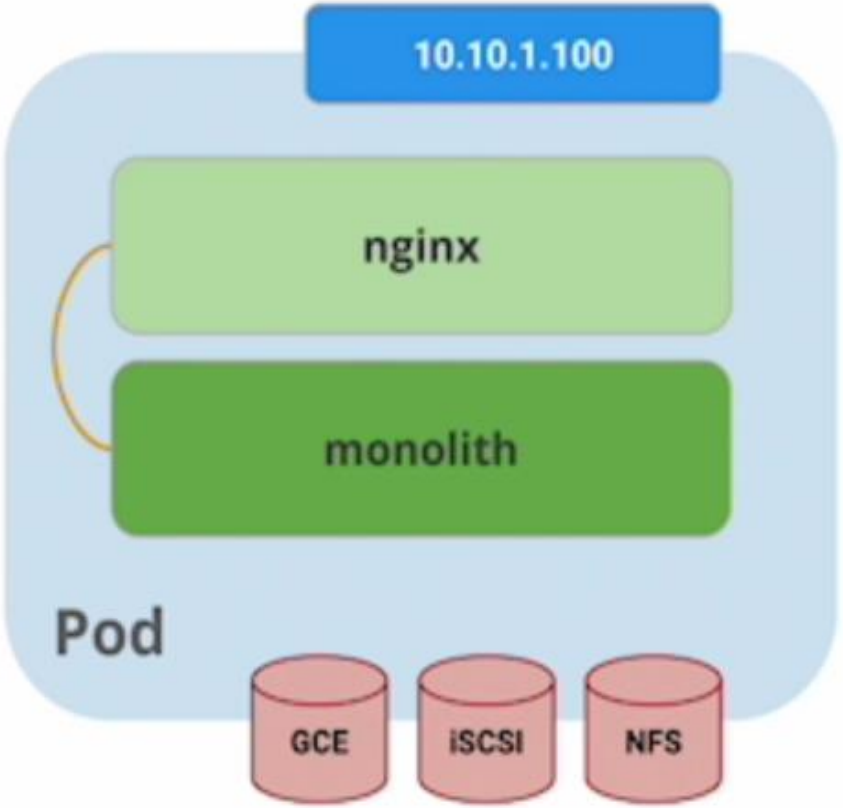
Concept of pods replica deployments service and namespace Rolling update

Pods

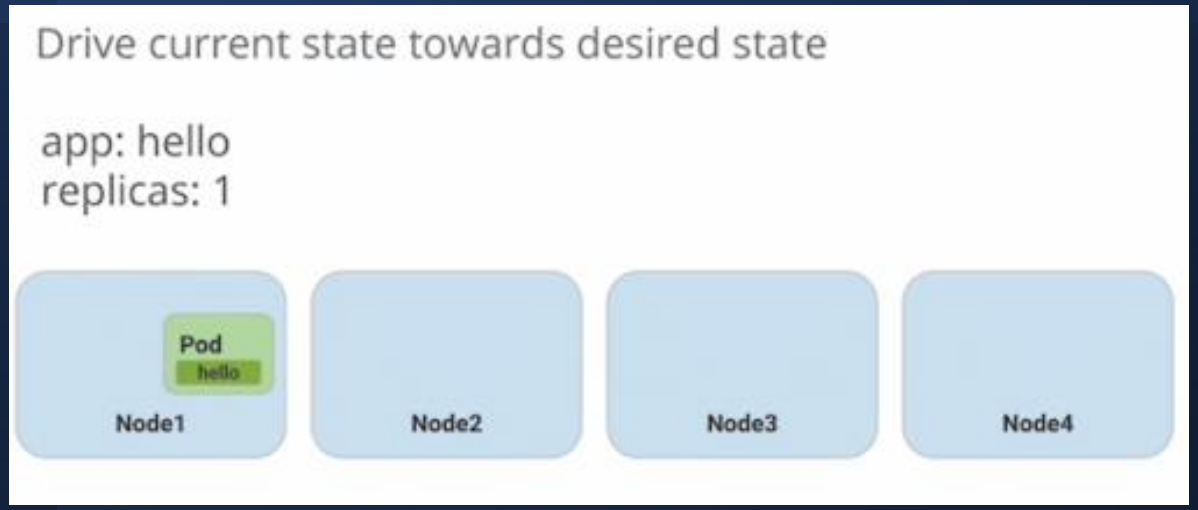
Logical Application

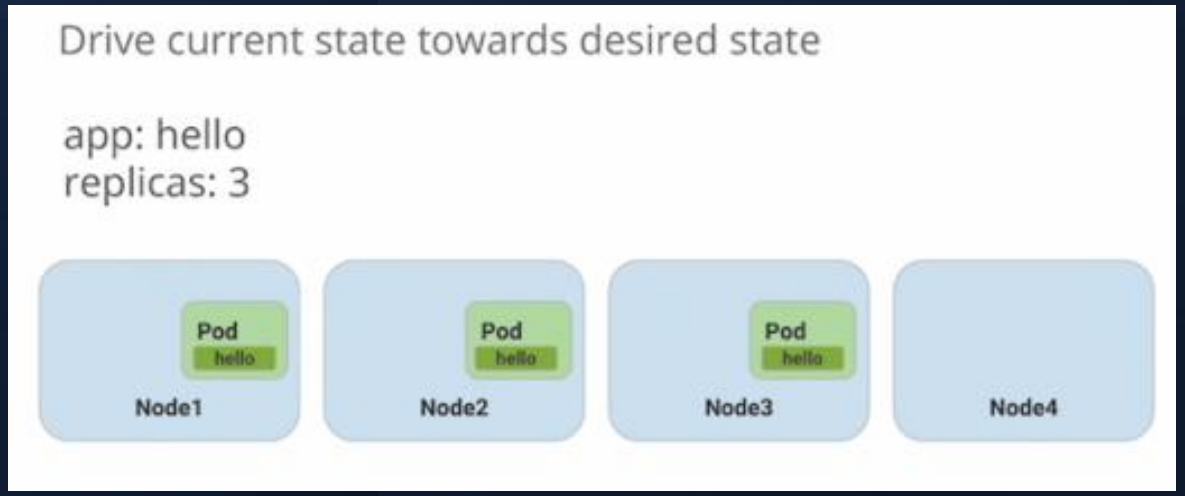
- One or more containers and volumes
- Shared namespaces
- One IP per pod





Deployment and replica





Deployments

Drive current state towards desired state

app: hello
replicas: 3

Deployments

Drive current state towards desired state

app: hello
replicas: 3

Dead 1 Node

Deployments

Drive current state towards desired state

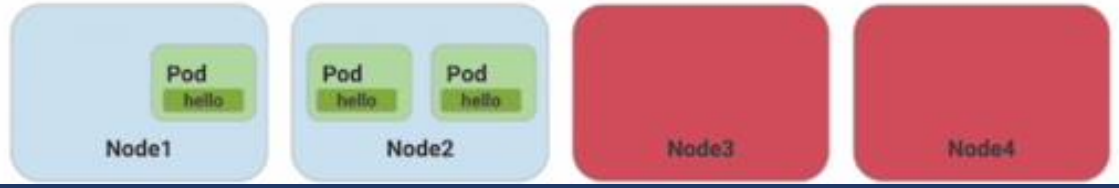
app: hello
replicas: 3

Sol :Dead 1 Node

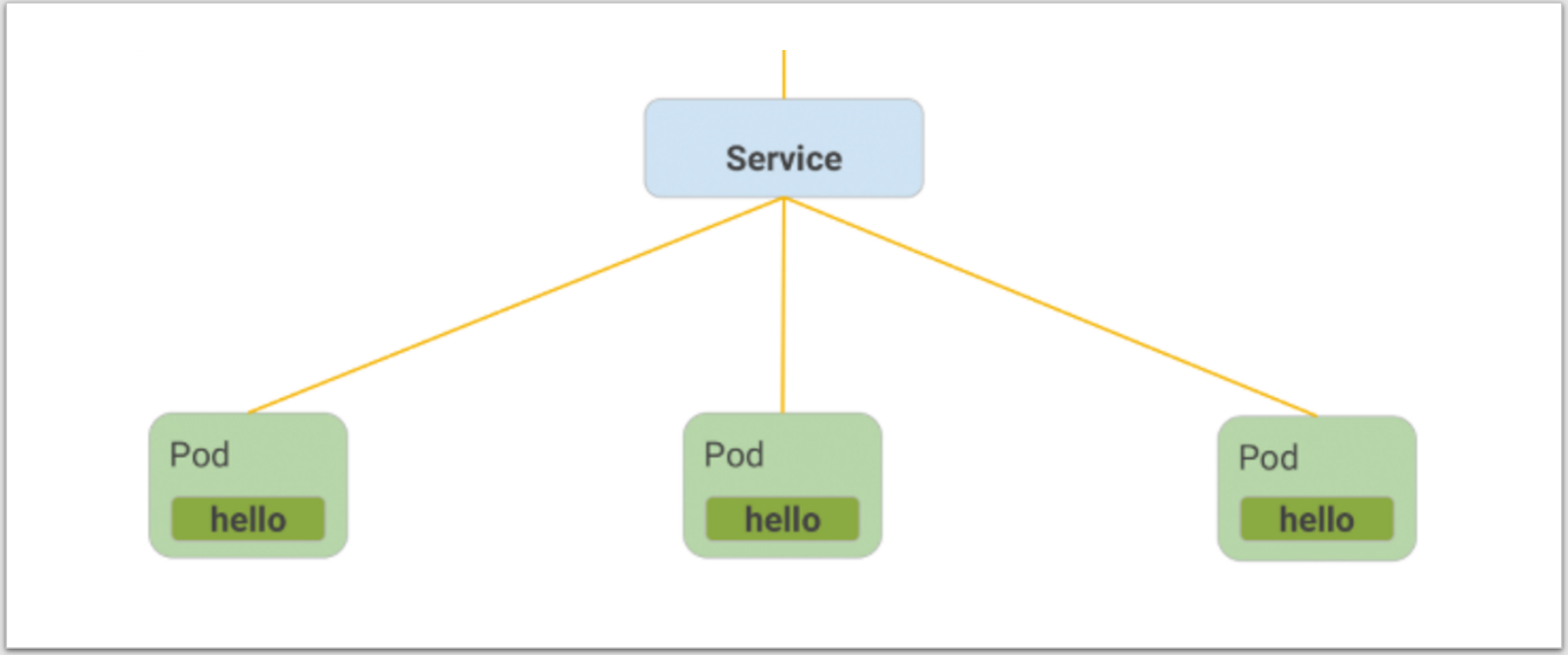
Deployments

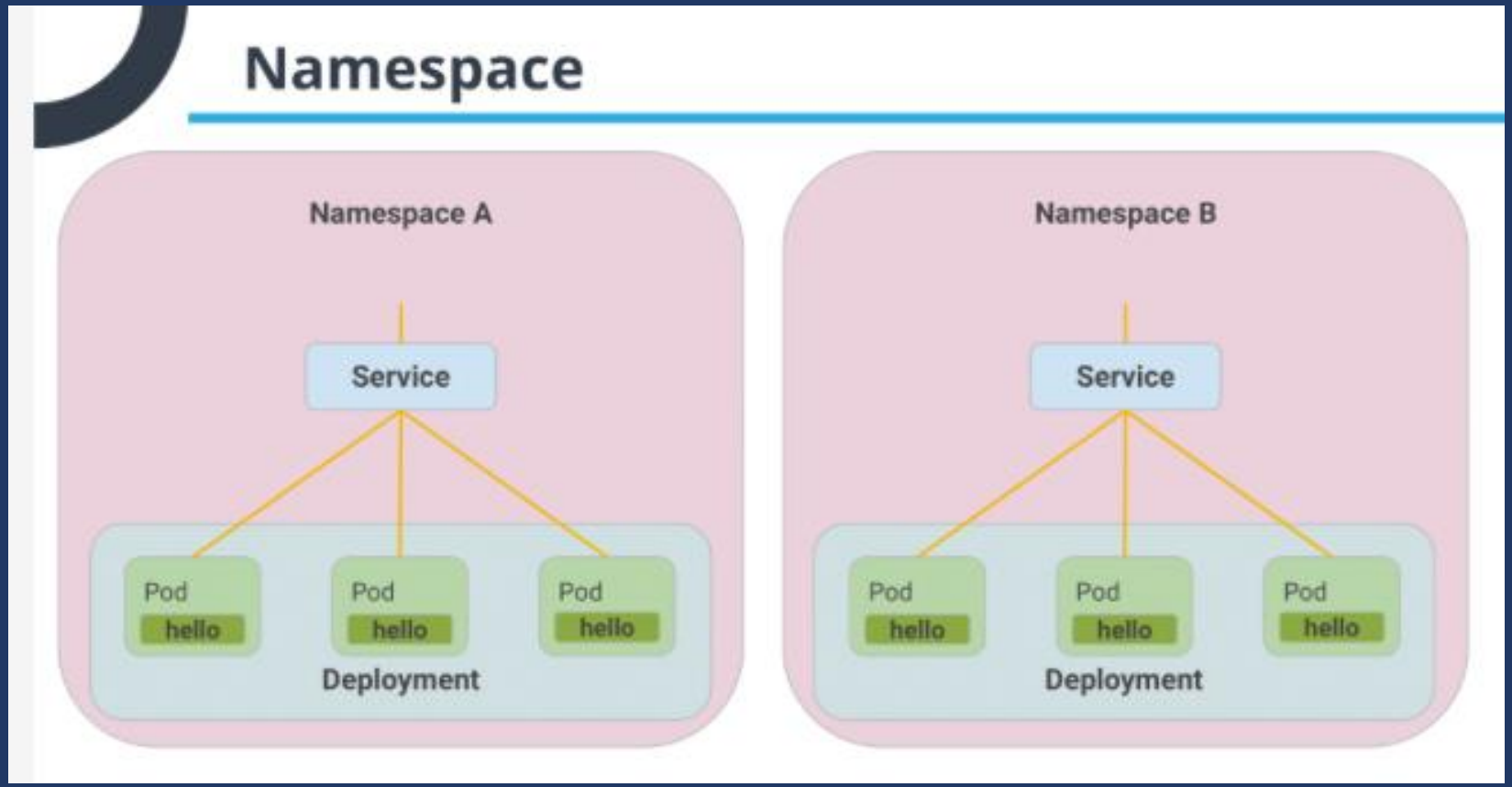
Drive current state towards desired state

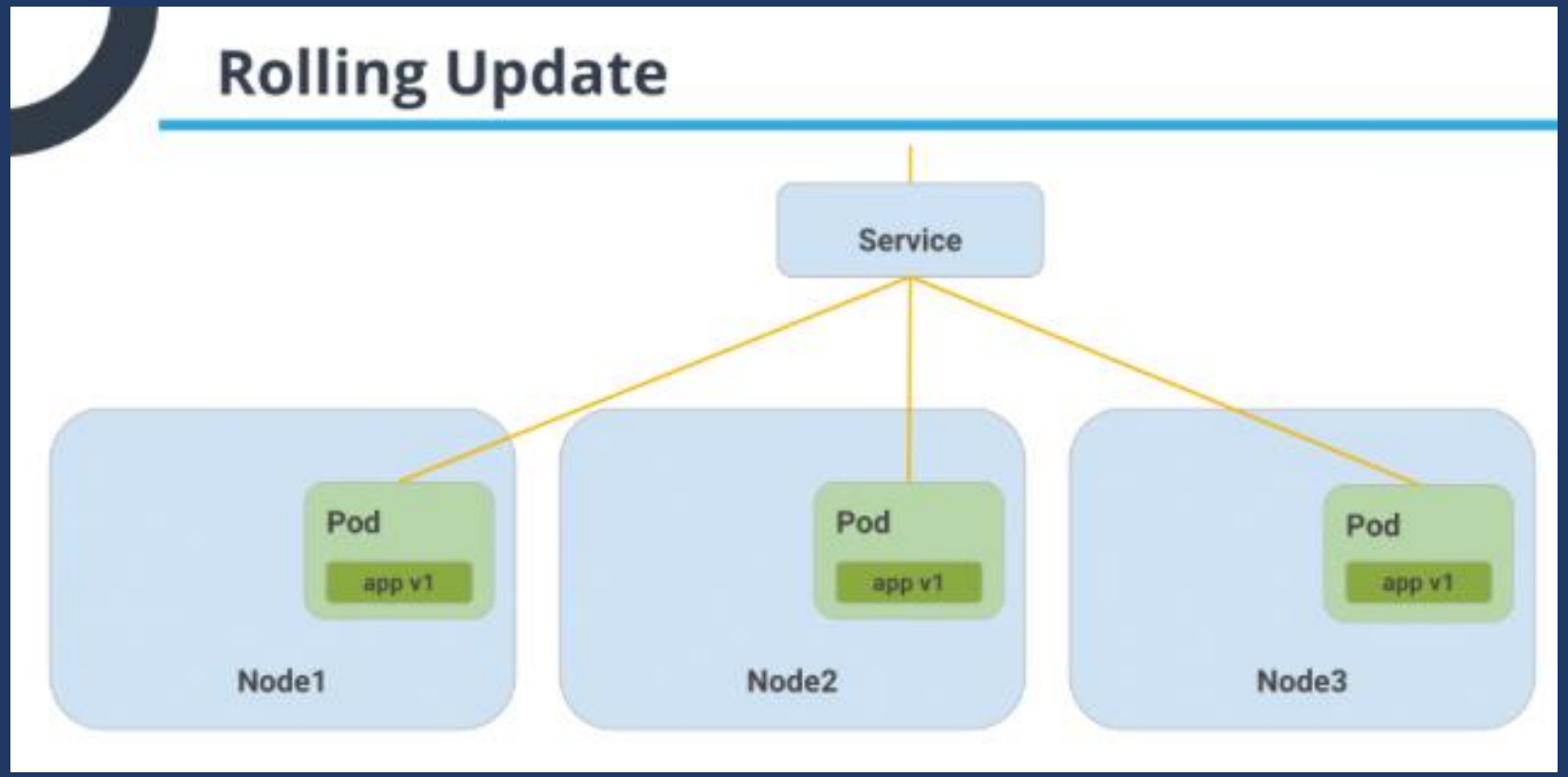
app: hello
replicas: 3

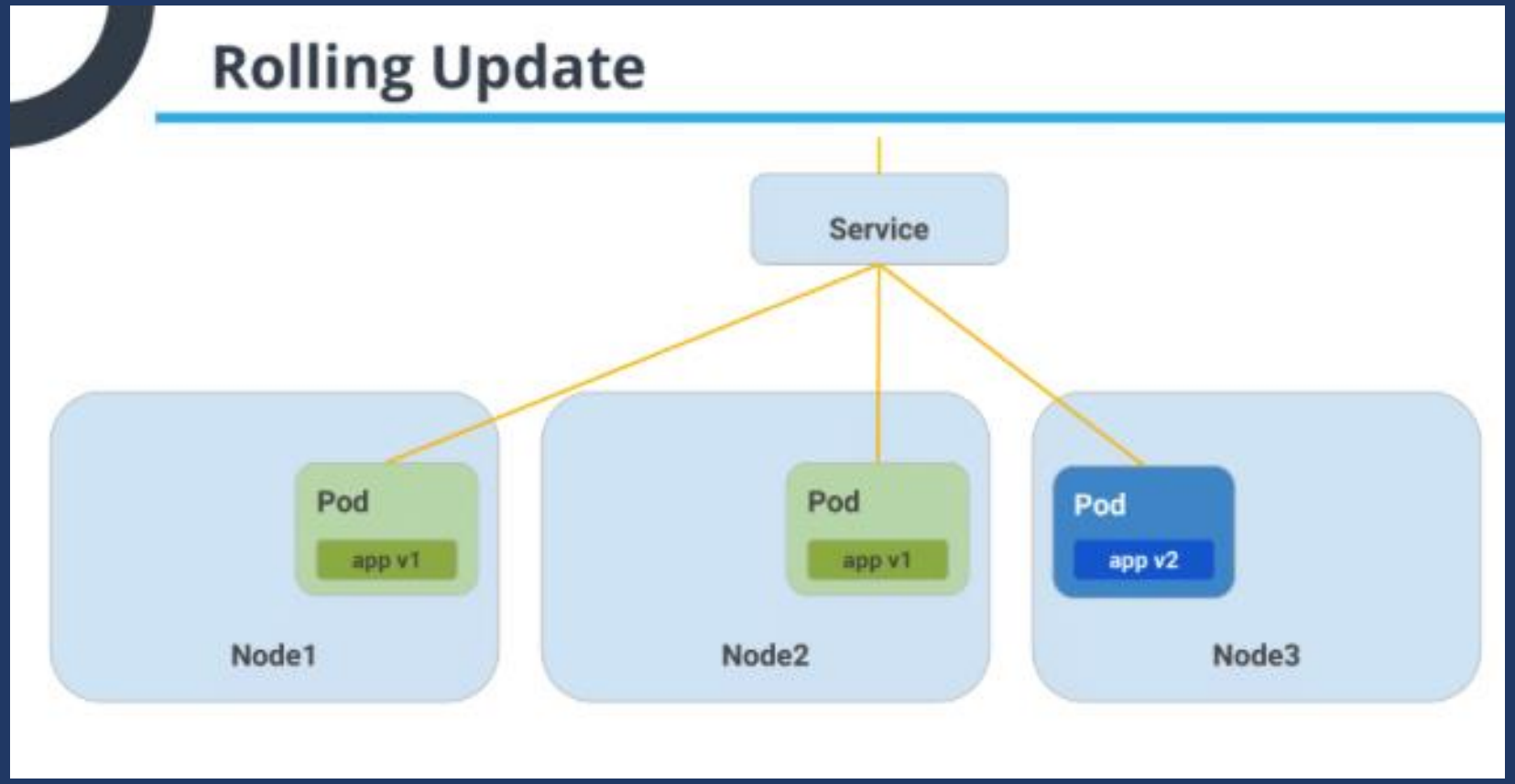


Service

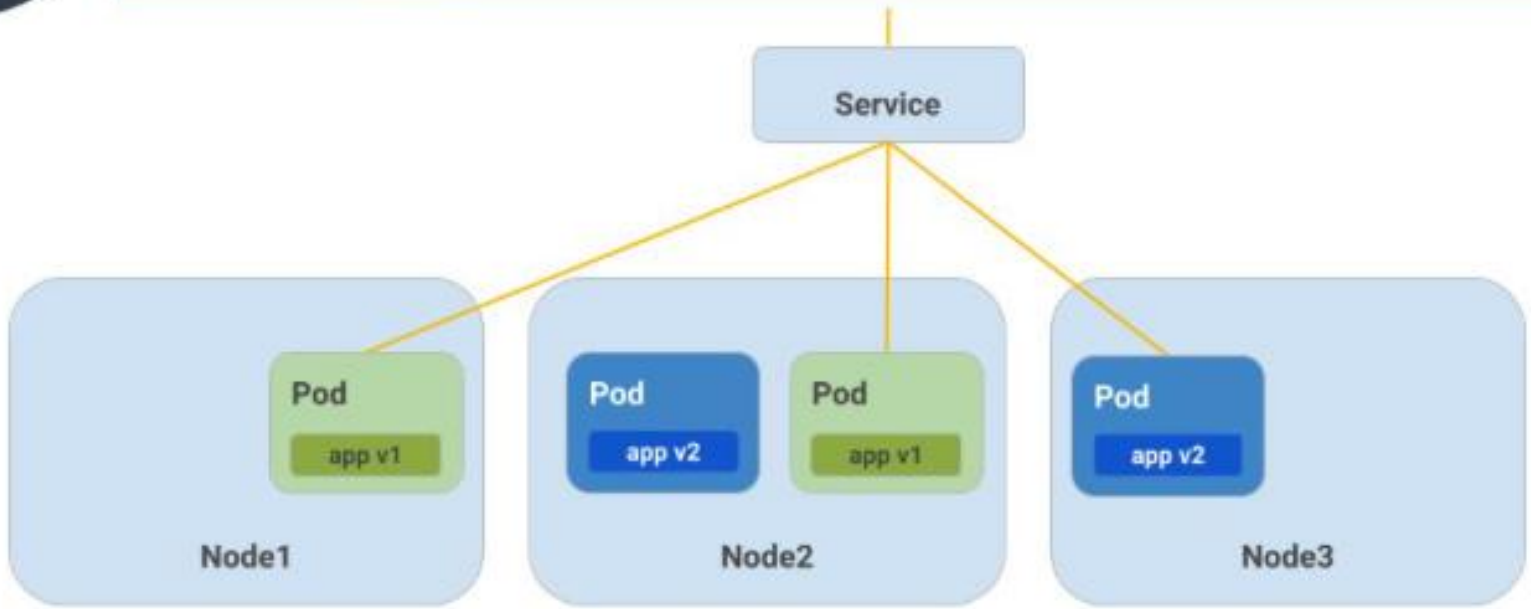


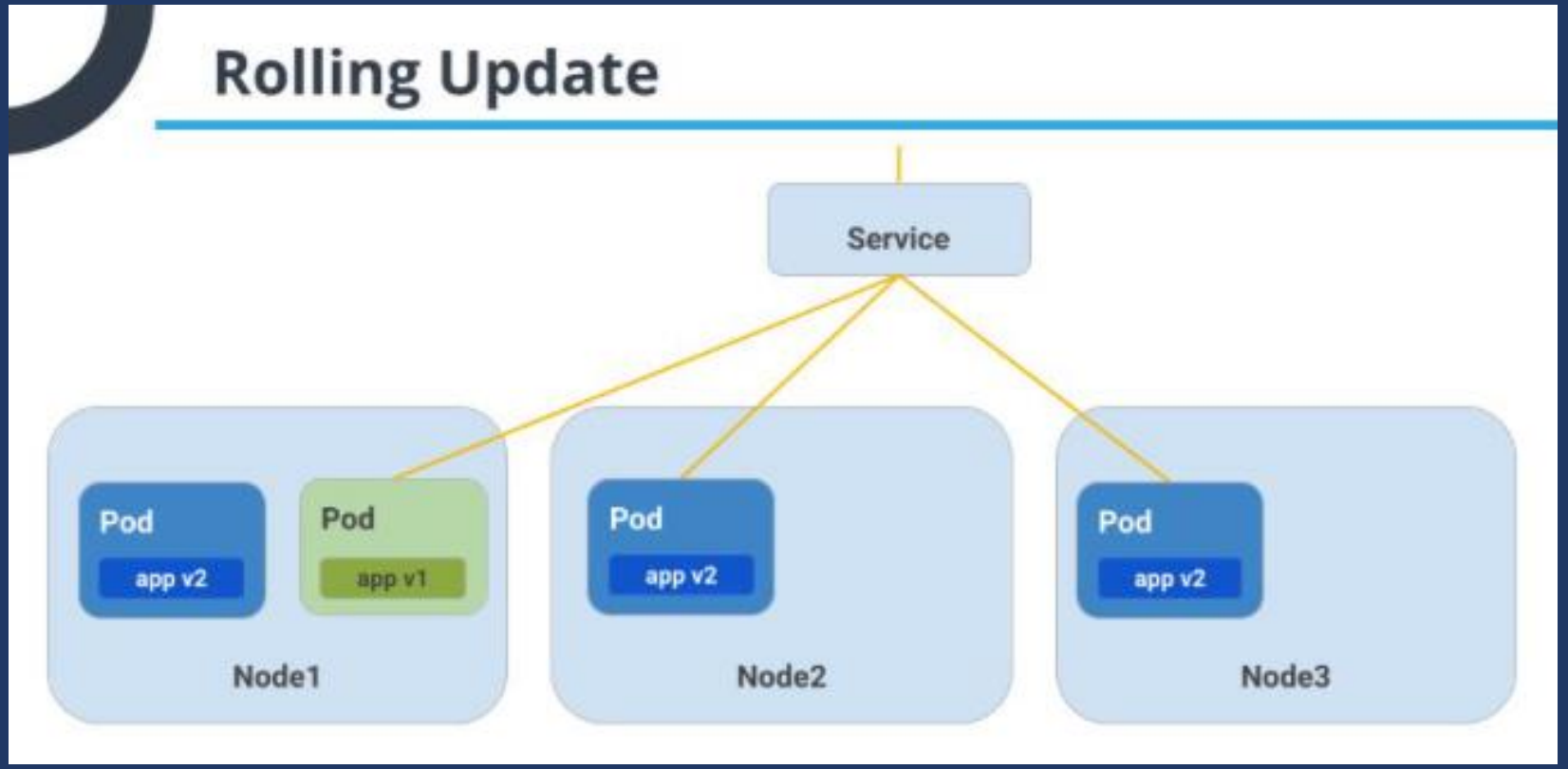

Namespace
Namespace A
Service
Pod
hello
Pod
hello
Pod
hello
Deployment
Namespace B
Service
Pod
hello
Pod
hello
Pod
hello
Deployment

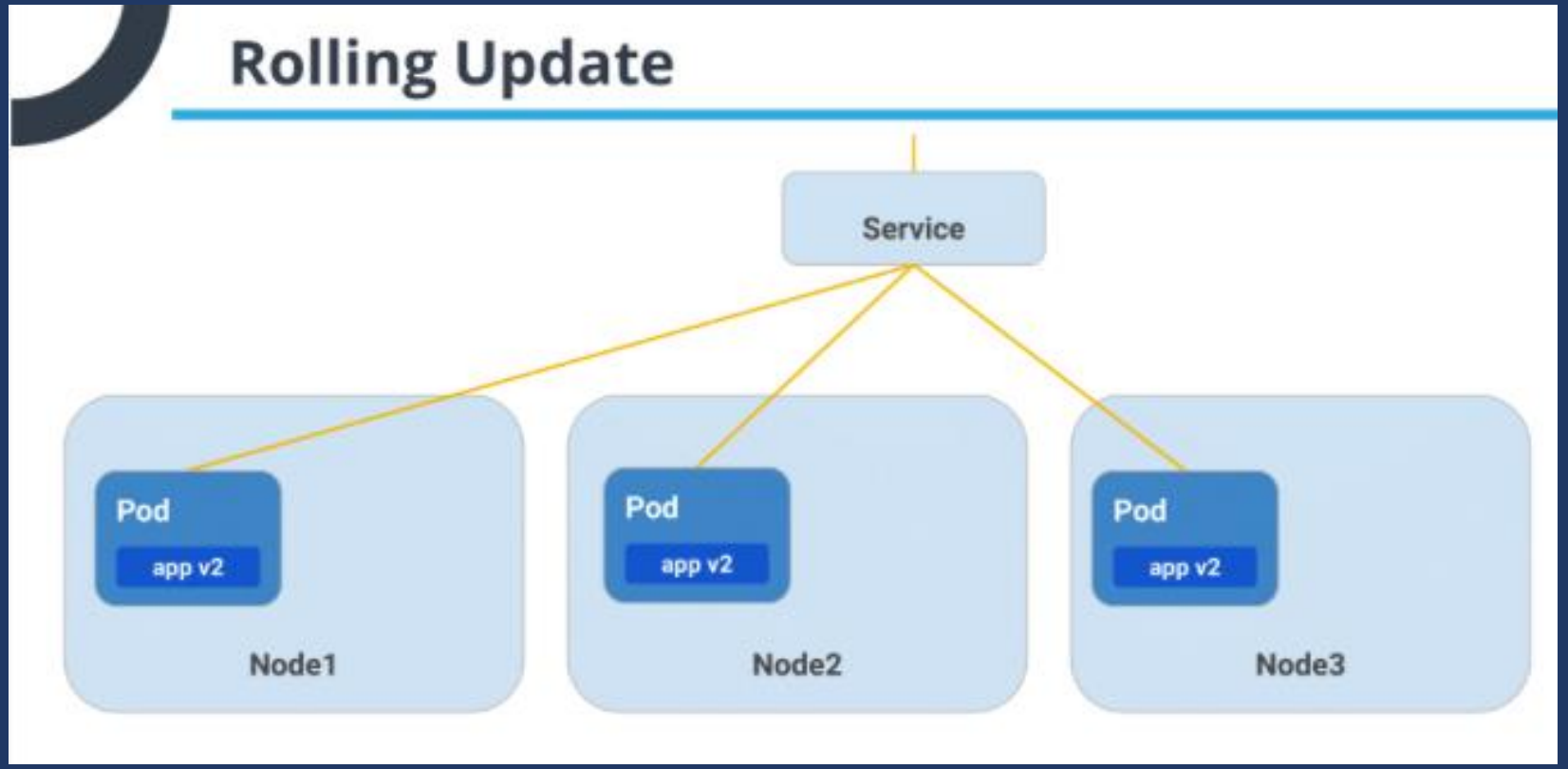

Rolling Update
Service
Pod
app v1
Pod
app v1
Pod
app v1
Node1
Node2
Node3


Rolling Update
Service
Pod
app v1
Pod
app v1
Pod
app v2
Node1
Node2
Node3

Rolling Update




Rolling Update
Service
Pod
app v2
Pod
app v1
Pod
app v2
Pod
app v2
Node1
Node2
Node3


Rolling Update
Service
Pod
app v2
Node1
Pod
app v2
Node2
Pod
app v2
Node3

Workflow for docker and kubernetes deployment:

Docker and Kubernetes Workflow

- 3 Files
  - Docker File
  - Deployment Yaml File
  - Service Yaml File

- Pod

https://kubernetes.io/docs/concepts/workloads/pods/

- Replicas set

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

- Deployment

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

Basic Command

Kubernetes Basic Command (get) :

kubectl get {object} –n {namespace}

- kubectl get pod –n kube-system
- kubectl get replicaset –n kube-system
- kubectl get deployment –n kube-system

Kubernetes Basic Command (describe):

kubectl describe {object} {pod name} –n {namespace}

```
kubectl describe pod {pod name} –n kube-system
kubectl describe replicaset {replicaset name} –n kube-system
kubectl describe deployment {deplotment name} –n kube-system
```

Kubernetes Basic Command (log) :

kubectl logs {pod name} –n {namespace}

kubectl logs pod {pod name} –n kube-system

Kubernetes Basic Command (log) :

kubectl exec -it {pod name} –n {namespace} -- sh/bash

```
kubectl exec -it {pod name} –n {namespace} -- sh/bash
```

LAB basic-command

Open a terminal window and run the following command to check the version of kubectl:

```
kubectl version
```

displays a table of all the nodes in the cluster

```
kubectl get node
```

display all the namespaces in the cluster, including their names, statuses

```
kubectl get namespace
```

Get information about resources in a specific namespace:

To get all pods across all namespaces in Kubernetes

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

Get a list of all pods in the kube-system namespace:

```
kubectlget pods -n kube-system
```

Get a list of all services in the kube-system namespace:

```
kubectl get services -n kube-system
```

Get a list of all deployments in the kube-system namespace:

```
kubectl get deployments -n kube-system
```

Pod

YAML

- YAML is markup language. It is **human-readable** and easy to understanding syntax.
- YAML has format for specific **configuration** information such as POD & Deployment definitions.
- Many software to relate like **Kubernetes**, **SDN** and **OpenStack** etc.
- Similar **JSON language**
- Advantage to use YAML format
  - **Infrastructure as a code** (IAAC) - keep it on standard template.
  - **Stability** - keep it on version control, keep history change.
  - **Flexibility** - higher dynamic than use command line.

```
---
# Identity customer records
group_one:
  name: John Franky
  job: Developer
  skills:
    - Python
    - Ruby
    - customer
```

YAML in Kubernetes

pod-definition.yml

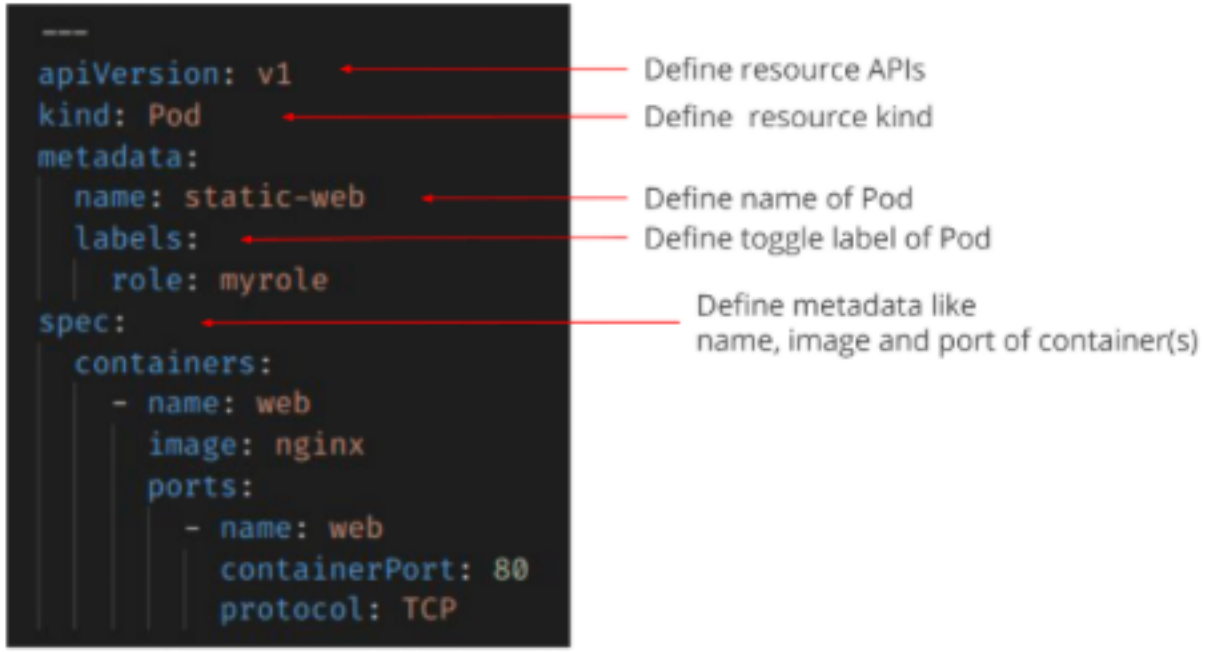
```
apiVersion:
kind:
metadata:



spec:
```

Kubernetes Manifest File

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: busybox
  namespace: default
spec:
  containers:
  - name: busybox
    image: busybox
    command:
      - sleep
      - "3600"
```

This is what called
Infrastructure as Code (IaC)

Manifest Structure



YAML in Kubernetes

pod-definition.yml

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



spec:
```

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

YAML in Kubernetes

pod-definition.yml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
      app: myapp
      type: front-end
spec:
```

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

YAML in Kubernetes

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
```

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

YAML in Kubernetes

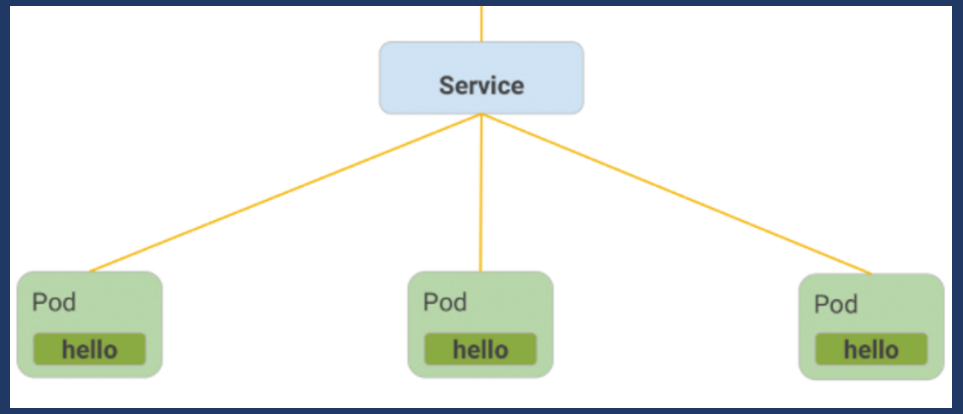
pod-definition.yml

```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 pod-definition.yml
```

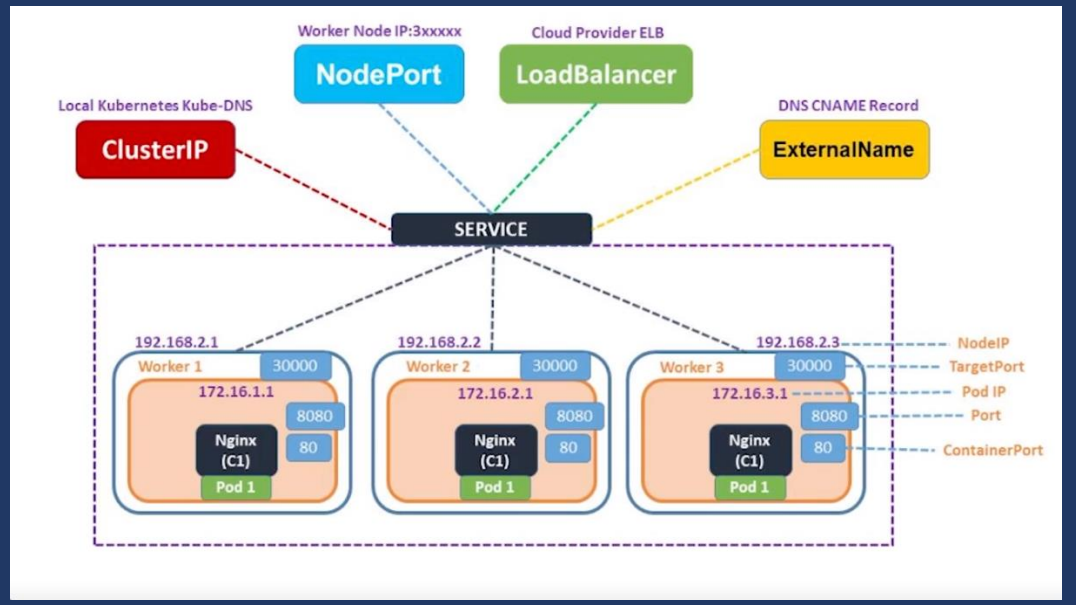
Kind	Version
POD	v1
Service	v1
ReplicaSet	apps/v1
Deployment	apps/v1

Service



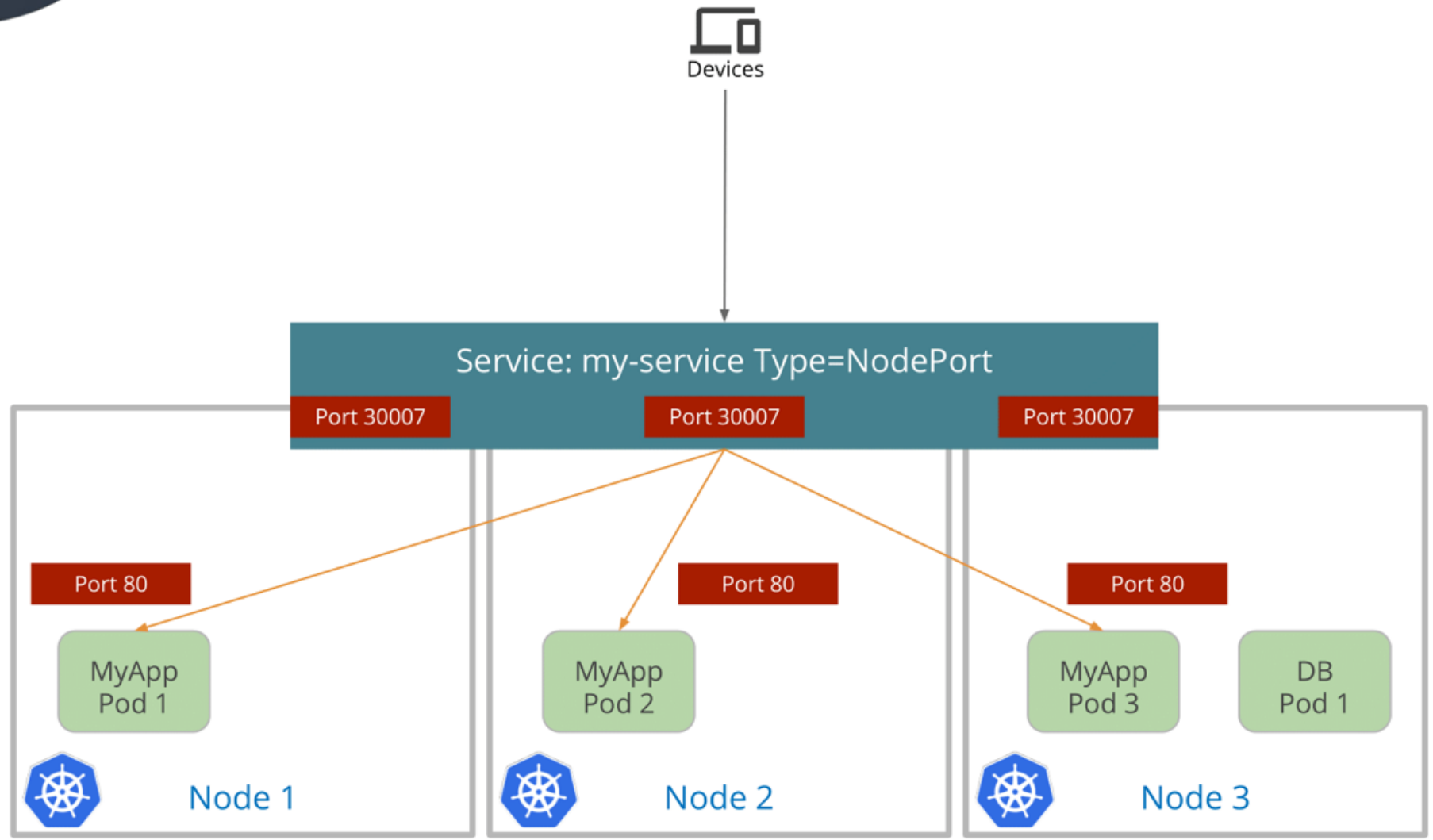
In Kubernetes, there are generally three types of services

1. NodePort: This service type allows the pods to be accessed from outside the cluster via a static port on each worker node. This is often used for development and testing purposes, but not recommended for production use as it can pose security risks.
2. ClusterIP: This is the default service type in Kubernetes. It provides a stable IP address and DNS name to access a set of pods within the same Kubernetes cluster. This type of service is used for internal communication between components within the cluster.
3. LoadBalancer: This service type is used to expose pods to the internet or other external networks by creating a load balancer in a public cloud provider, such as AWS or GCP. It provides a single entry point for traffic to reach the pods and distributes the traffic across multiple nodes.

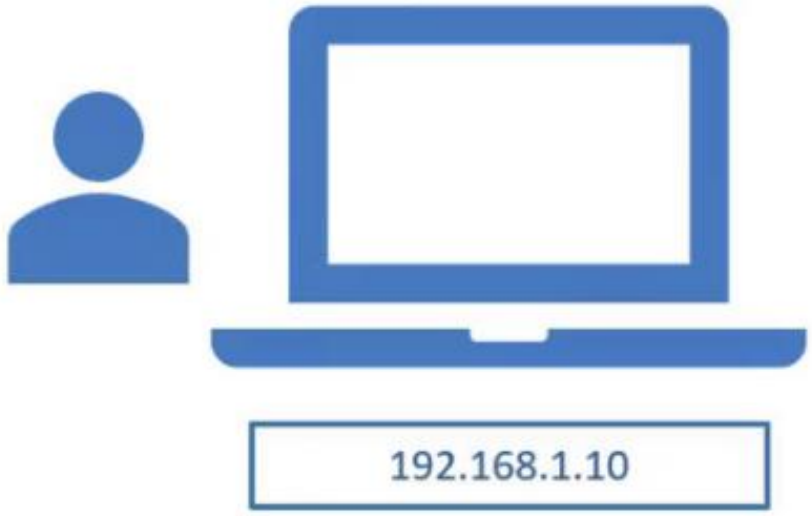


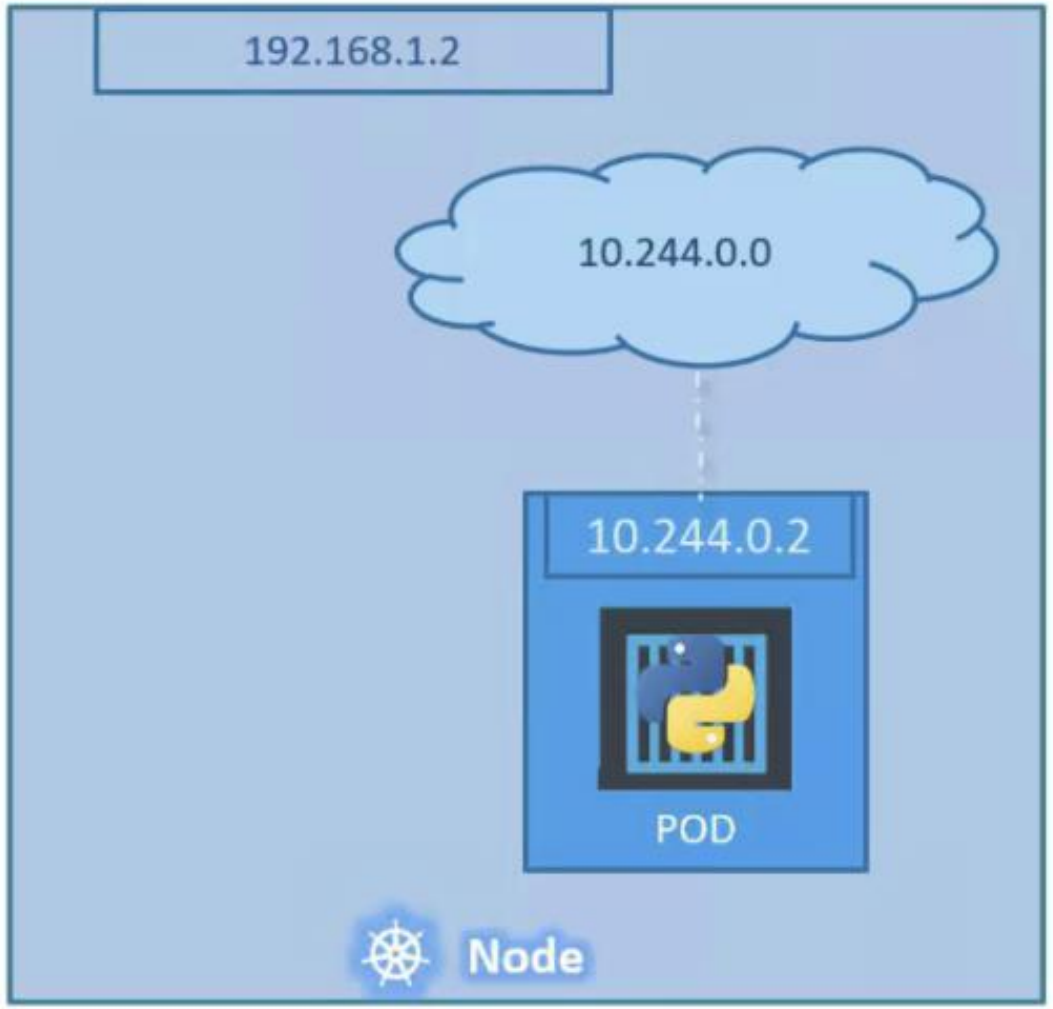
Node Pod

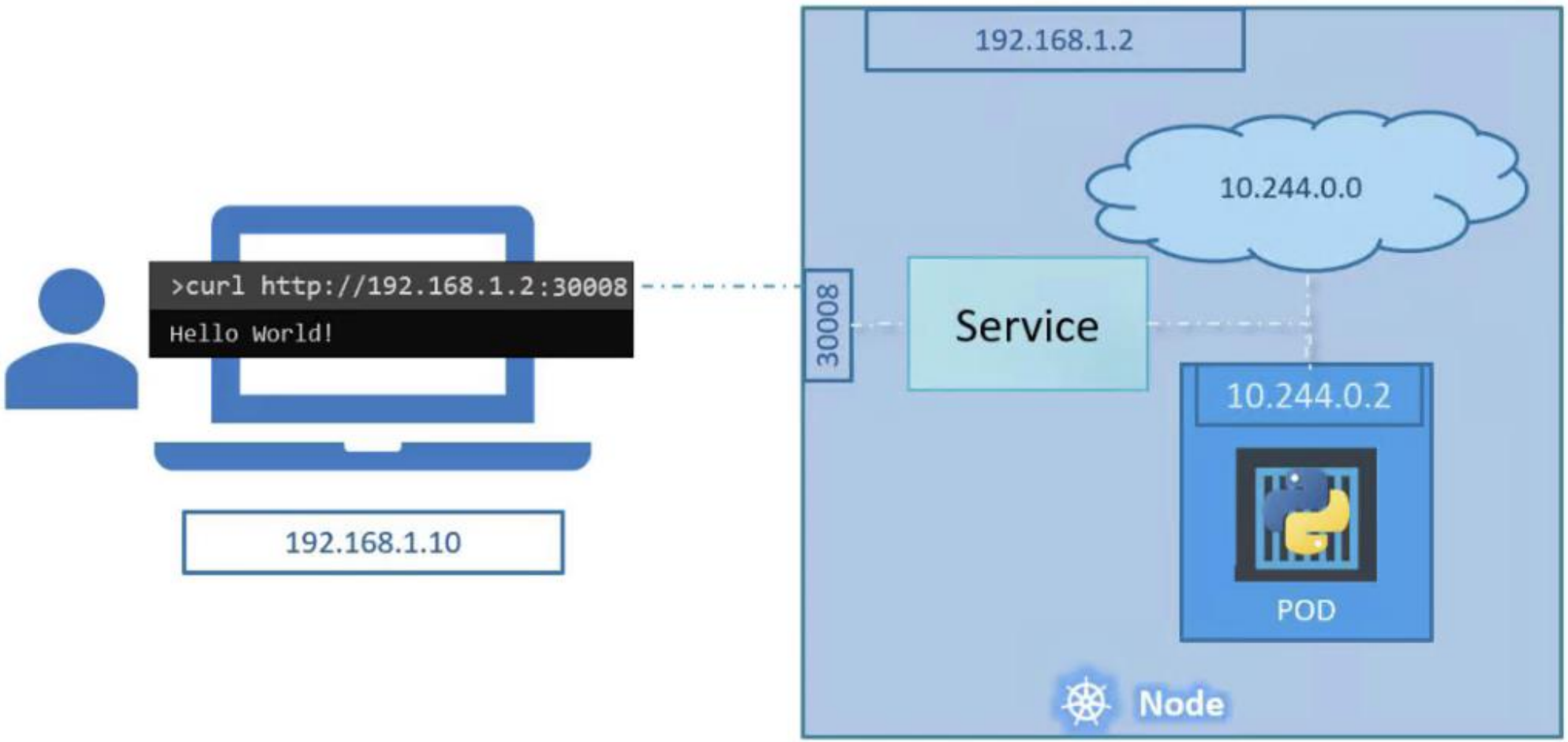
Service Type: Node Port



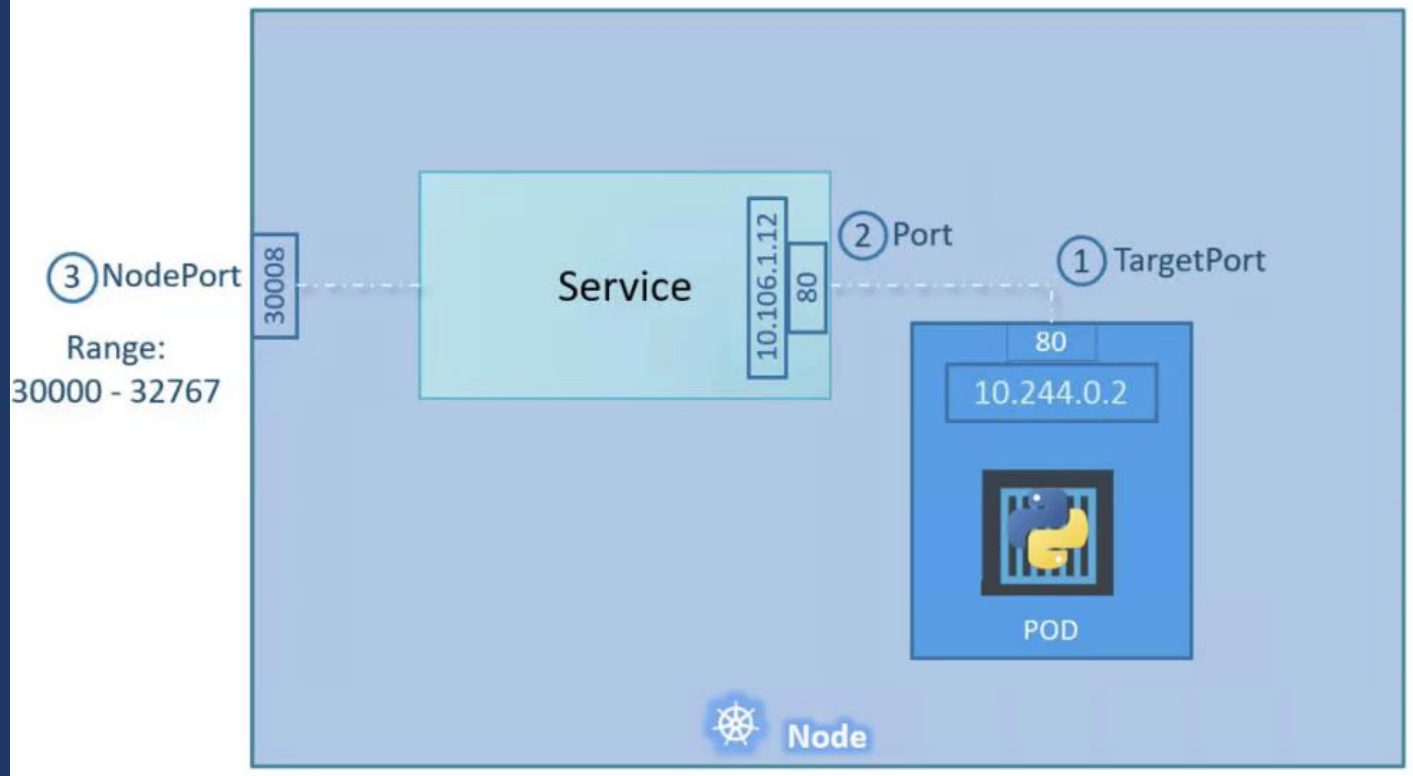

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  type: NodePort
  selector:
    app: MyApp
  ports:
      # By default and for convenience, the `targetPort` is set to the same value as the `port` field.
    - port: 80
      targetPort: 80
      # Optional field
      # By default and for convenience, the Kubernetes control plane will allocate a port from a range (default: 30000-32767)
      nodePort: 30007
```


192.168.1.10


192.168.1.2
10.244.0.0
10.244.0.2
POD
Node


>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

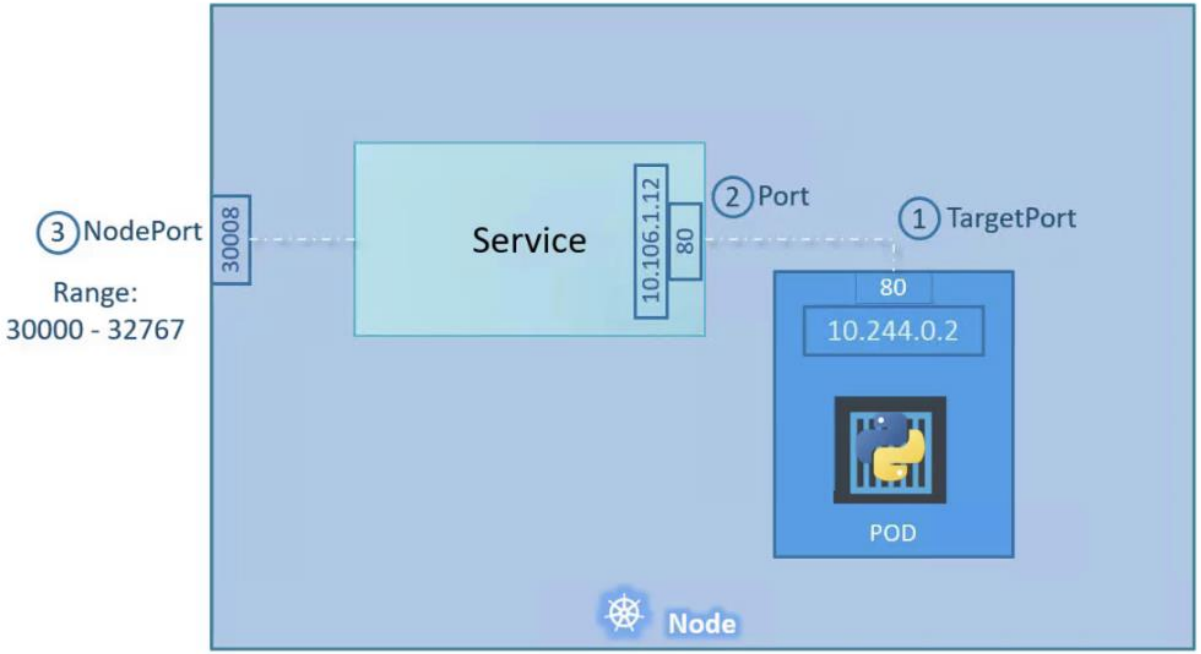
Service - NodePort



service-definition.yml

```
apiVersion:
kind:
metadata:


spec:
```

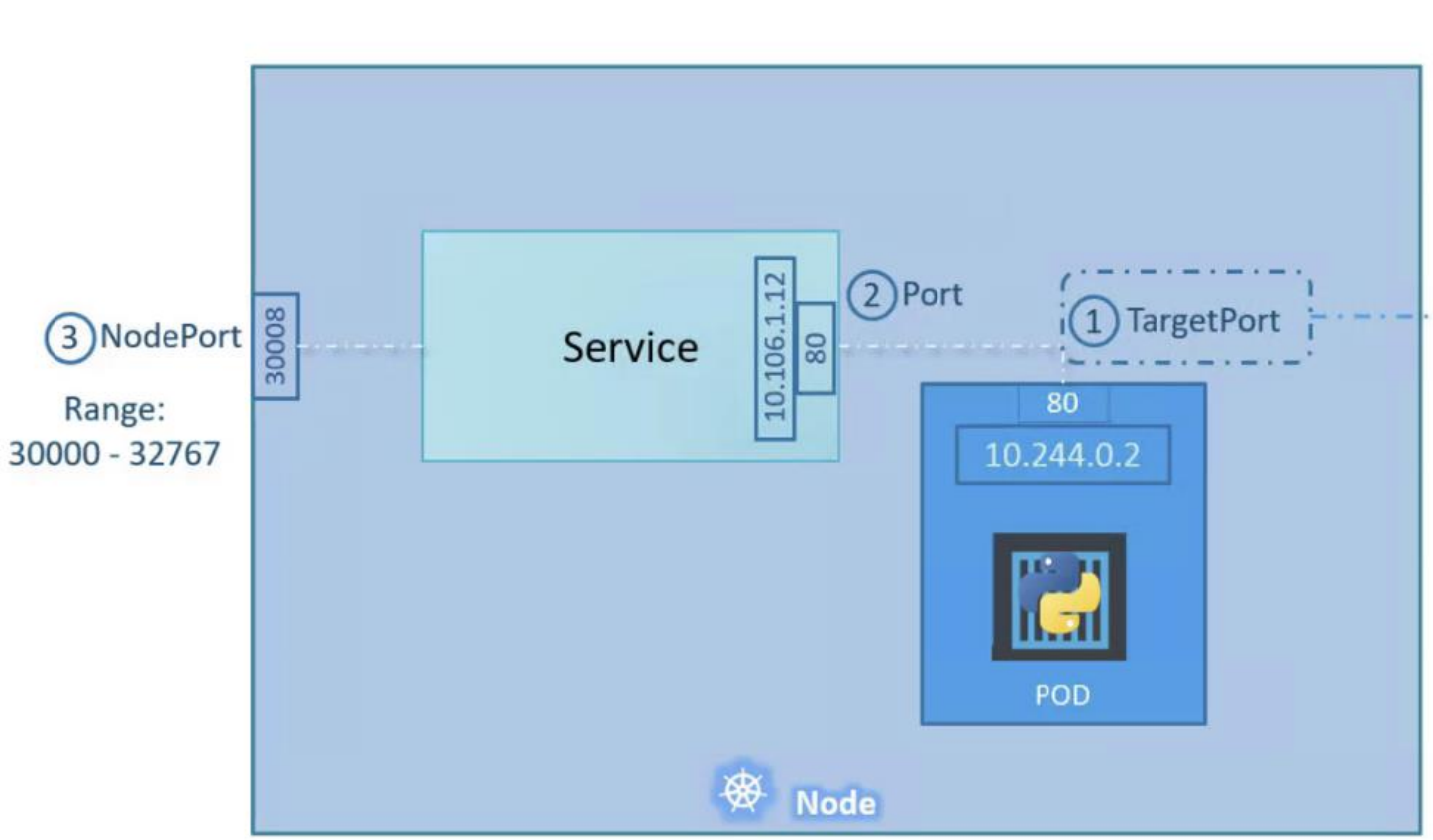
Service - NodePort



service-definition.yml

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

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

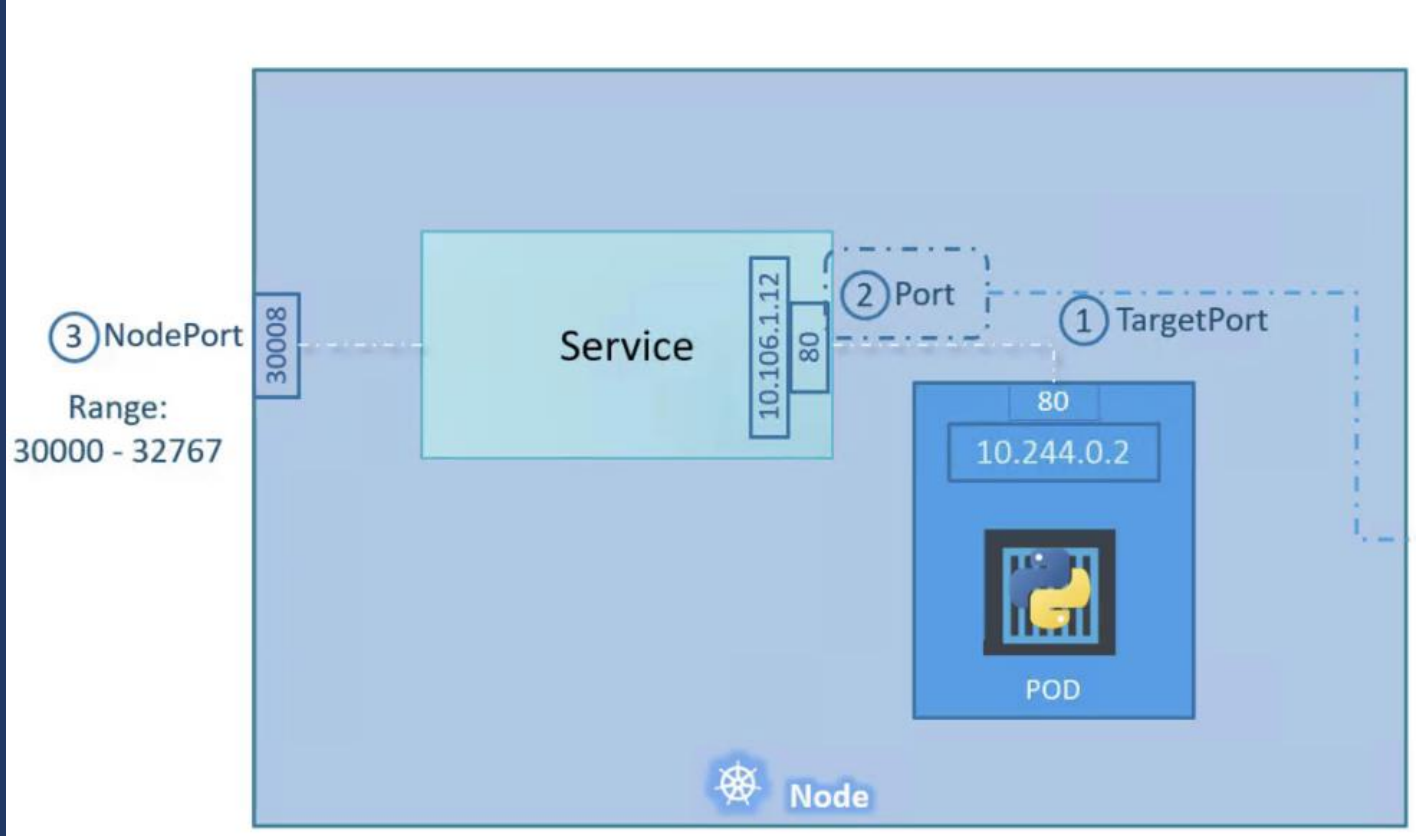
Service - NodePort



service-definition.yml

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

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

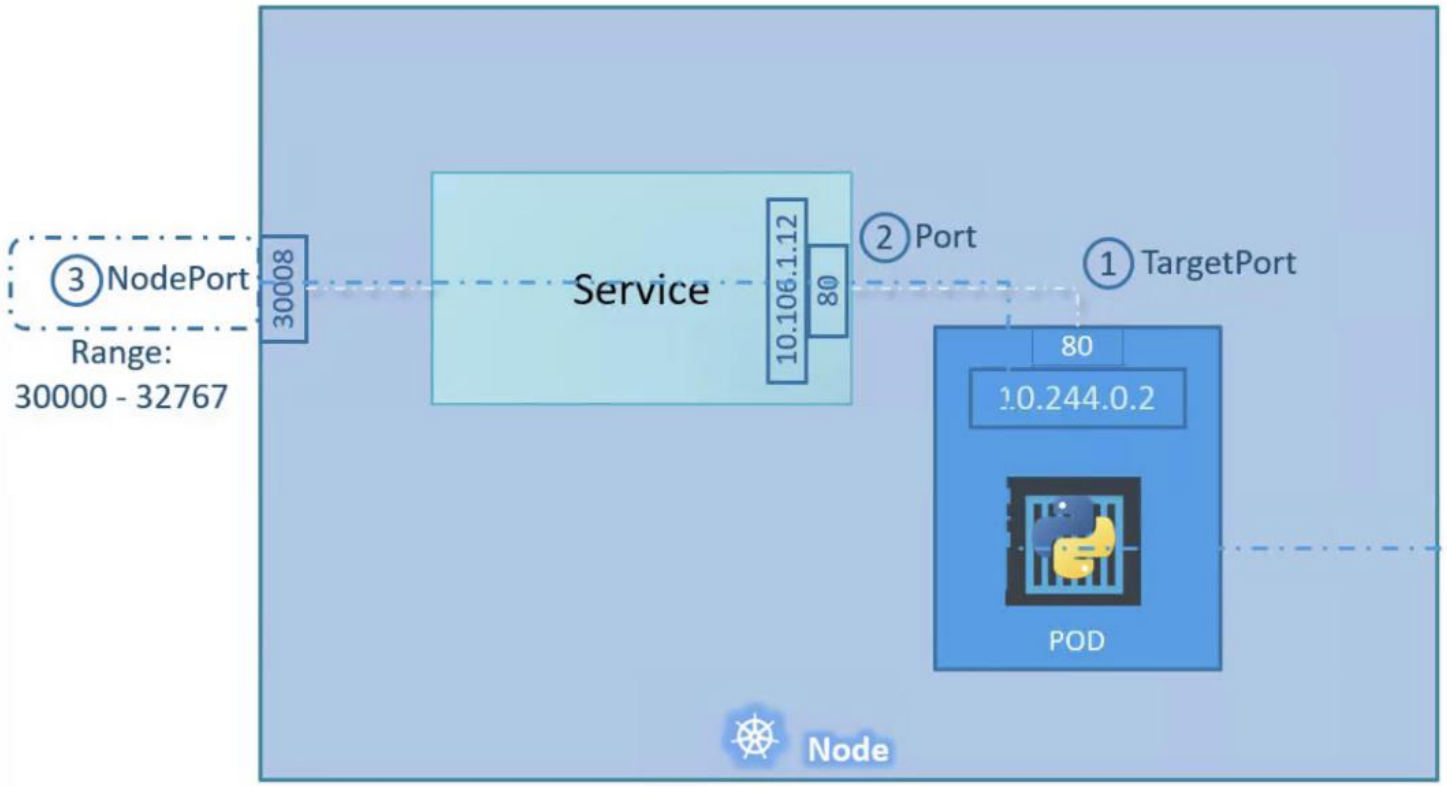
Service - NodePort



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
```

Service - NodePort

service-definition.yml

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

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

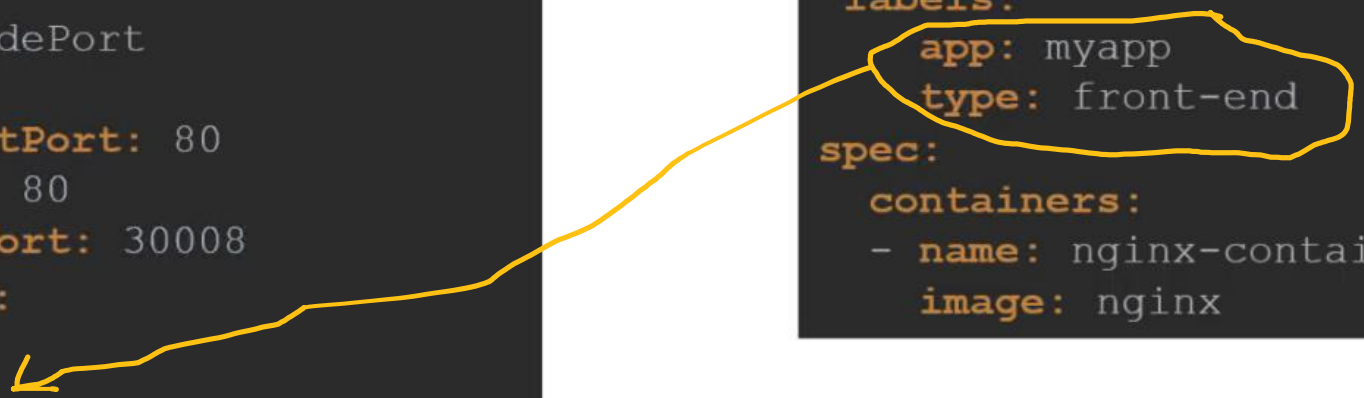
Service - NodePort

service-definition.yml

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

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

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
```



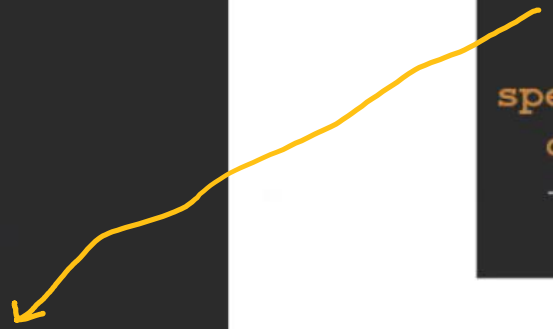
Service - NodePort

service-definition.yml

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

spec:
    type: NodePort
    ports:
     - targetPort: 80
       port: 80
       nodePort: 30008
    selector:
       app: myapp
       type: front-end
```

pod-definition.yml

```yaml
apiVersion: v1
kind: Pod

metadata:
 name: myapp-pod
 labels:


spec:
  containers:
  - name: nginx-container
    image: nginx
```



Service - NodePort

service-definition.yml

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

spec:
    type: NodePort
    ports:
     - targetPort: 80
       port: 80
       nodePort: 30008
    selector:
       app: myapp
       type: front-end
```

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

Service - NodePort

service-definition.yml

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

spec:
    type: NodePort
    ports:
     - targetPort: 80
       port: 80
       nodePort: 30008
    selector:
       app: myapp
       type: front-end
```

```
> 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
```

Service - NodePort

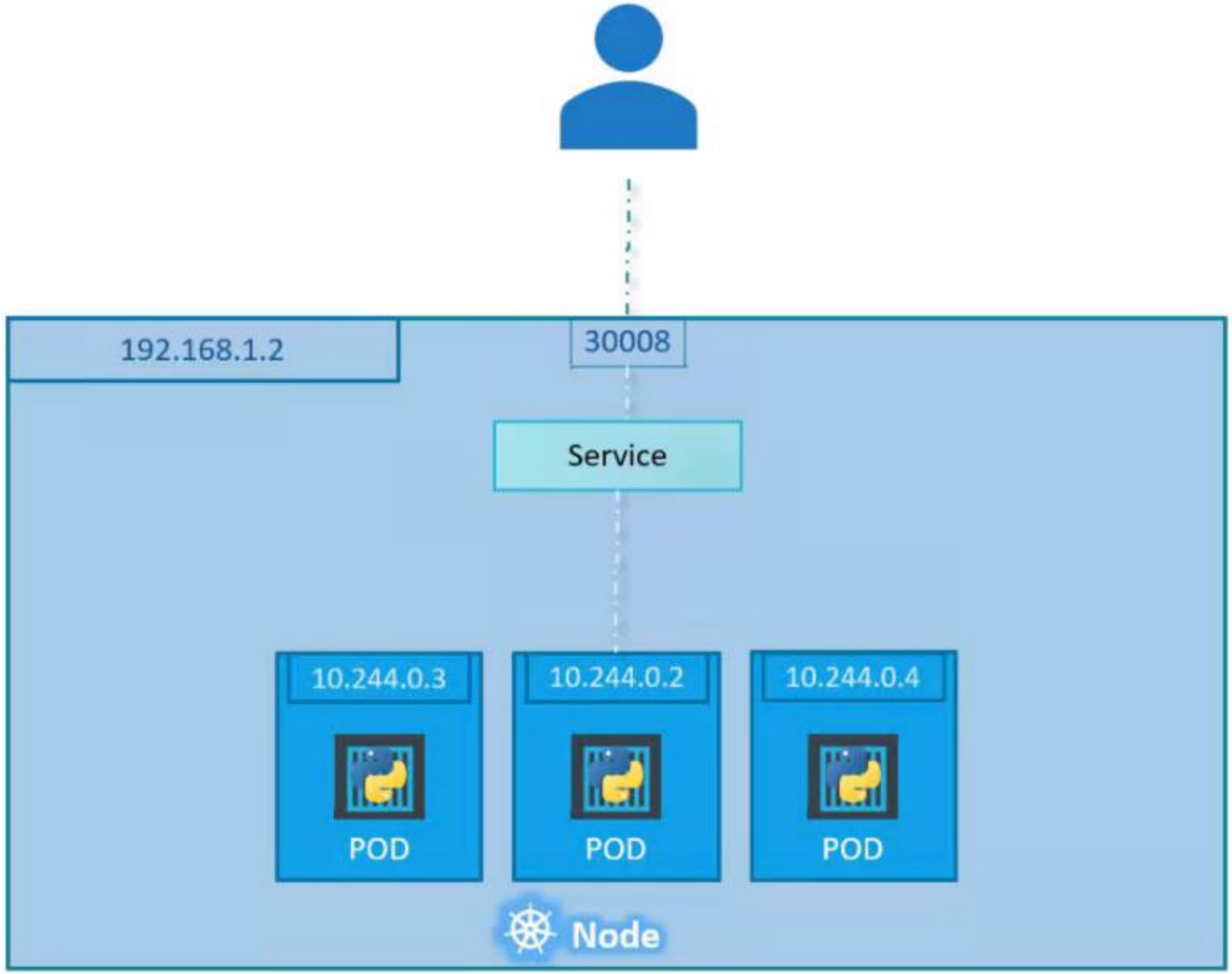
service-definition.yml

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

spec:
    type: NodePort
    ports:
     - targetPort: 80
       port: 80
       nodePort: 30008
    selector:
       app: myapp
       type: front-end
```

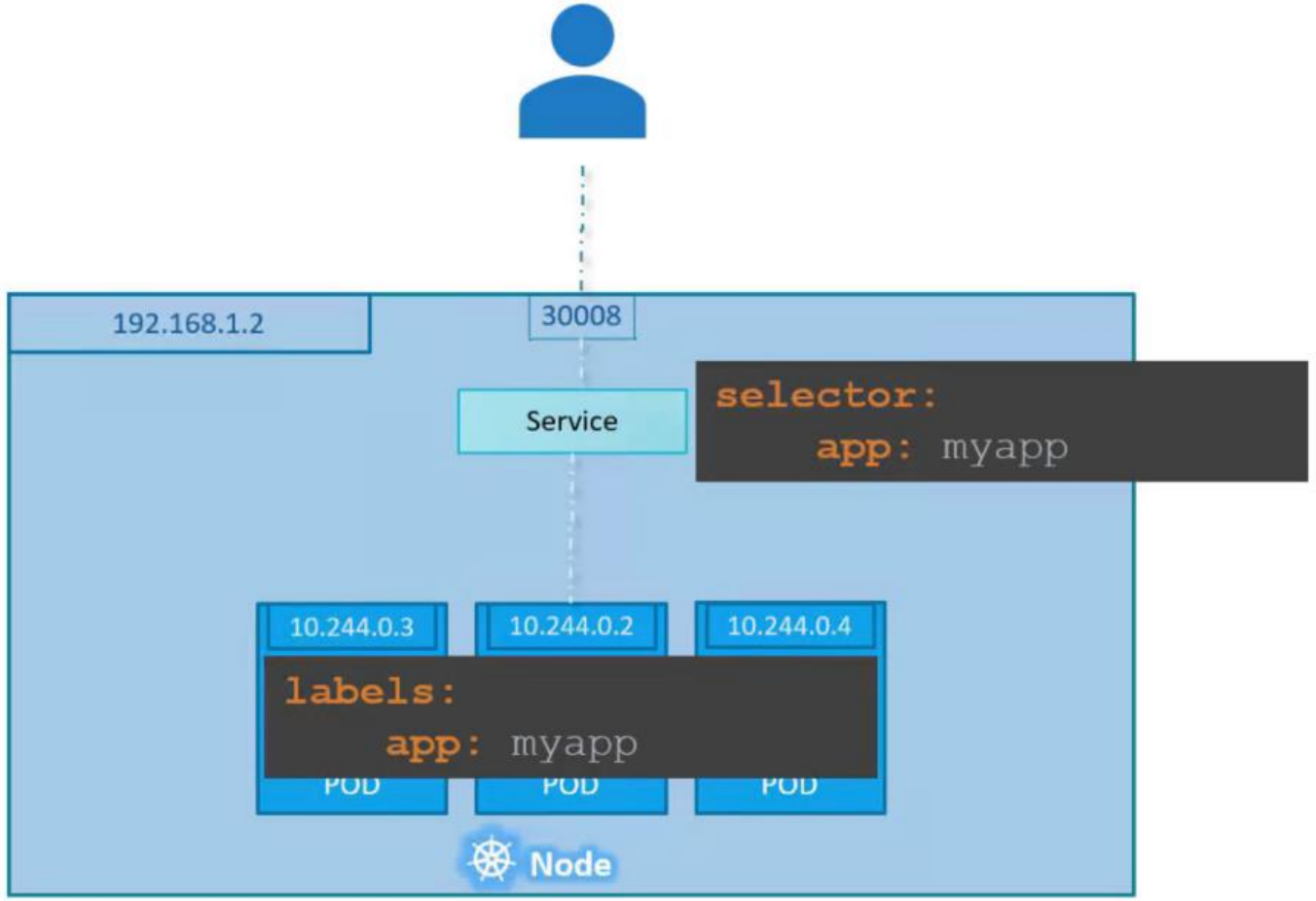
```
> kubectl create -f service-definition.yml
service "myapp-service" created
```

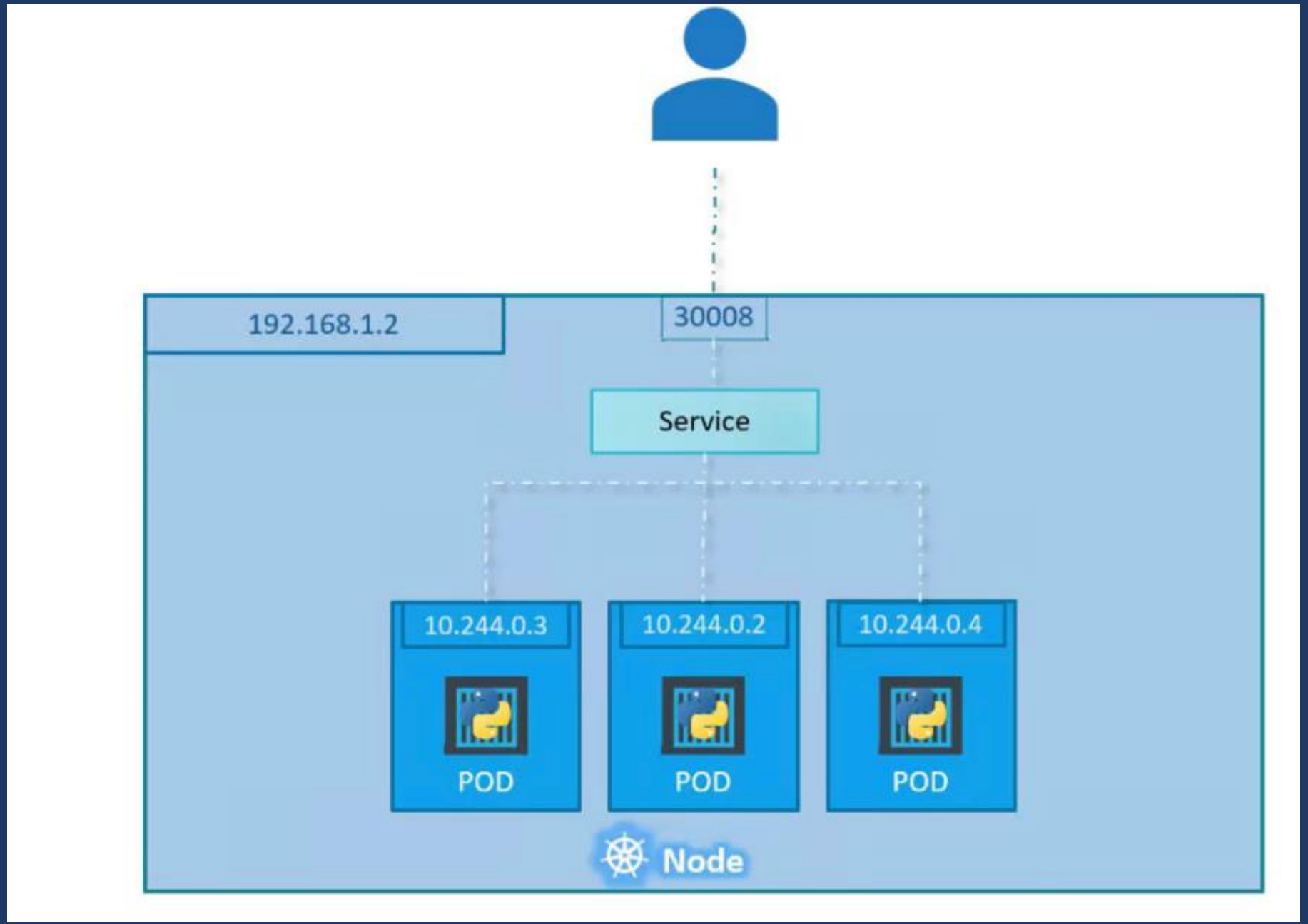
```
> kubectl get services
NAME            TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
kubernetes      ClusterIP   10.96.0.1        <none>        443/TCP          16d
myapp-service   NodePort    10.106.127.123   <none>        80:30008/TCP     5m
```

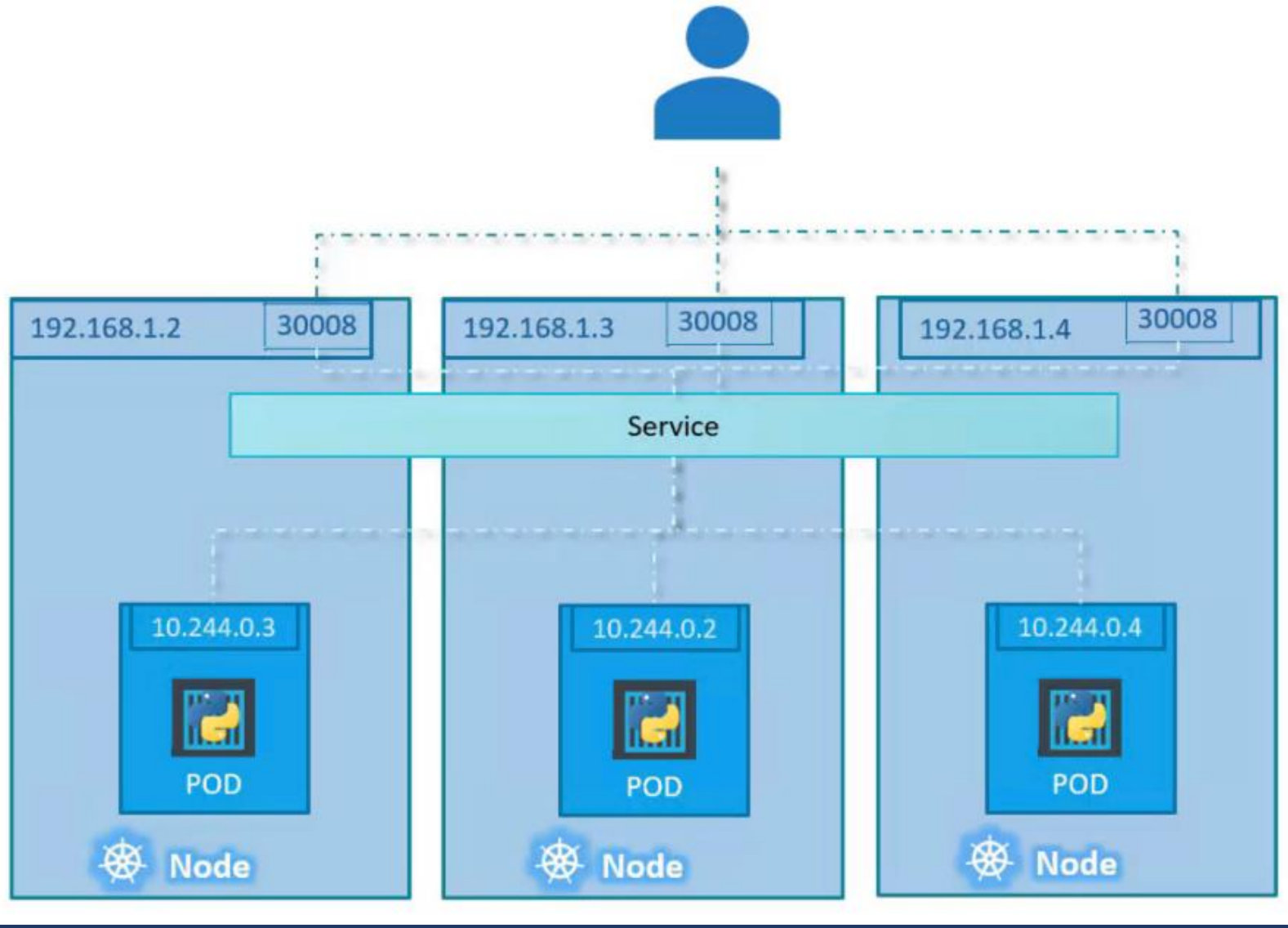
```
> curl http://192.168.1.2:30008
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
```

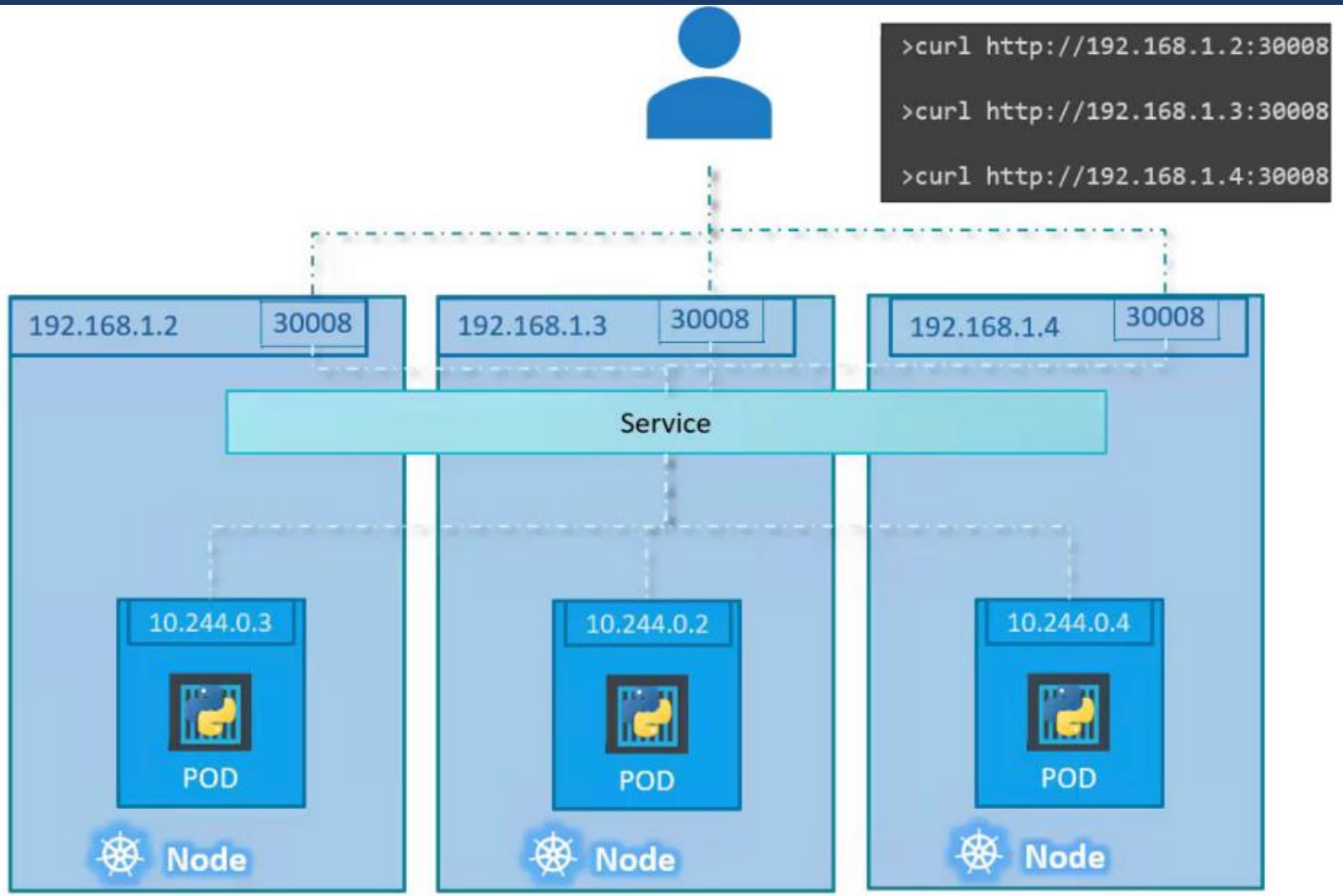

192.168.1.2
30008
Service
10.244.0.3
10.244.0.2
10.244.0.4
POD
POD
POD
Node

Service - NodePort



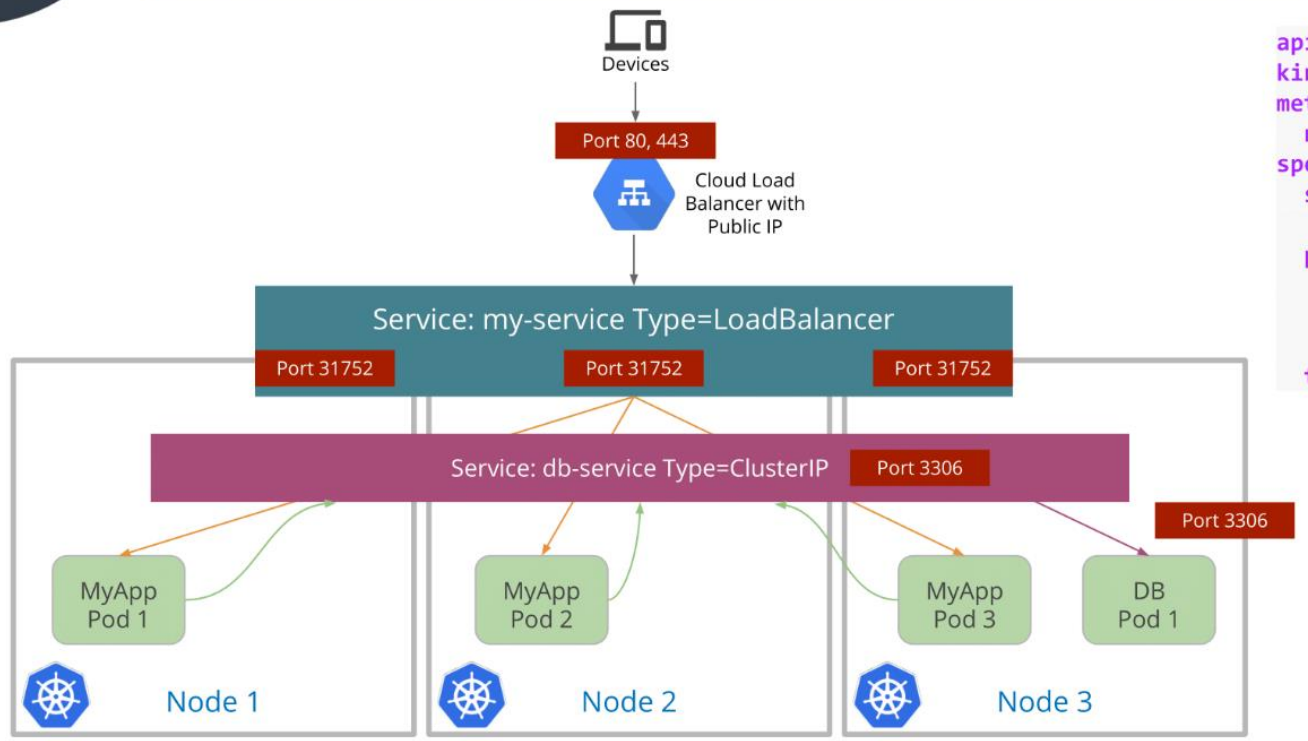

192.168.1.2
30008
Service
10.244.0.3
10.244.0.2
10.244.0.4
POD
POD
POD
Node


192.168.1.2
30008
192.168.1.3
30008
192.168.1.4
30008
Service
10.244.0.3
POD
10.244.0.2
POD
10.244.0.4
POD
Node
Node
Node

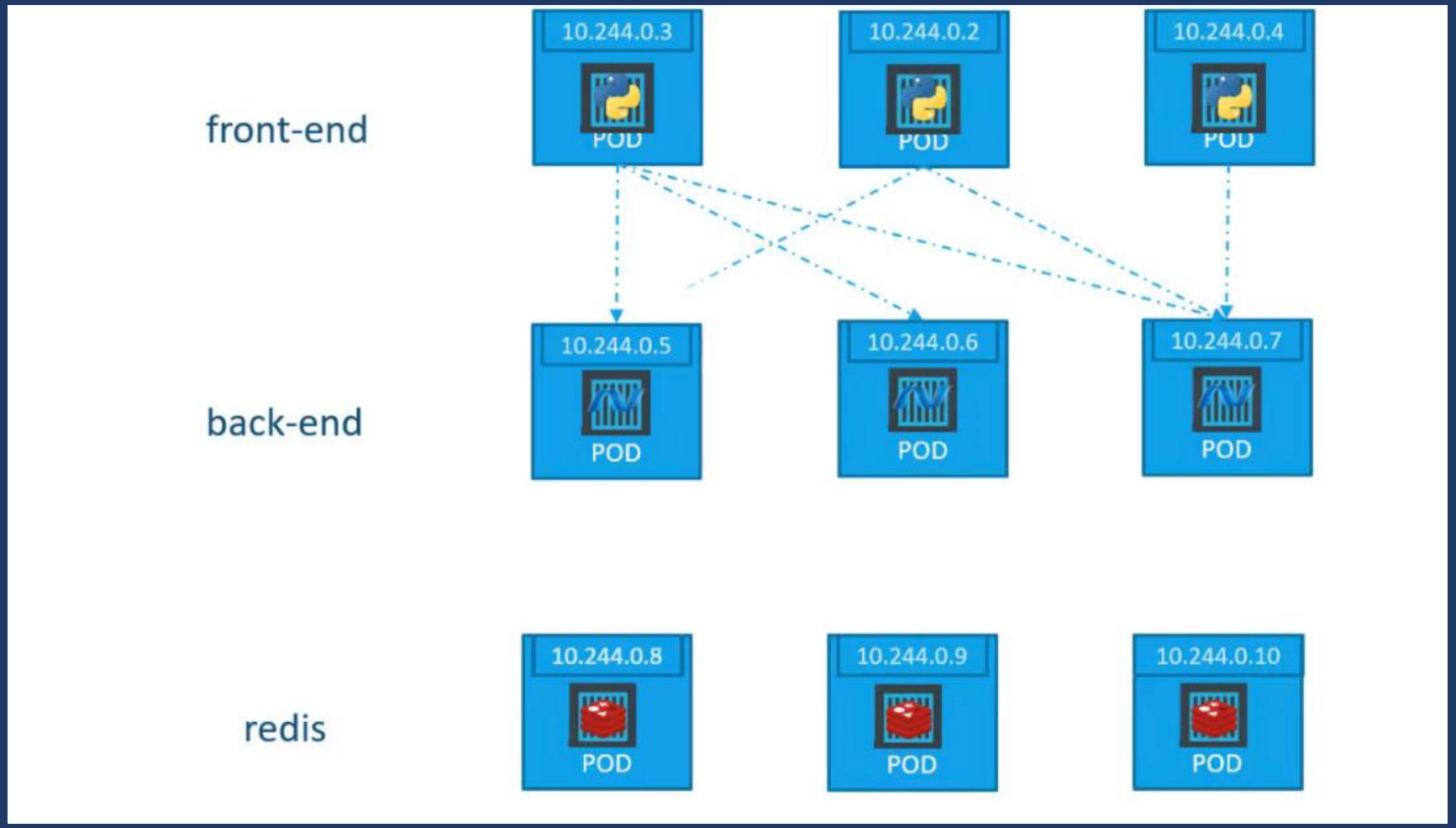

>curl http://192.168.1.2:30008
>curl http://192.168.1.3:30008
>curl http://192.168.1.4:30008
192.168.1.2
30008
192.168.1.3
30008
192.168.1.4
30008
Service
10.244.0.3
POD
10.244.0.2
POD
10.244.0.4
POD
Node
Node
Node

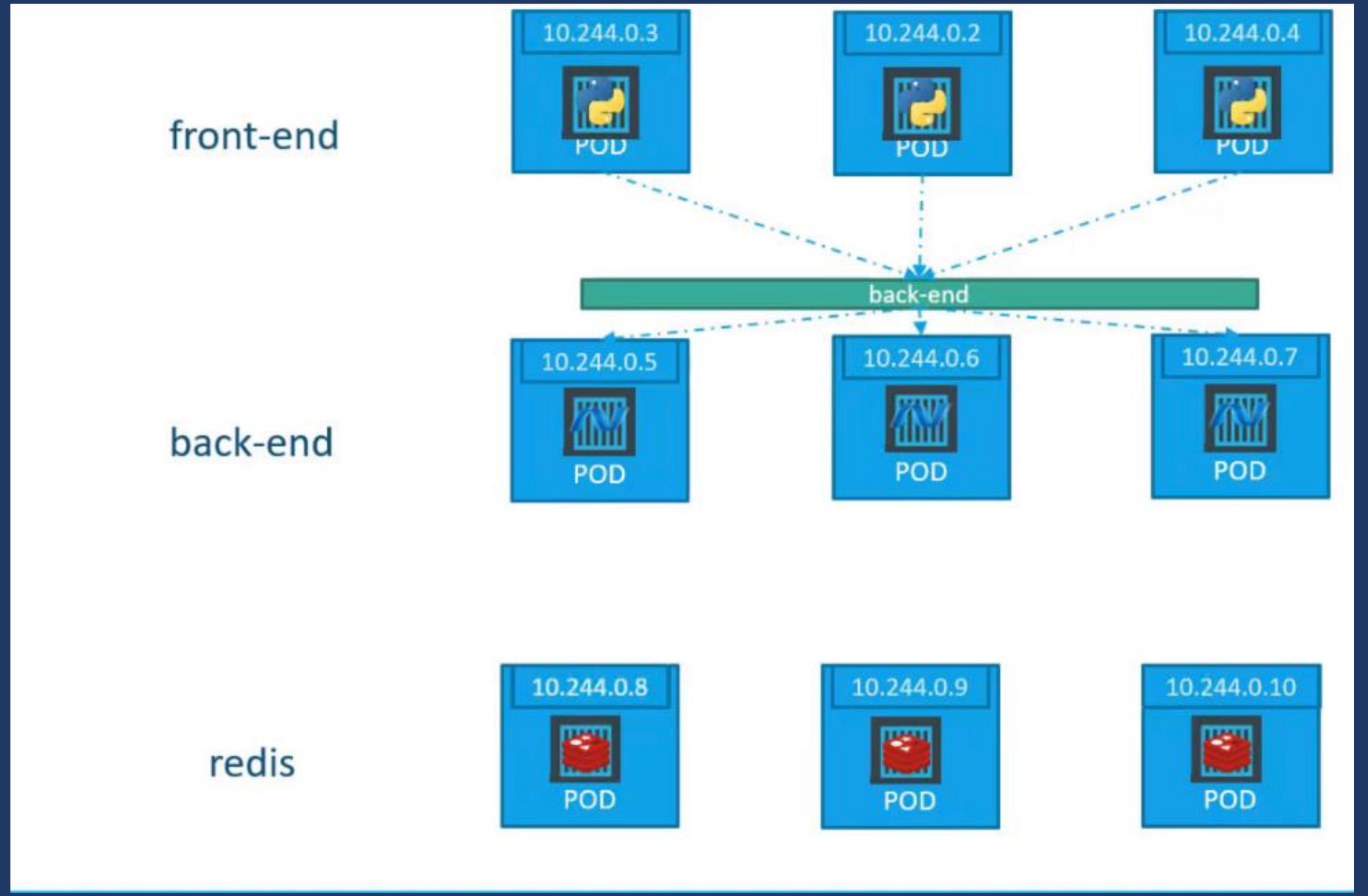
ClusterIP

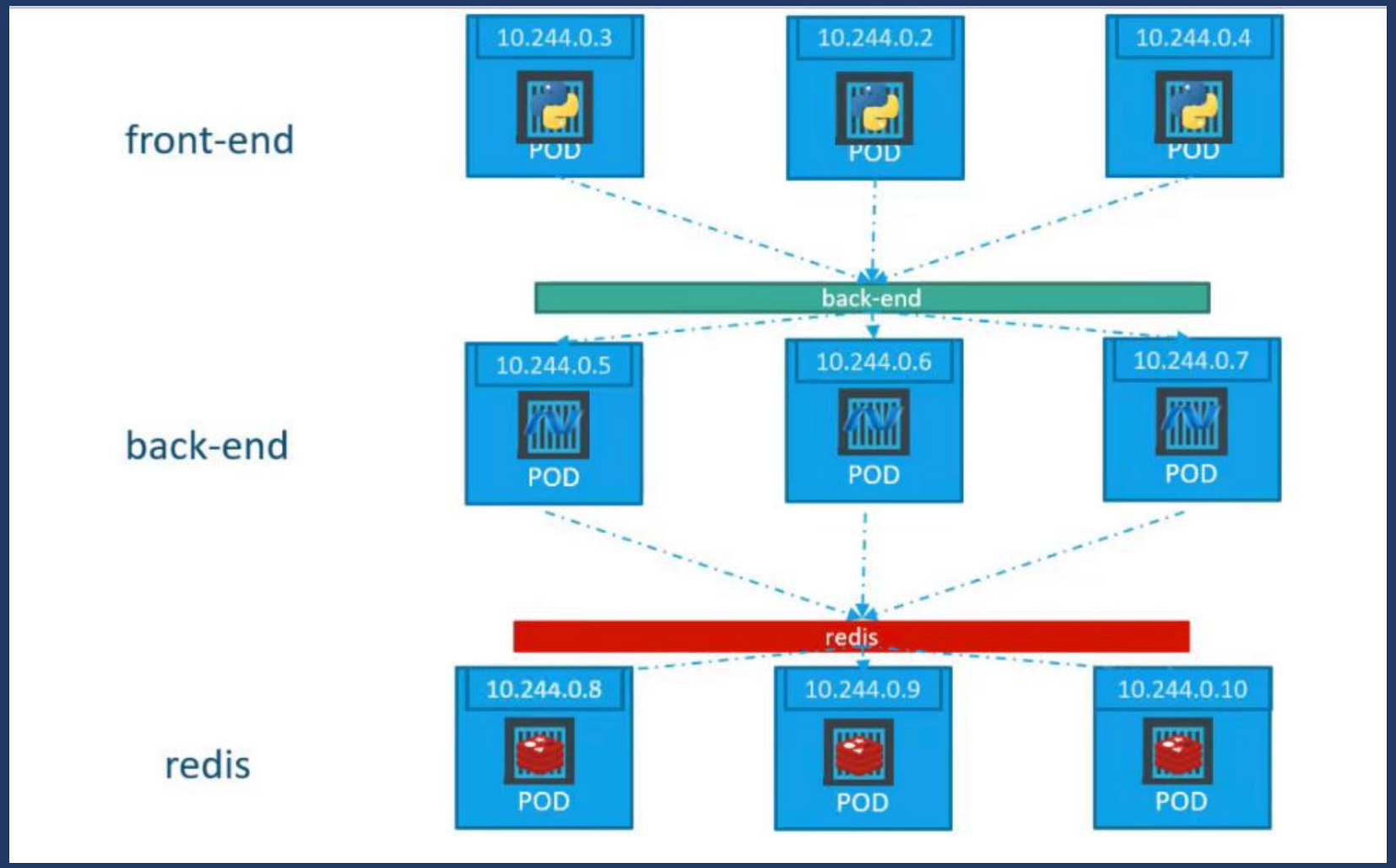
Service Type: ClusterIP




```yaml
apiVersion: v1
kind: Service
metadata:
  name: db-service
spec:
  selector:
    app: DB
  ports:
    - protocol: TCP
      port: 3306
      targetPort: 3306
  type: ClusterIP
```


front-end
10.244.0.3
POD
10.244.0.2
POD
10.244.0.4
POD
back-end
10.244.0.5
POD
10.244.0.6
POD
10.244.0.7
POD
redis
10.244.0.8
POD
10.244.0.9
POD
10.244.0.10
POD


front-end
10.244.0.3
POD
10.244.0.2
POD
10.244.0.4
POD
back-end
back-end
10.244.0.5
POD
10.244.0.6
POD
10.244.0.7
POD
redis
10.244.0.8
POD
10.244.0.9
POD
10.244.0.10
POD


front-end
10.244.0.3
POD
10.244.0.2
POD
10.244.0.4
POD
back-end
back-end
10.244.0.5
POD
10.244.0.6
POD
10.244.0.7
POD
redis
redis
10.244.0.8
POD
10.244.0.9
POD
10.244.0.10
POD

service-definition.yml

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

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

service-definition.yml

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

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

    selector:
      app: myapp
      type: back-end
```

pod-definition.yml

```yaml
apiVersion: v1
kind: Pod

metadata:
 name: myapp-pod
 labels:


spec:
  containers:
  - name: nginx-container
    image: nginx
```

service-definition.yml

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

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

    selector:
       app: myapp
       type: back-end
```

```
> 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
```