Module 5

Docker & Kubernetes

Docker

✓ Docker allows developers to **package software** in **portable containers.**

✓ Containers ensure consistent **running of software on any system.**

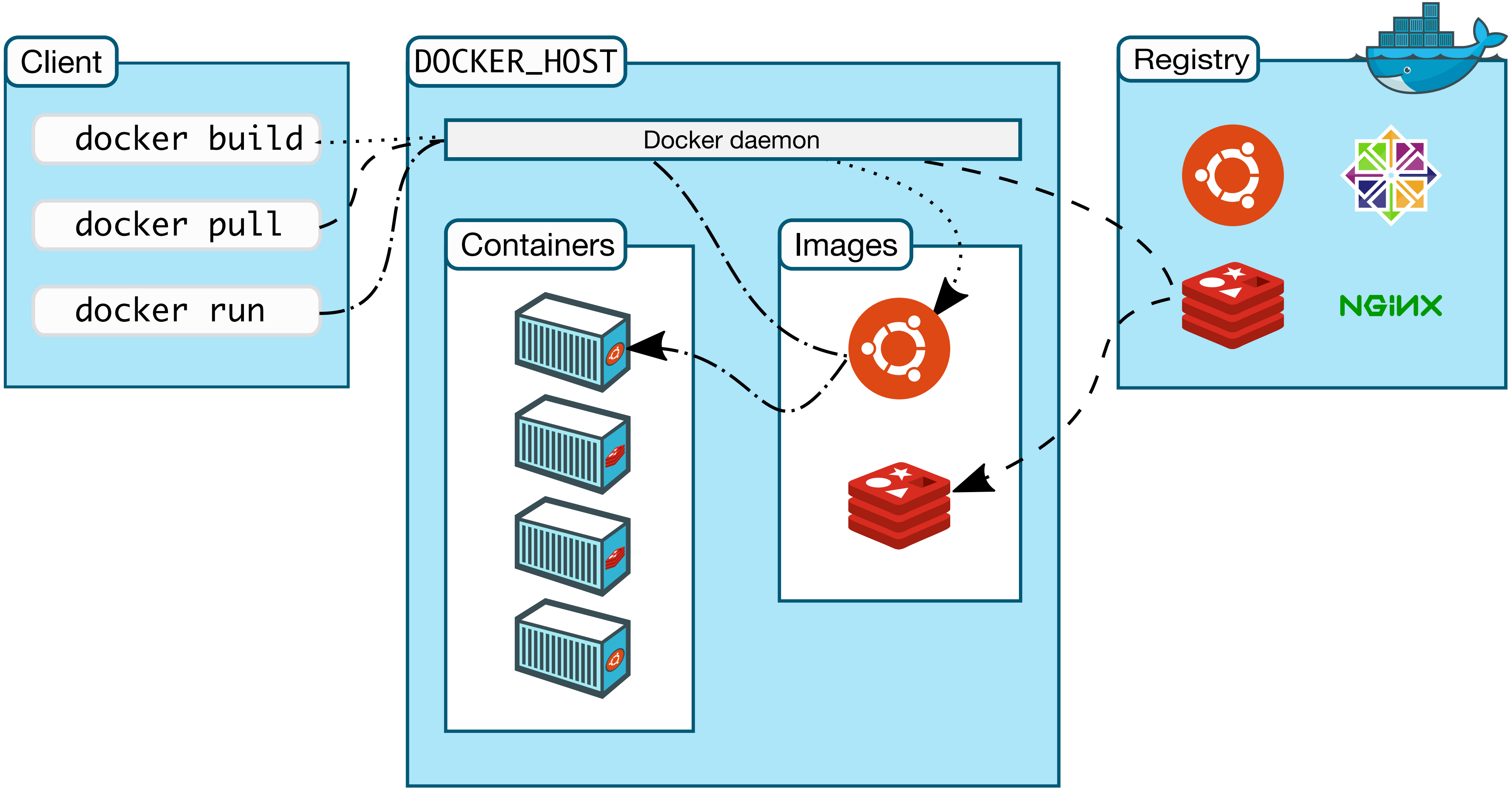
✓ Docker provides tools to **build**, **package** and **distribute containers.**

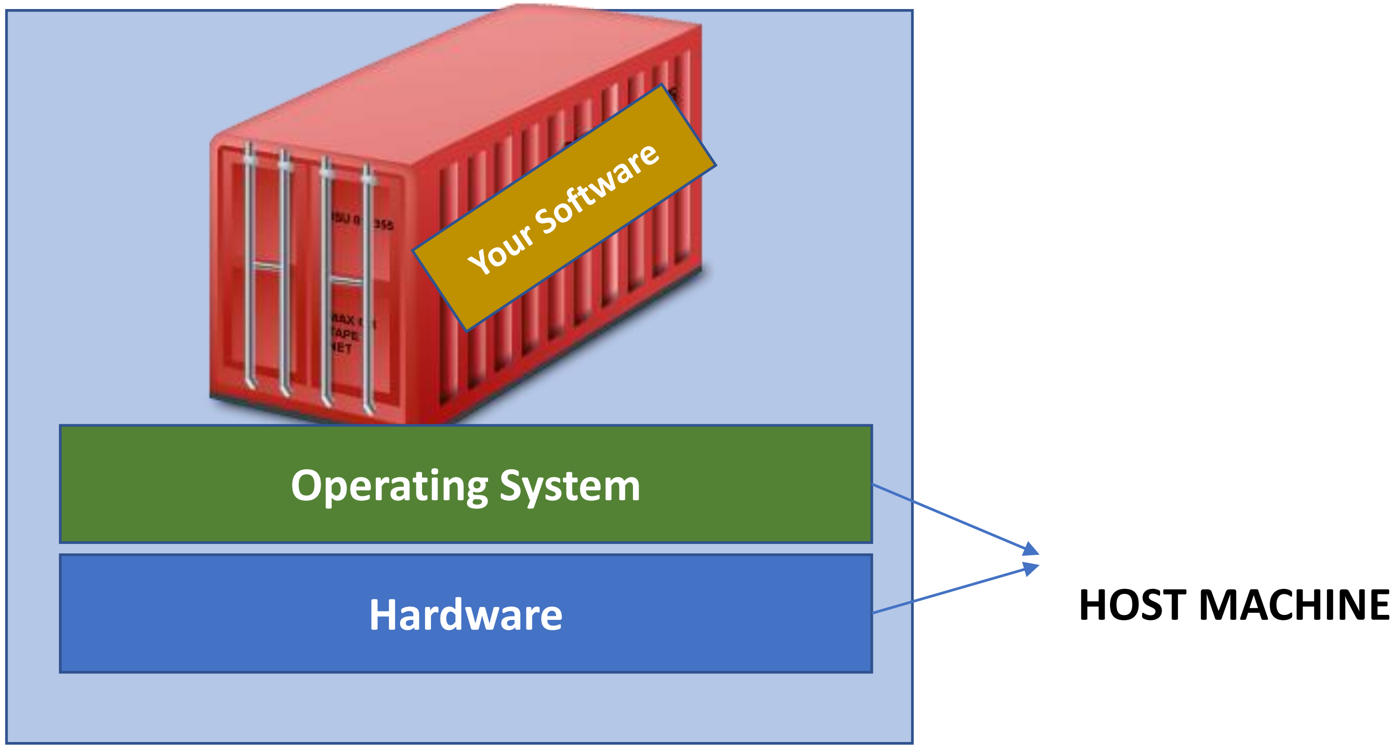
✓ It makes it easy to **move applications** between **different environments.**

✓ Docker also provides tools to **manage** and **orchestrate containers.**

✓ It i**mproves developer productivity** and **reduce the difficulty of deploying software.**

How it works ?


Client
docker build
docker pull
docker run
DOCKER_HOST
Docker daemon
Containers
Images
Registry
NGiNX



Your Software

Operating System

Hardware

HOST MACHINE

Steps

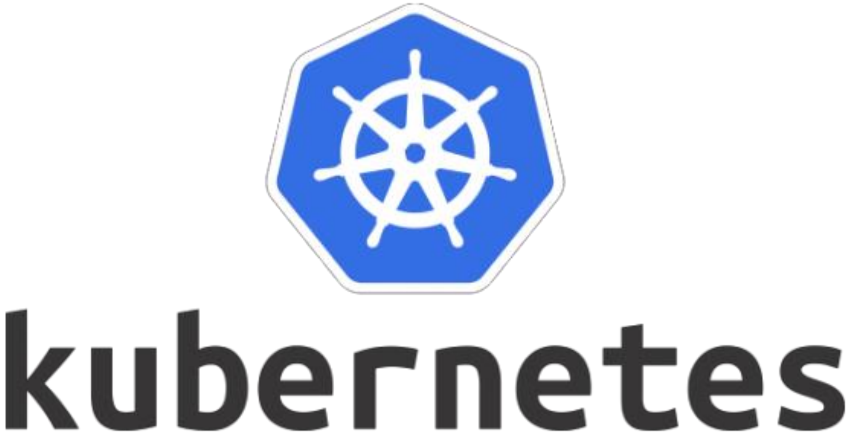
Docker file

Image

Container

Kubernetes

- Kubernetes is an open-source platform for automating deployment, scaling, and management of containerized applications.
- It was originally developed by Google, and is now maintained by the Cloud Native Computing Foundation (CNCF).



How it works ?

- Manages containers, which are **lightweight** and **portable software units.**
- Provides a unified way to **define**, **deploy**, and **manage containers.**
- Uses **pods** as logical units for **containers** and **deployments** for **groups of pods.**
- **Automates scaling**, **rollouts**, and **rollbacks** of applications.
- Manages **network** and **storage** resources needed by containers.
- Helps **build**, **deploy**, and **run applications** at scale.

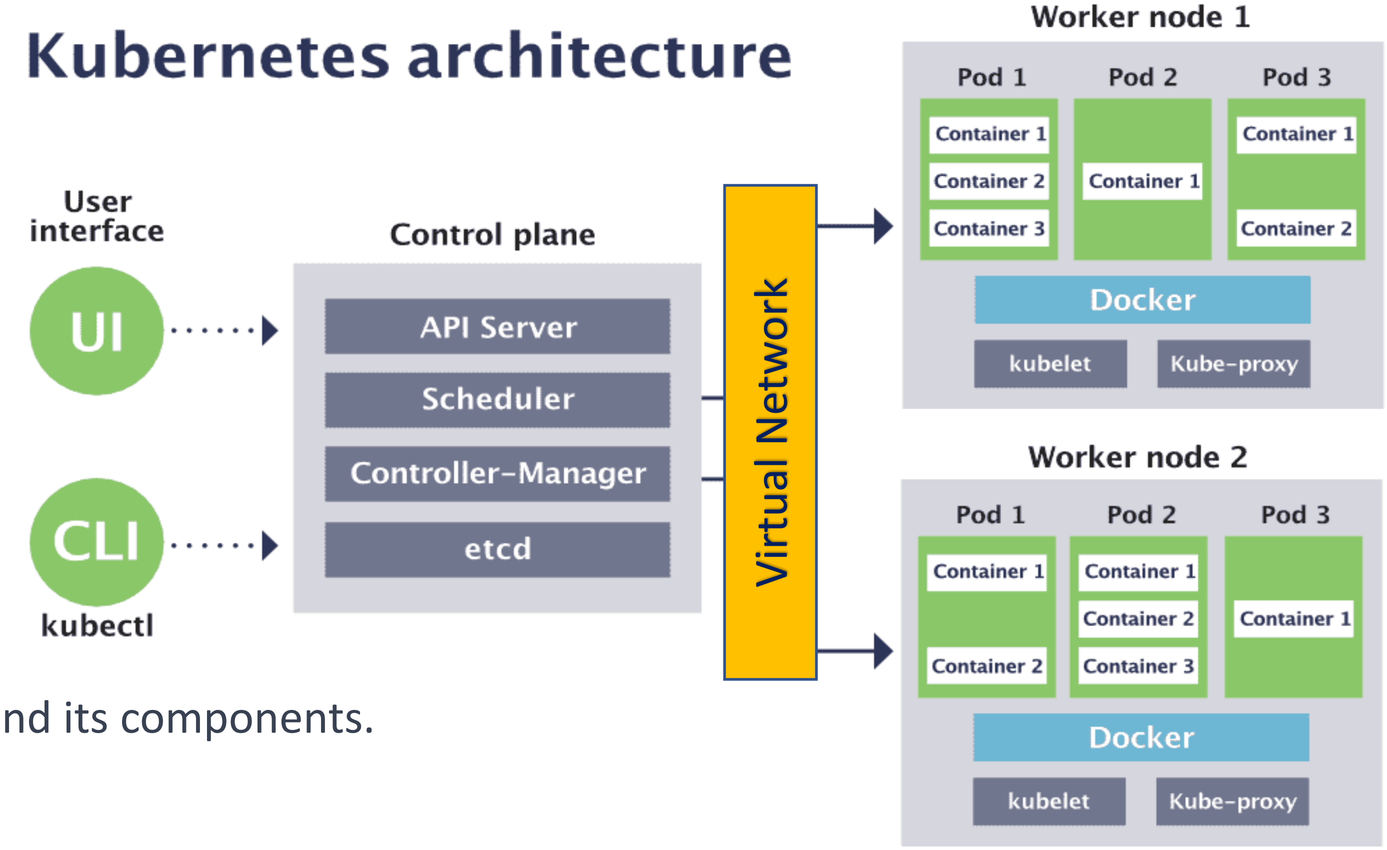
Kubernetes features

- Managing **availability**, **security**, and **performance of applications.**
- Includes **automatic load balancing** and **self-healing.**
- Supports **rolling upgrades.**
- **Disaster recovery**
- Helps **build**, **deploy**, and **run applications** at scale.
- Popular platform for deploying **cloud-native applications.**

Architecture of Kubernetes

Master Node/control plane

Central control plane that manages and orchestrates the operations of the cluster.



API Server

- Exposing the API used by other components.
- Storing the shared state of the cluster.

Controller Manager

- Controllers regulate the state of the cluster and its components.
- They are a set of control loops.
- Examples of controllers include:
  - Replicating pods.
  - Tracking the status of nodes.
  - Managing the lifecycle of individual objects.
- Controllers help maintain the desired state of the cluster.

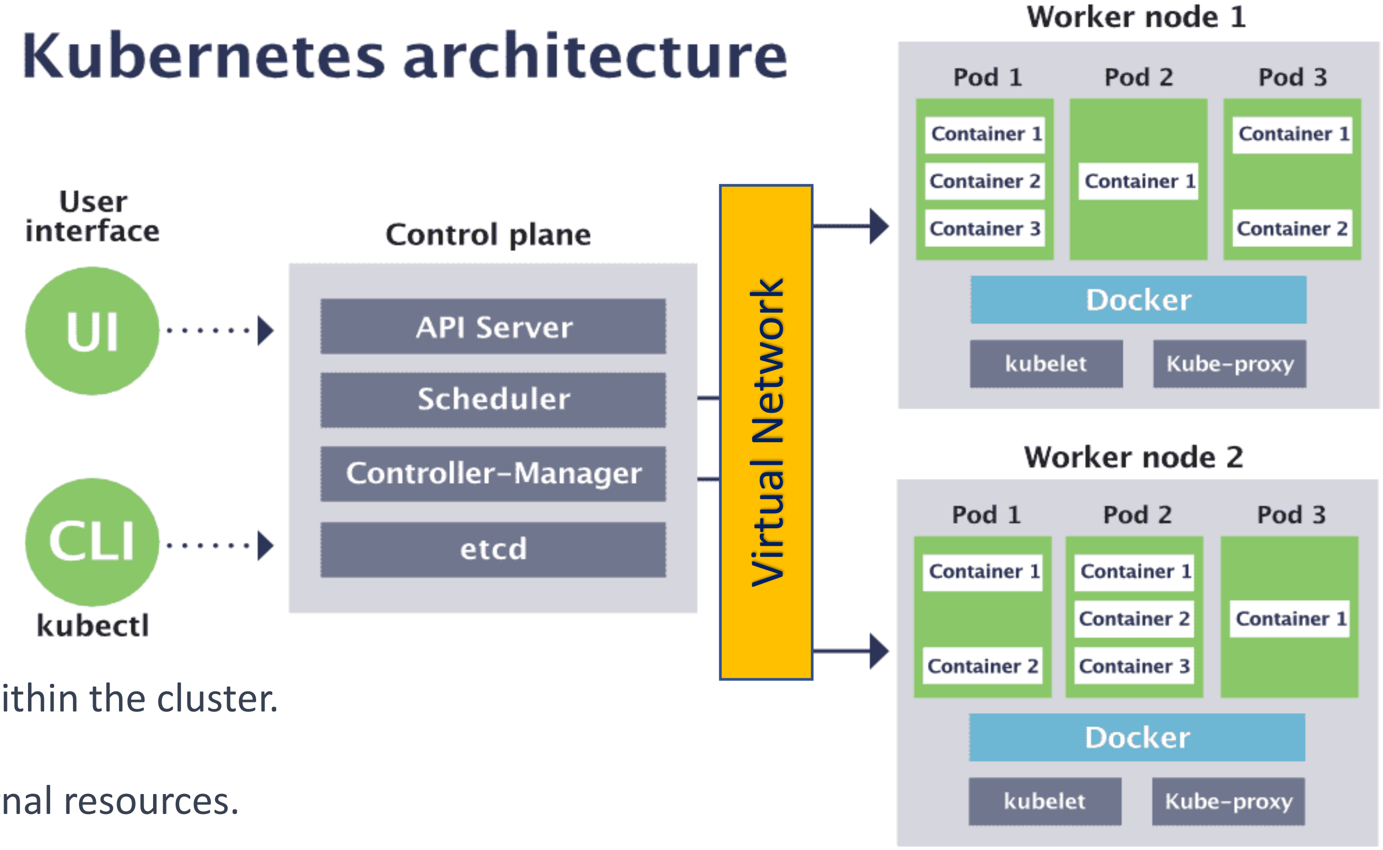
Architecture of Kubernetes

Scheduler

Determines which nodes in the cluster should run each pod.

etcd

An consistent and highly available key-value store that holds all of the cluster's configuration data.



Virtual Network

- Virtual network in Kubernetes is a networking setup within the cluster.
- It allows communication between pods and with external resources.
- The virtual network provides a logical network space separate from the physical network.
- It enables pods to communicate with each other and access external services.

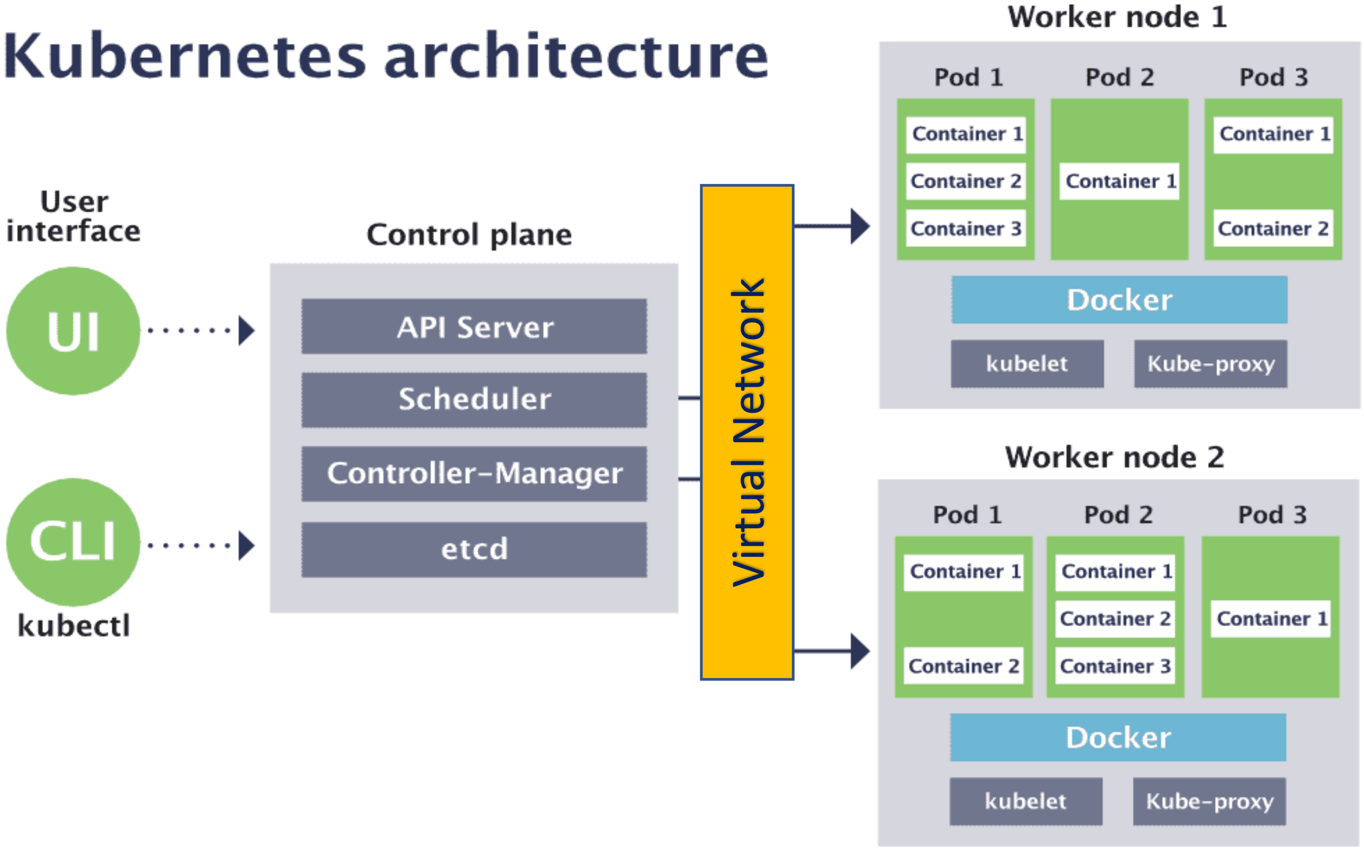
Architecture of Kubernetes

Worker nodes

- ✓ Worker nodes run applications and workloads in a Kubernetes cluster.
- ✓ They are managed by master nodes.
- ✓ Master nodes coordinate and schedule the activities of worker nodes.

Kubelet

- ✓ An agent that runs on each node and is responsible for maintaining the state of the pods running on that node.



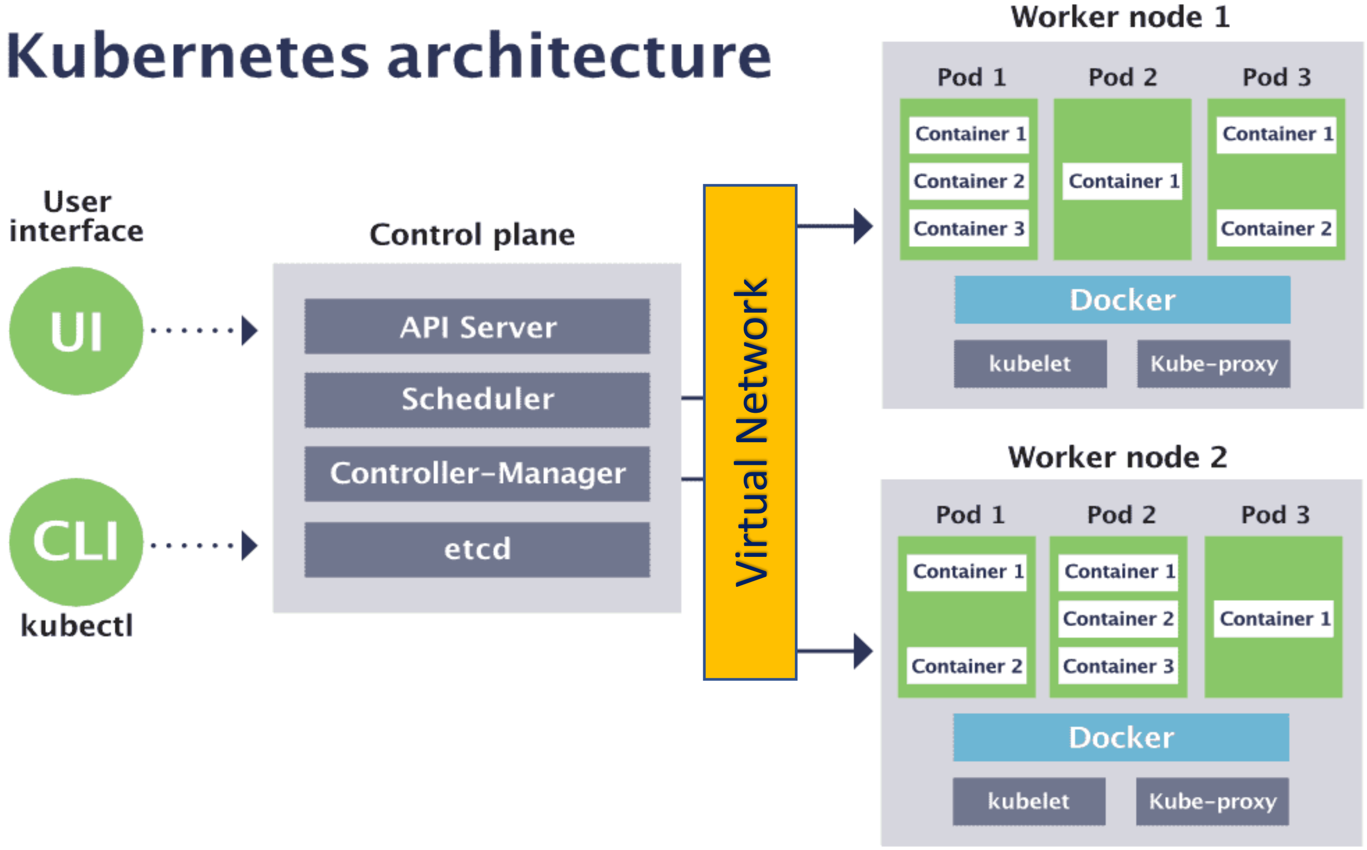
Architecture of Kubernetes

Kube-proxy

- ✓ Kube-proxy is a part of Kubernetes that helps with **network connectivity** for the pods by **directing traffic** and **acting as a load balancer.**
- ✓ **TCP** and **UDP** stream forwarding

kubectl

- ✓ The command-line tool used to interact with the Kubernetes API.



Container Runtime

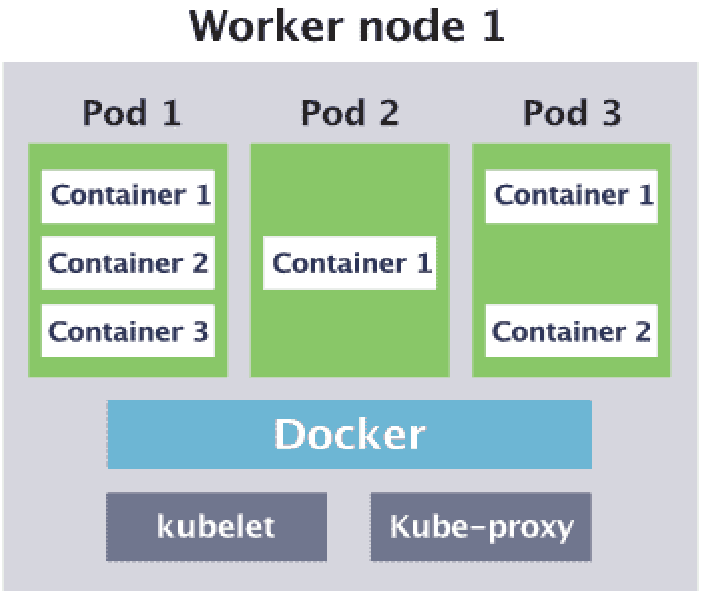
- ✓ The component **responsible** for **starting** and **stopping containers on nodes**. Commonly used runtimes include Docker and Containerd.

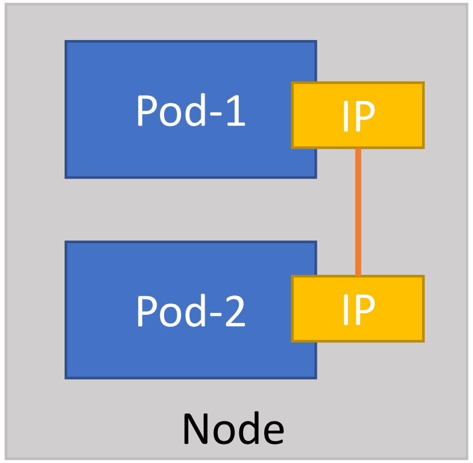
Kubernetes resources

- ✓Pod
- ✓ConfigMap
- ✓Service
- ✓Secret
- ✓Ingress
- ✓Deployment
- ✓StatefulSet
- ✓DaemonSet

Pod

- ✓ A pod is the smallest unit in the Kubernetes object model
- ✓ A pod represents a **single instance** of a **running process in a cluster**
- ✓ Pods host **one or more containers**, which are the actual running instances of the application
- ✓ Pods provide an **isolated environment** for the containers
- ✓ Pods can be **created** or **destroyed** as needed for scaling.





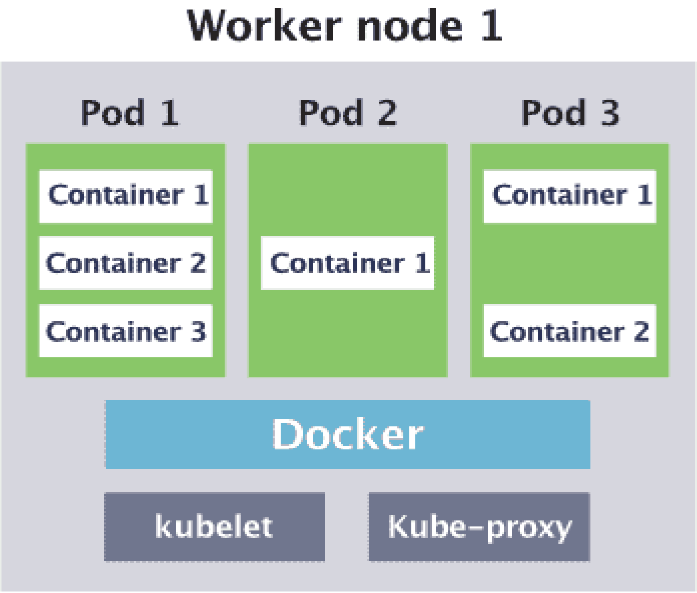
Service

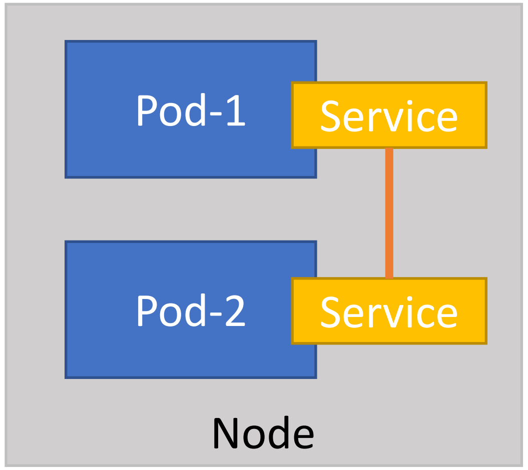
- ✓ A Service in Kubernetes **provides network communication between pods** and **external resources.**

- ✓ It **hides the network details** and **pod identities** and provides a **stable IP address and DNS name** for easier communication.

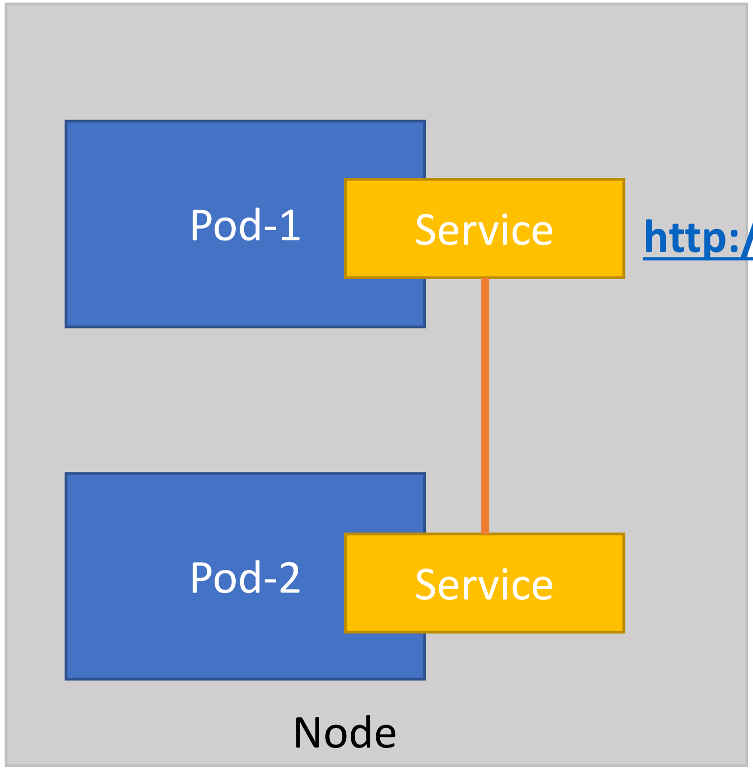
- ✓ The Service is **designed to handle changes in the IP addresses of the pods.**

- ✓ There are various types of Services available, each suited for different network communication needs.





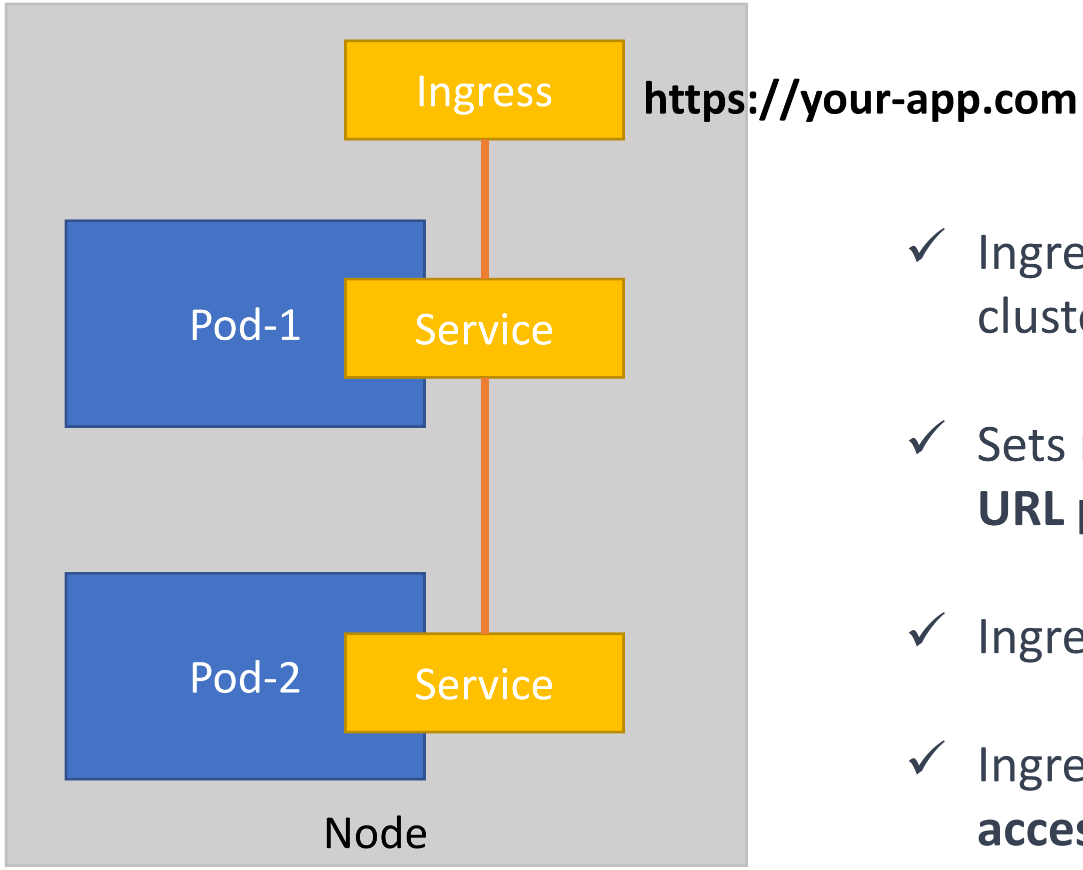
Service



http://10.21.65.251:8080 → https://your-app.com

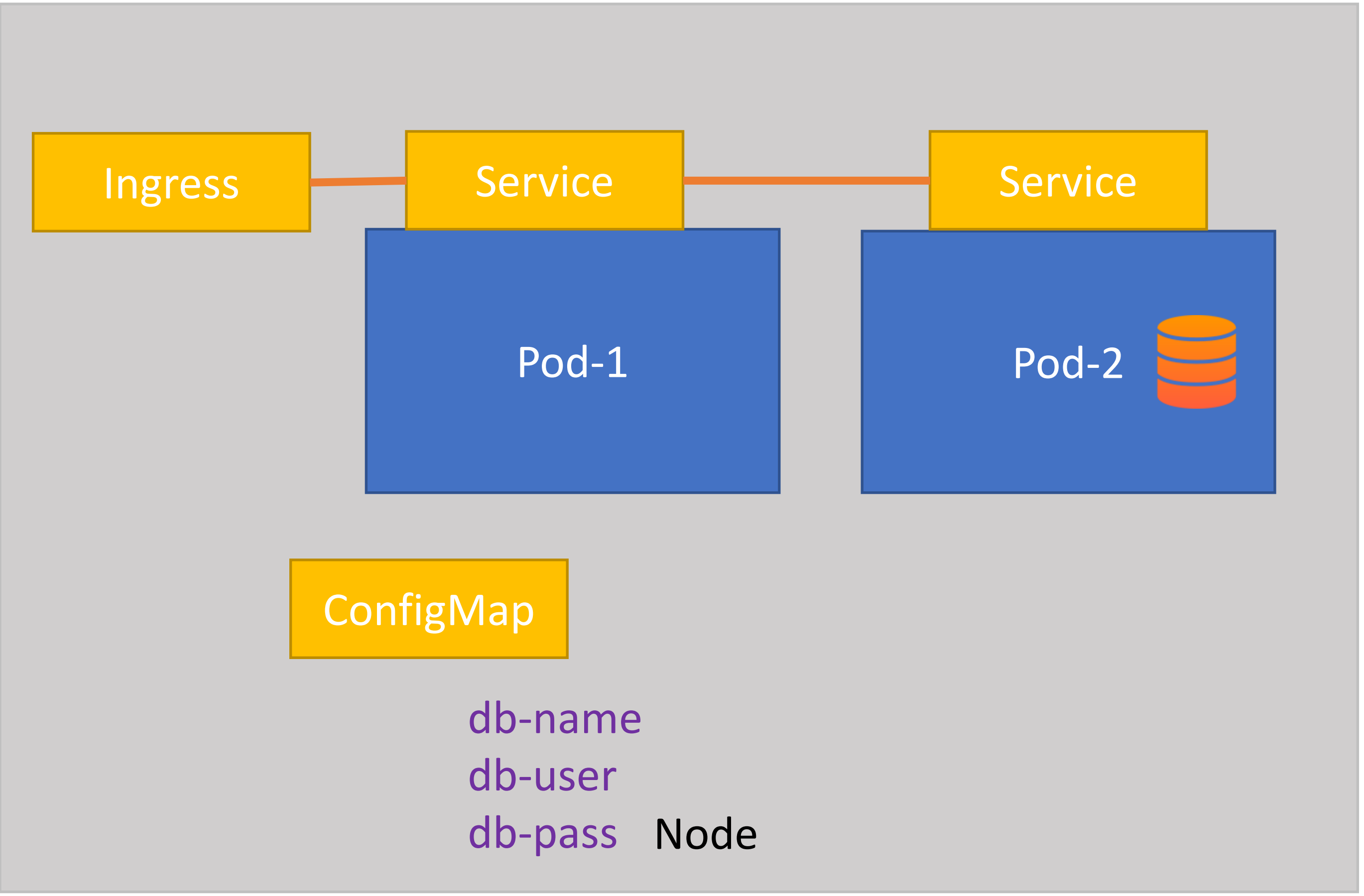
✓ This can be managed by another component of Kubernetes called Ingress

Ingress



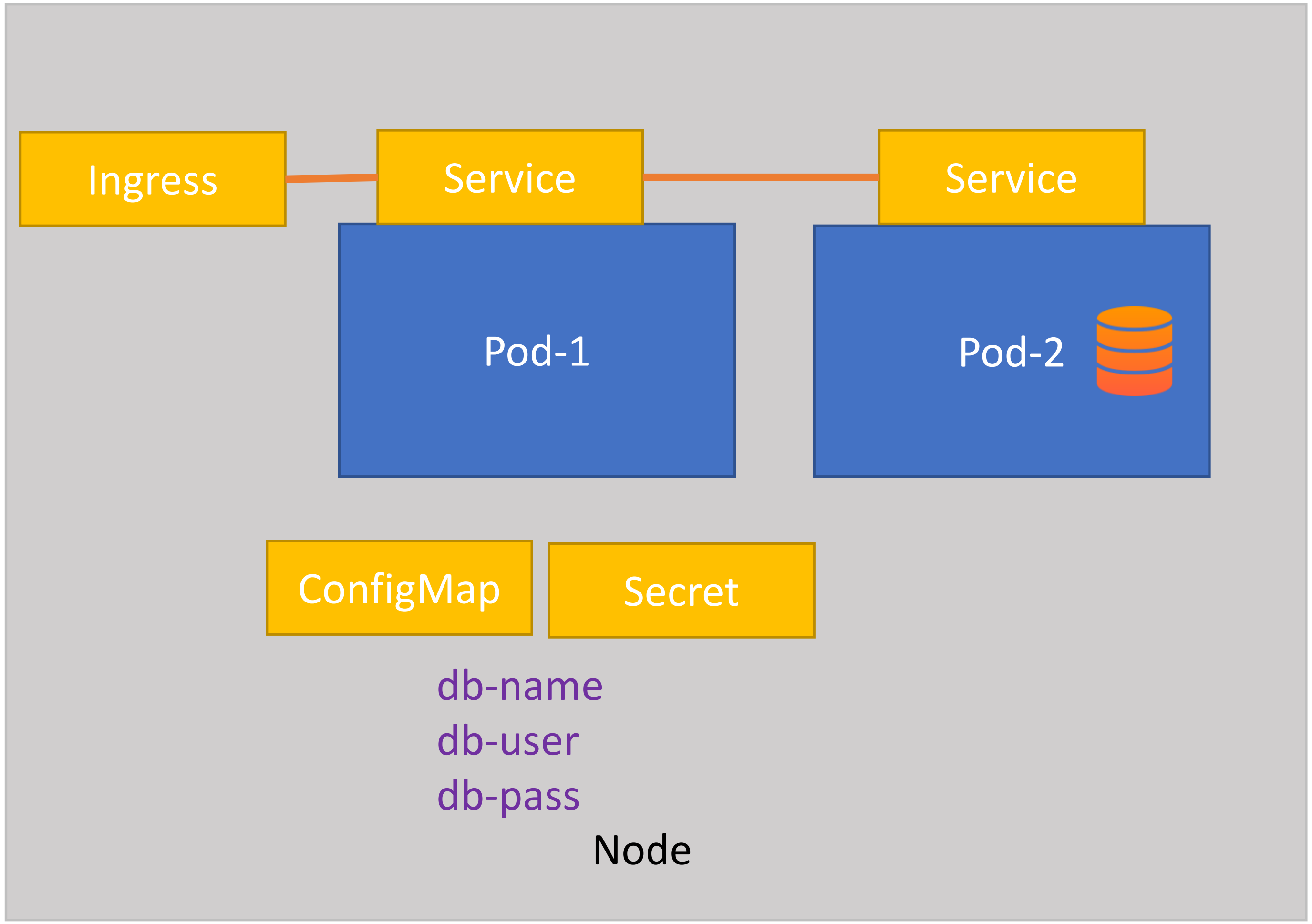
- ✓ Ingress **controls incoming traffic** to **services** in a cluster
- ✓ Sets rules for **routing traffic based on hostname** and **URL path**
- ✓ Ingress controller enforces the rules and directs traffic
- ✓ Ingress resource makes it easier to **manage external access to services** in the cluster

ConfigMap



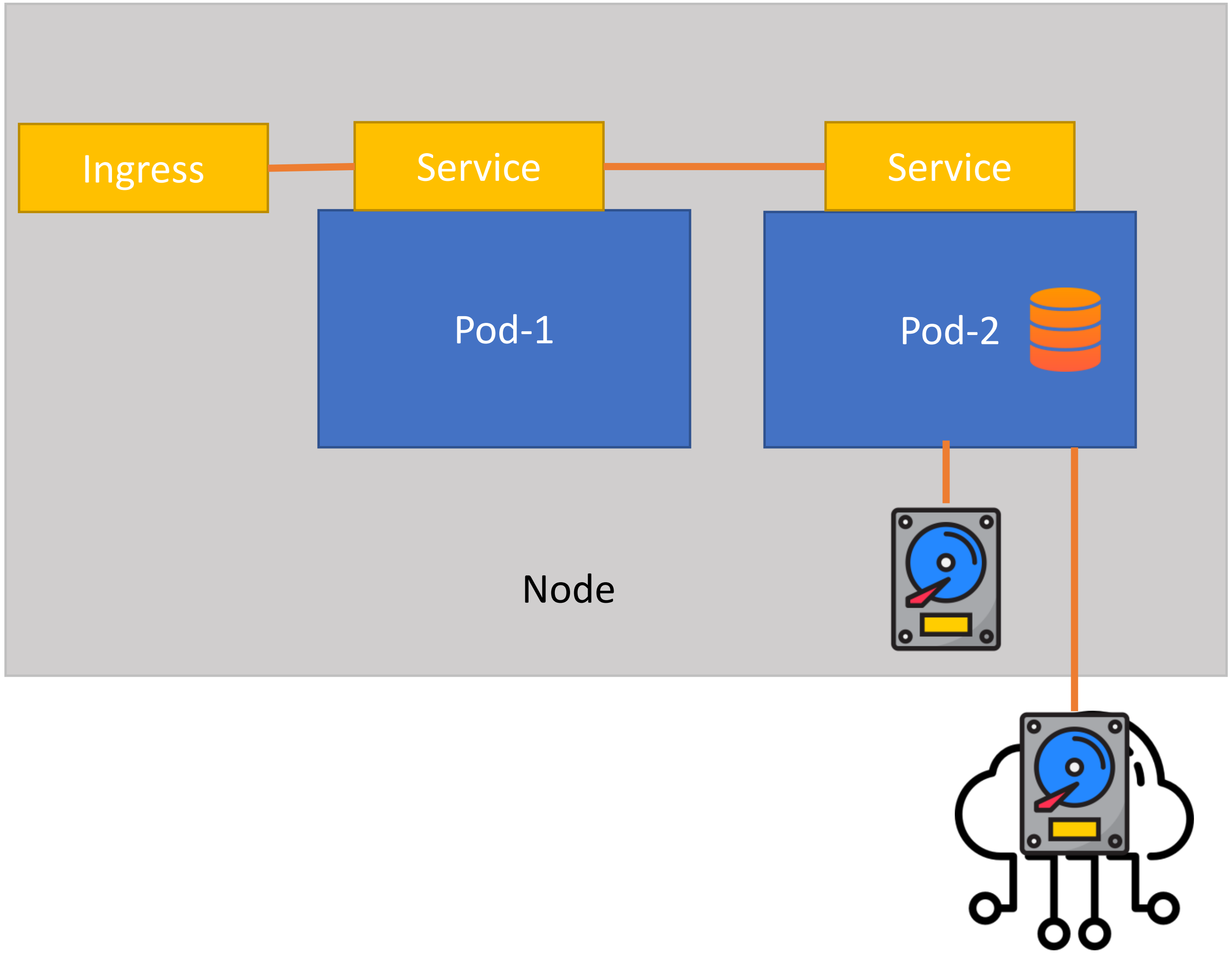
- ✓ ConfigMap is a resource in Kubernetes for **storing configuration data**
- ✓ The data is stored as **key-value pairs**
- ✓ The data can be used by **containers** and **system components** in the cluster
- ✓ ConfigMaps help separate **configuration data** from containers, making it **easier to update without affecting them**
- ✓ The data is stored in **etcd** and can be **accessed by pods through environment variables** or **volume mounts.**

Secret



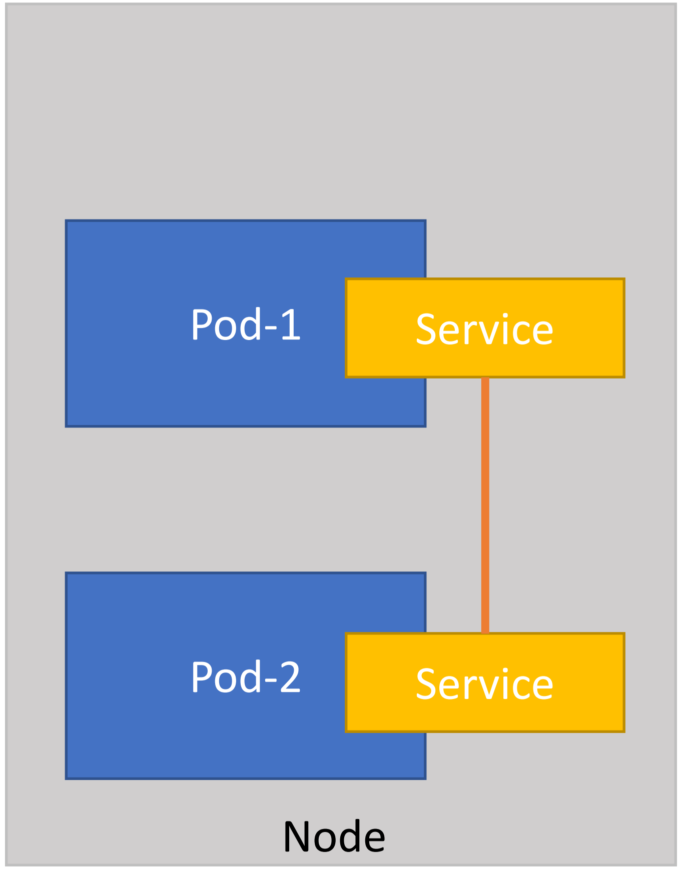
- ✓ **Secrets in Kubernetes** are used to store **sensitive information** such as **passwords**, **tokens**, and **certificates**.
- ✓ **Secrets are encrypted** and **stored in etcd.**
- ✓ **Pods access** Secrets through **environment variables** or **volume mounts.**
- ✓ Secrets store data as **binary data** for added security, unlike ConfigMaps which store configuration data as key-value pairs.

Volume

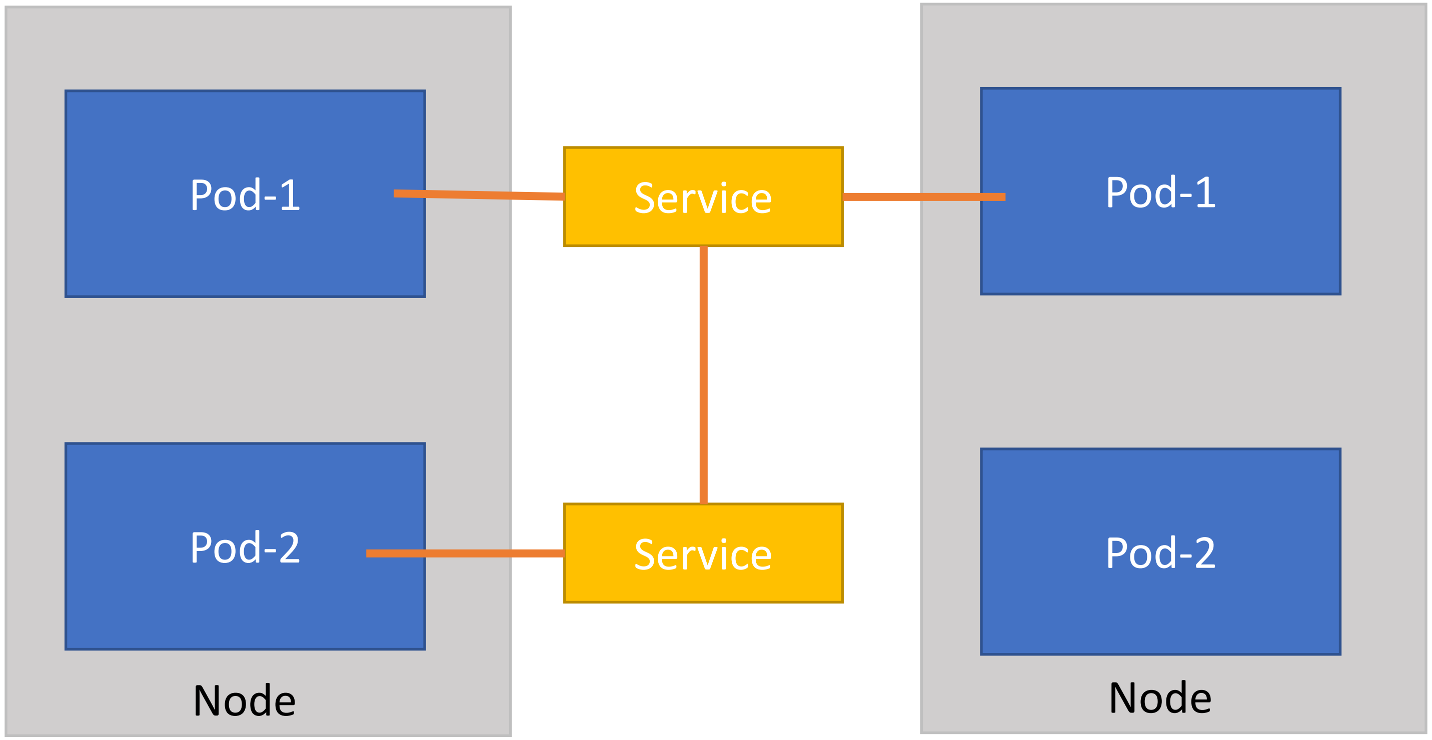


- ✓ Volumes in Kubernetes **are persistent data stores for containers**
- ✓ They **allow containers** to access and **store data even after deletion** or **recreation**
- ✓ Different types of volumes are available, such as **local**, **network attached**, and **cloud storage**
- ✓ Volumes can be mounted as **file systems into a pod** to ensure **data persistence.**

Replication



Replication



Deployment

- ✓ Deployment in Kubernetes manages **multiple copies of an application**
- ✓ Ensures **desired number of copies are running** and **available**
- ✓ Provides features for easier management: **rolling updates**, **rollbacks**, **scaling**, and **pause/resume**.

Rolling updates:

- A deployment allows you to update an application by rolling out the new version gradually to some of the copies, reducing the risk of interruption.

Rollbacks

- With a deployment, if a new version of the application creates issues, you can quickly revert back to a previous version.

Scaling

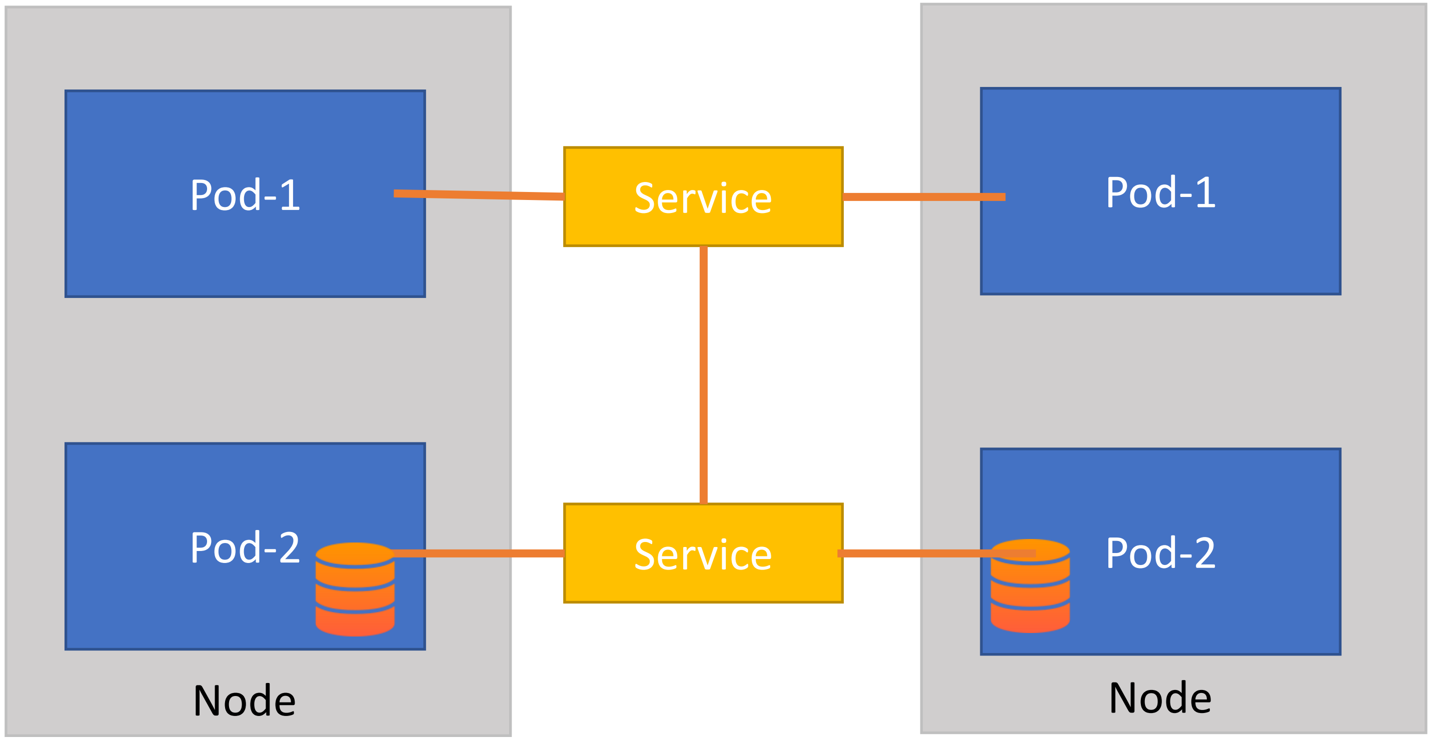
- You can change the number of copies of an application by adjusting the deployment's specifications, making it easy to scale up or down.

Deployment

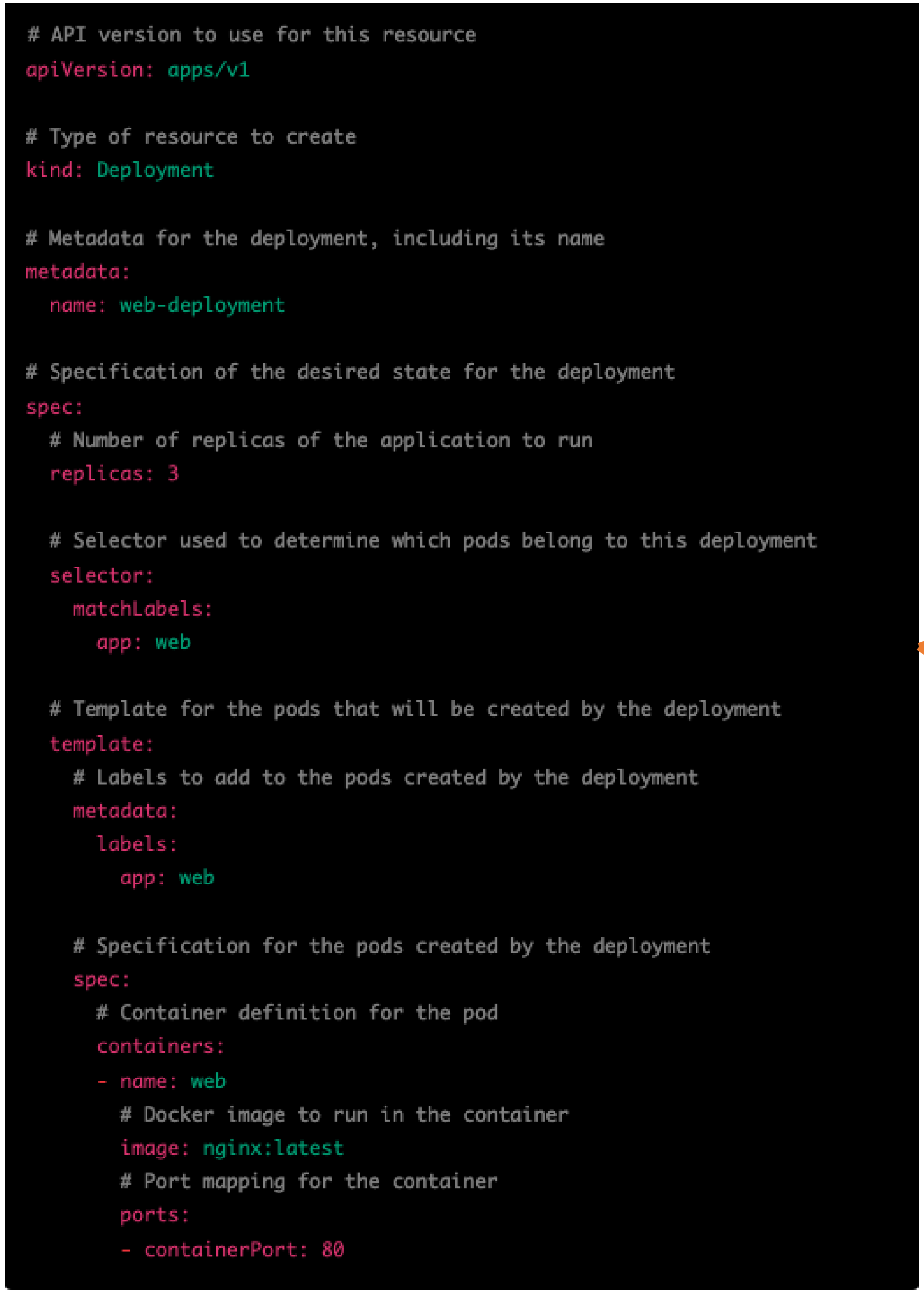
Pause and resume

- A deployment lets you temporarily stop and restart updates, providing greater control over the update process.

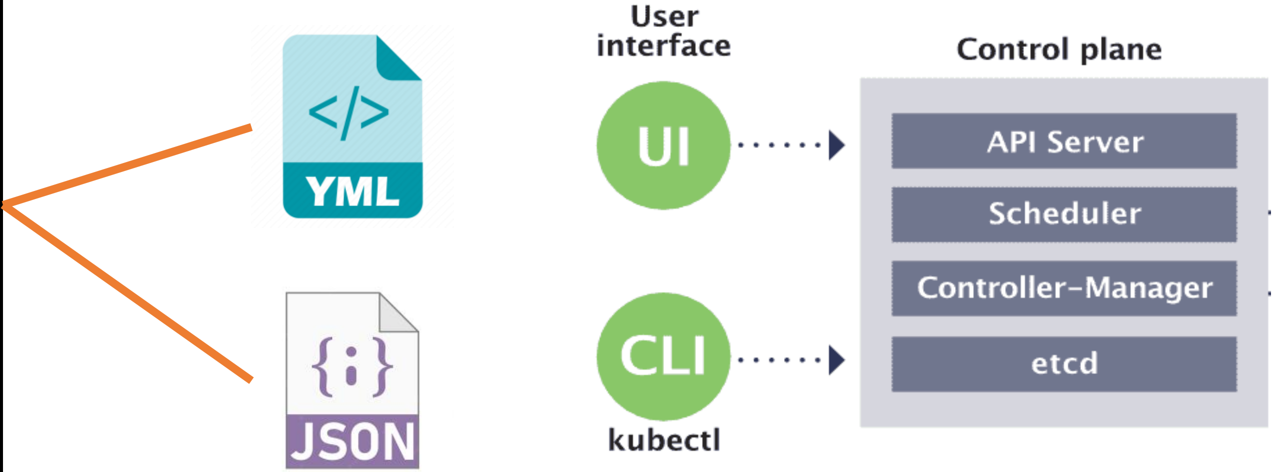
StatefulSet



Kubernetes Config File



```
# API version to use for this resource
apiVersion: apps/v1

# Type of resource to create
kind: Deployment

# Metadata for the deployment, including its name
metadata:
  name: web-deployment

# Specification of the desired state for the deployment
spec:
  # Number of replicas of the application to run
  replicas: 3

  # Selector used to determine which pods belong to this deployment
  selector:
    matchLabels:
      app: web

  # Template for the pods that will be created by the deployment
  template:
    # Labels to add to the pods created by the deployment
    metadata:
      labels:
        app: web

    # Specification for the pods created by the deployment
    spec:
      # Container definition for the pod
      containers:
      - name: web
        # Docker image to run in the container
        image: nginx:latest
        # Port mapping for the container
        ports:
        - containerPort: 80
```



Minikube

- Minikube allows you to run a **single-node Kubernetes cluster locally on your computer** inside a virtual machine.

- Supports various **operating systems** and **virtualization technologies.**

- Can be managed using the **minikube CLI** after installation.

- Used by developers, **testers**, and **administrators to try out Kubernetes** and test applications before deployment to a **production environment.**

Hands-on

Kubernetes