Basics of Kubernetes

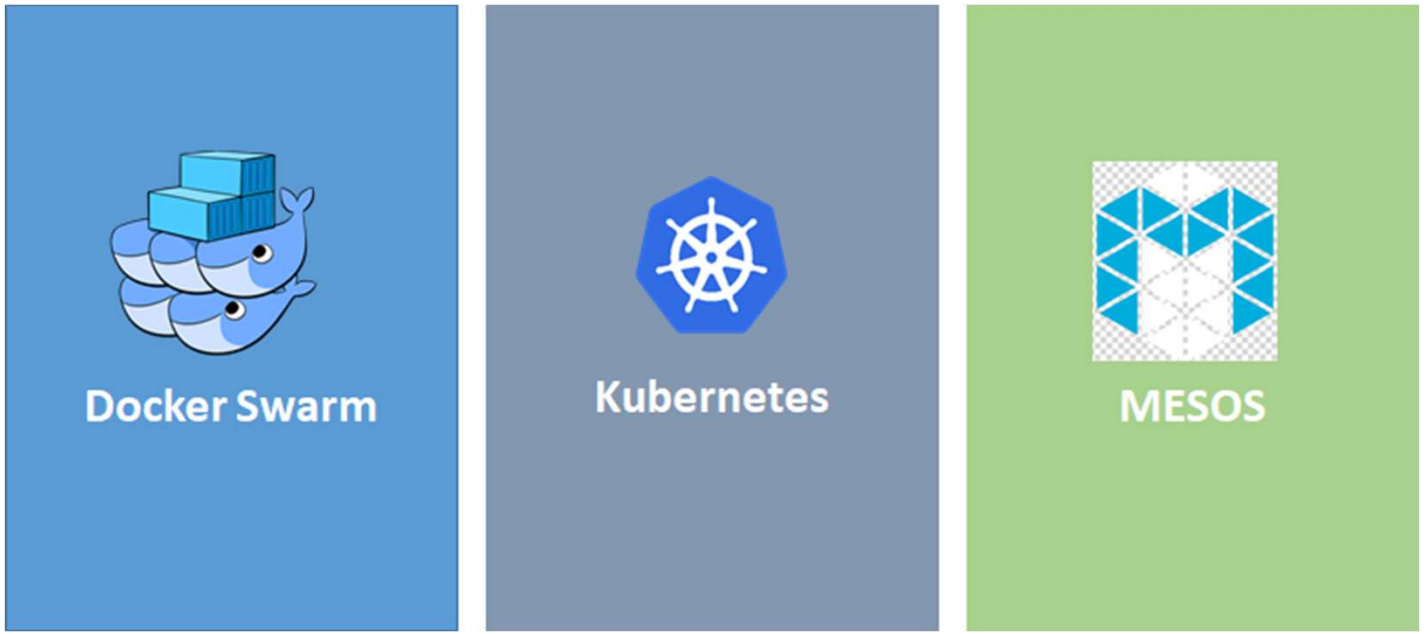
Agenda:

- Introduction
- Containers Orchestration Tools
- Kubernetes origin and definition
- Kubernetes Architecture
- Kubernetes Server Roles
- Kubernetes Cluster
- Kubernetes Components
- Kubernetes Basic Objects
- Benefits of Kubernetes
- Kubectl Definition and Basic Commands

Introduction

- **Immutable Infrastructure**
  - Never modifies directly.
  - For making any change to server:
    - Create copy of server from base image **->** changes made **->** replace old with new one
- **Containers**
  - Offers a way to package code, runtime, system tools, system libraries, and configs altogether.
  - Lightweight shipment, standalone executable.
  - Consistent application behaviour on multiple environments

Containers Orchestrators



Kubernetes

Origin:

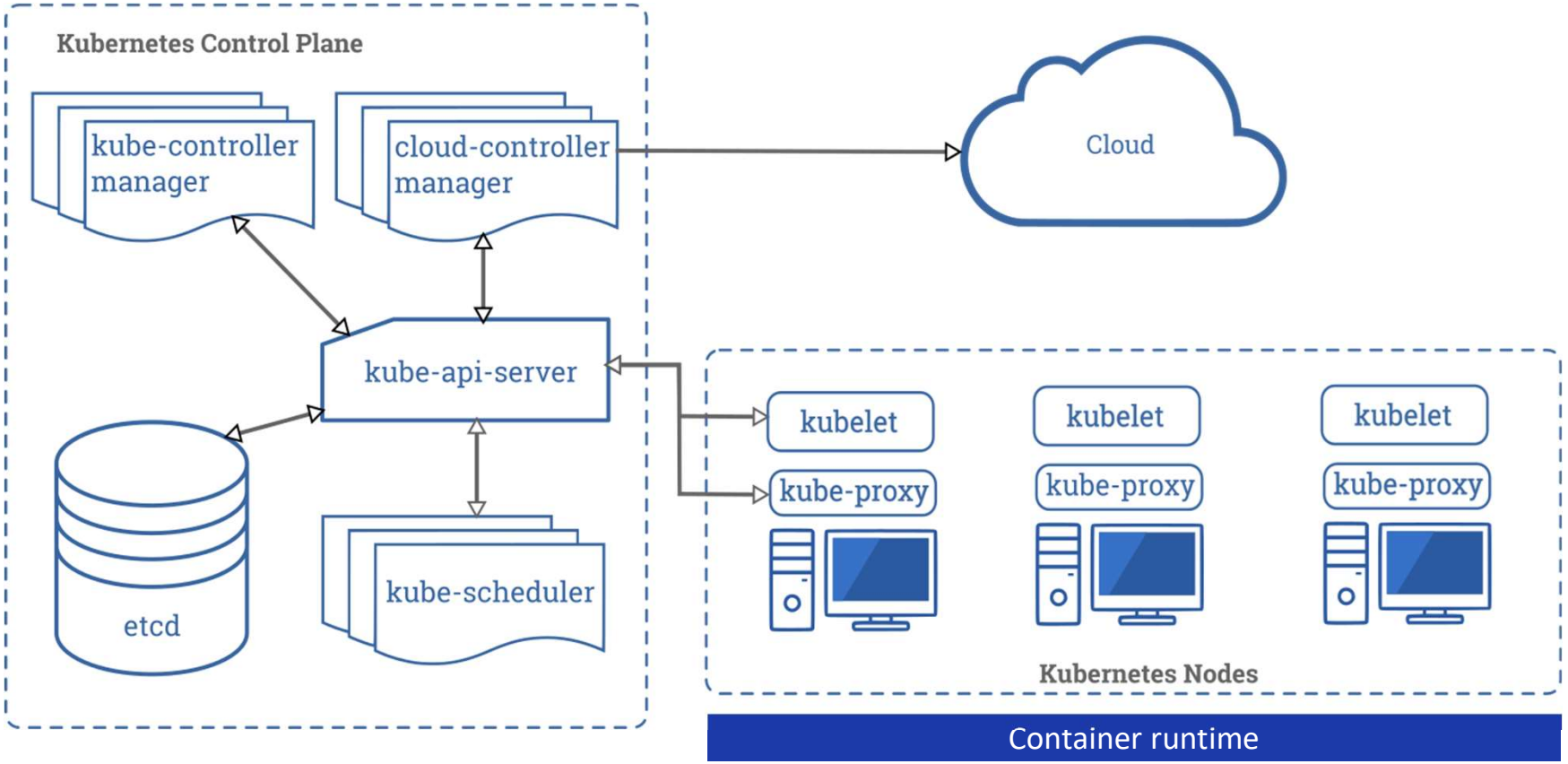
- Greek word - means **helmsman or pilot**
- Abbreviated as **k8s**
- Open sourced by Google in 2014

Definition:

"Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation"

https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

Kubernetes Architecture



Control Plane Components

The control plane's components make global decisions about the cluster (for example, scheduling), as well as detecting and responding to cluster events (for example, starting up a new pod when a deployment's replicas field is unsatisfied).

Control plane components can be run on any machine in the cluster. However, for simplicity, set up scripts typically start all control plane components on the same machine, and do not run user containers on this machine.

kube-controller-manager

Control plane component that runs controller processes.

Logically, each controller is a separate process, but to reduce complexity, they are all compiled into a single binary and run in a single process.

Some types of these controllers are:

Node controller: Responsible for noticing and responding when nodes go down.

Job controller: Watches for Job objects that represent one-off tasks, then creates Pods to run those tasks to completion.

EndpointSlice controller: Populates EndpointSlice objects (to provide a link between Services and Pods).

ServiceAccount controller: Create default ServiceAccounts for new namespaces.

Cloud-controller-manager

A Kubernetes control plane component that embeds cloud-specific control logic. The cloud controller manager lets you link your cluster into your cloud provider's API, and separates out the components that interact with that cloud platform from components that only interact with your cluster.

The following controllers can have cloud provider dependencies:

- Node controller: For checking the cloud provider to determine if a node has been deleted in the cloud after it stops responding
- Route controller: For setting up routes in the underlying cloud infrastructure
- Service controller: For creating, updating and deleting cloud provider load balancers

Kube-apiserver

- The API server is a component of the Kubernetes control plane that exposes the Kubernetes API. The API server is the front end for the Kubernetes control plane.
- The main implementation of a Kubernetes API server is kube-apiserver. kube-apiserver is designed to scale horizontally—that is, it scales by deploying more instances. You can run several instances of kube-apiserver and balance traffic between those instances.

Kube-scheduler

- Control plane component that watches for newly created Pods with no assigned node, and selects a node for them to run on.
- Factors taken into account for scheduling decisions include: individual and collective resource requirements, hardware/software/policy constraints, affinity and anti-affinity specifications, data locality, inter-workload interference, and deadlines.

etcd

- Consistent and highly-available key value store used as Kubernetes' backing store for all cluster data.
- If your Kubernetes cluster uses etcd as its backing store, make sure you have a back up plan for those data.

kubelet

- An agent that runs on each node in the cluster. It makes sure that containers are running in a Pod.
- The kubelet takes a set of PodSpecs that are provided through various mechanisms and ensures that the containers described in those PodSpecs are running and healthy. The kubelet doesn't manage containers which were not created by Kubernetes.

kube-proxy

- kube-proxy is a network proxy that runs on each node in your cluster, implementing part of the Kubernetes Service concept.
- kube-proxy maintains network rules on nodes. These network rules allow network communication to your Pods from network sessions inside or outside of your cluster.
- kube-proxy uses the operating system packet filtering layer if there is one and it's available. Otherwise, kube-proxy forwards the traffic itself.

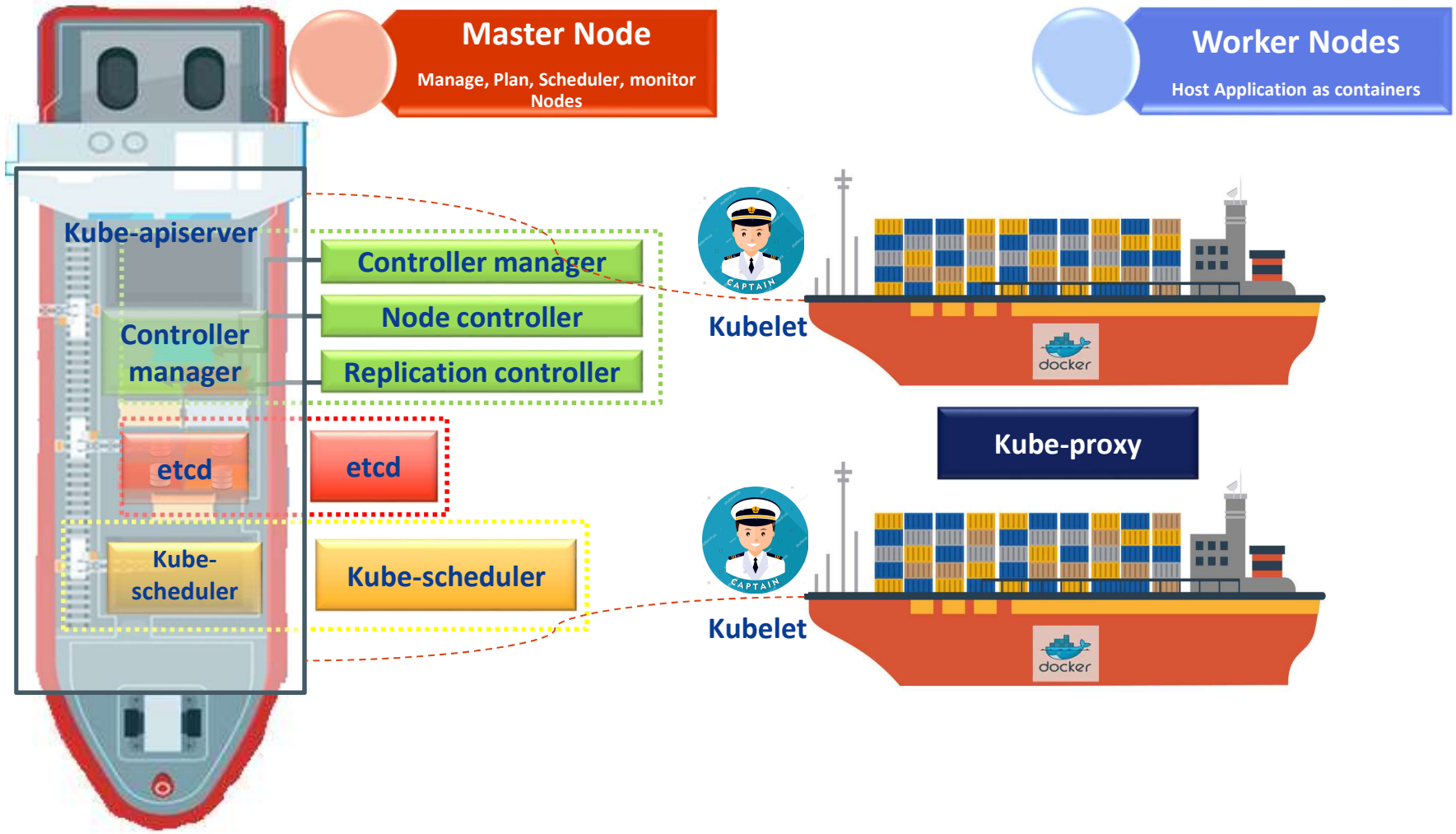
Container runtime

- The container runtime is the software that is responsible for running containers.
- Kubernetes supports container runtimes such as containerd, CRI-O, and any other implementation of the Kubernetes CRI (Container Runtime Interface).

Kubernetes Server Roles

- **Master**
  - Gateway and brain for the cluster
  - Exposes an API for users and clients
  - Responsible for health checking , scheduling
  - Orchestration communication between other components
- **Node**
  - Accepts and runs workloads
  - Receives work instructions from the master server and creates or destroys containers accordingly
  - Adjusts networking rules to route and forward traffic appropriately

Kubernetes Cluster


Master Node
Manage, Plan, Scheduler, monitor Nodes
Worker Nodes
Host Application as containers
Kube-apiserver
Controller manager
Controller manager
Node controller
Replication controller
etcd
etcd
Kube-scheduler
Kube-scheduler
Kubelet
Kube-proxy
Kubelet
CAPTAIN
CAPTAIN
docker
docker

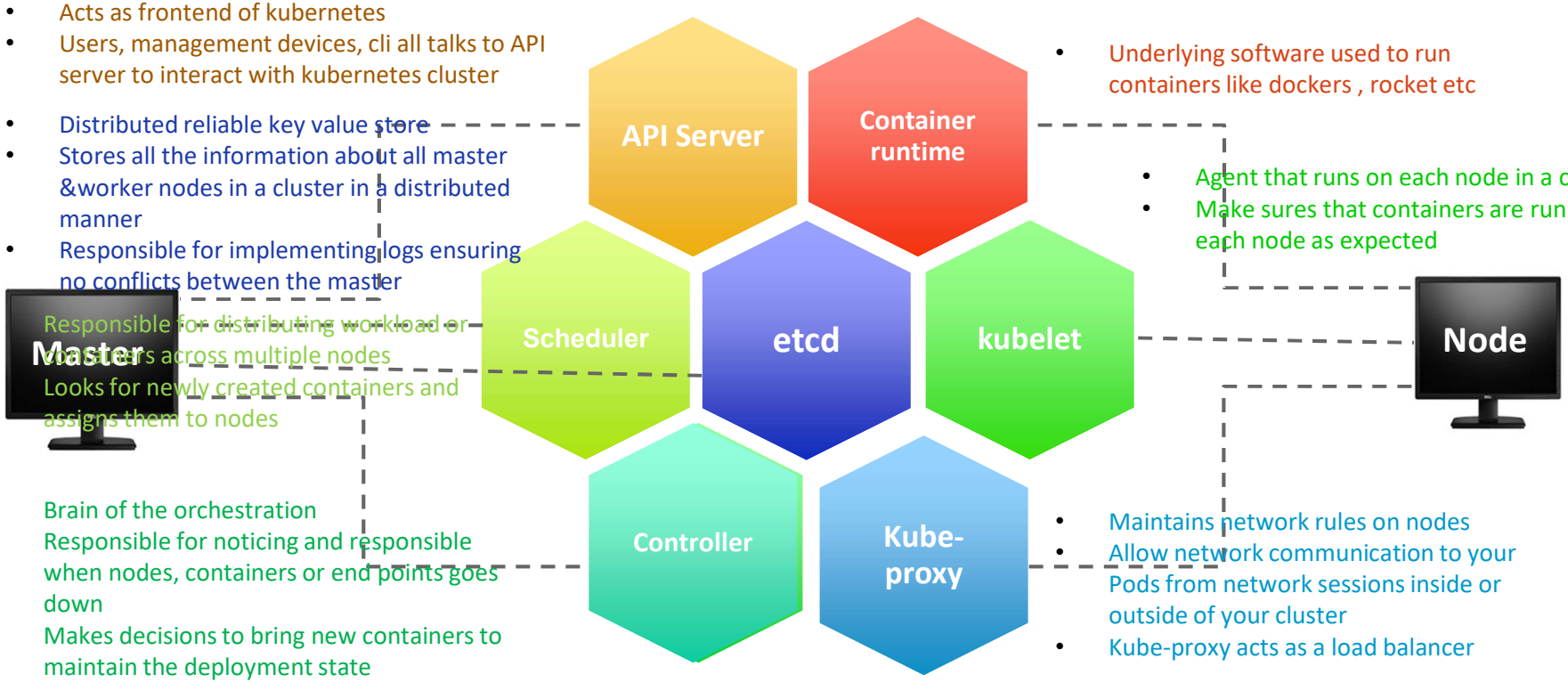
Kubernetes Components

- Acts as frontend of kubernetes
- Users, management devices, cli all talks to API server to interact with kubernetes cluster

- Distributed reliable key value store
- Stores all the information about all master &worker nodes in a cluster in a distributed manner
- Responsible for implementing logs ensuring no conflicts between the master

- Responsible for distributing workload or containers across multiple nodes
- Looks for newly created containers and assigns them to nodes

- Brain of the orchestration
- Responsible for noticing and responsible when nodes, containers or end points goes down
- Makes decisions to bring new containers to maintain the deployment state



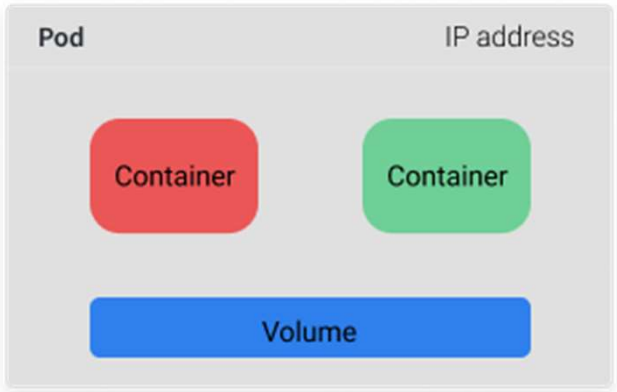
- Underlying software used to run containers like dockers , rocket etc

- Agent that runs on each node in a cluster
- Make sures that containers are running on each node as expected

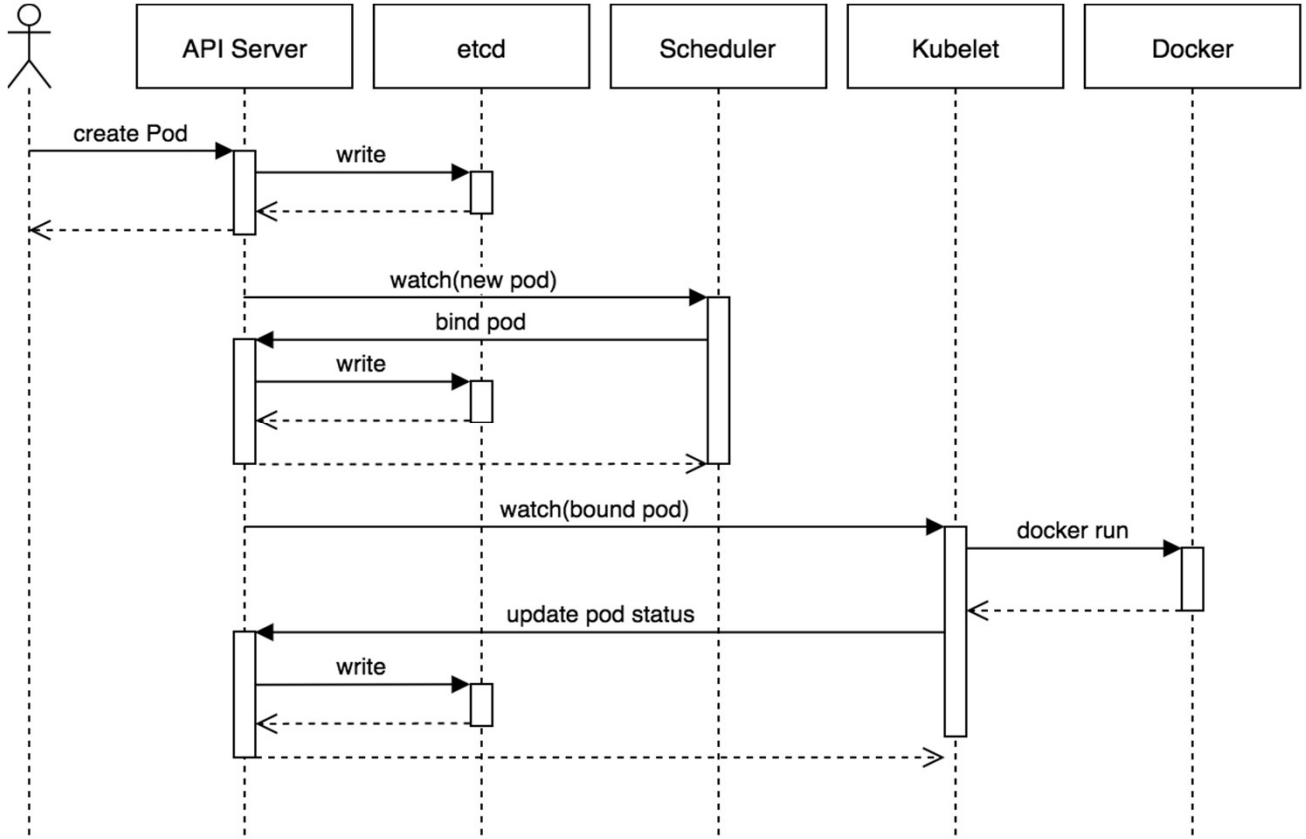
- Maintains network rules on nodes
- Allow network communication to your Pods from network sessions inside or outside of your cluster
- Kube-proxy acts as a load balancer

Kubernetes Basic Objects

- **Pod**
  - Fundamental building block in Kubernetes
  - Comprised of one or more (tightly related) containers, a shared networking layer, and shared file system volumes
  - Similar to containers, pods are designed to be ephemeral

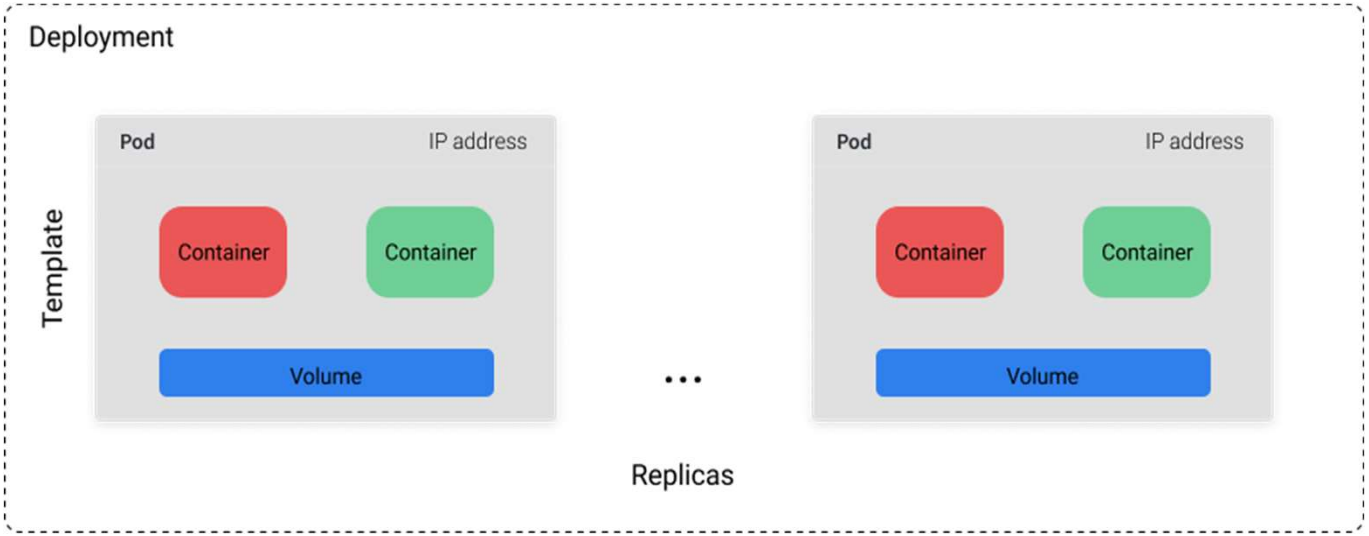


The following sequence diagram, from Heptio's blog, shows the components involved during a simple Pod creation process. It's a great illustration of the API Server and etcd interaction.



Deployment

- Contains a collection of pods defined by a template and a replica
- Best suited for stateless applications
- Allow us to specify the strategy of rolling updates when we have new versions of our container image



Deployment Example

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ml-model-serving
  labels:
    app: ml-model
spec:
  replicas: 10
  selector:
    matchLabels:
      app: ml-model
  template:
    metadata:
      labels:
        app: ml-model
    spec:
      containers:
      - name: ml-rest-server
        image: ml-serving:1.0
        ports:
        - containerPort: 80
```

How many Pods should be running?

How do we find Pods that belong to this Deployment?

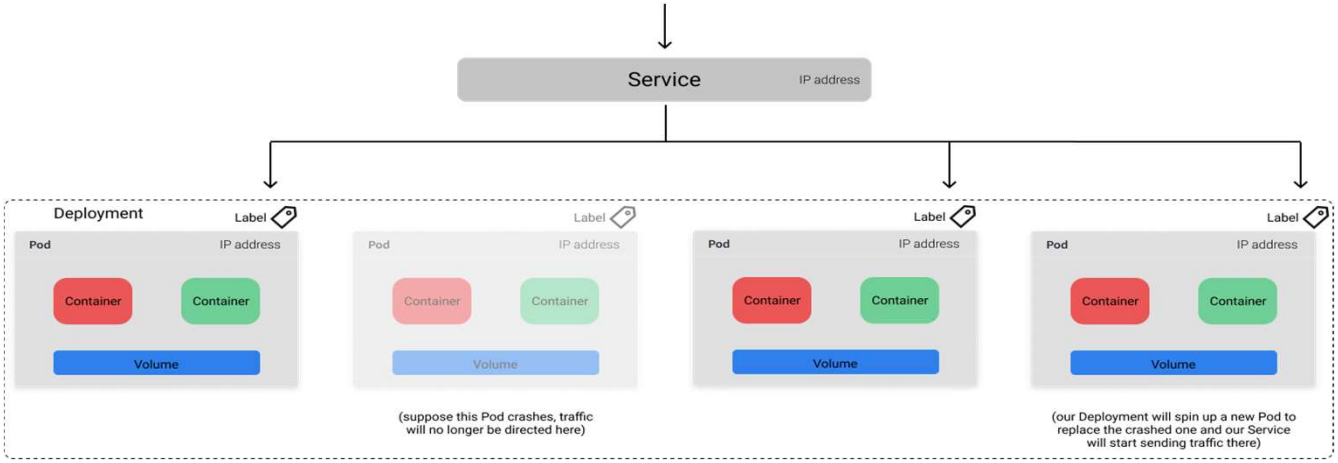
What should a Pod look like?

Add a label to the Pods so our Deployment can find the Pods to manage.

What containers should be running in the Pod?

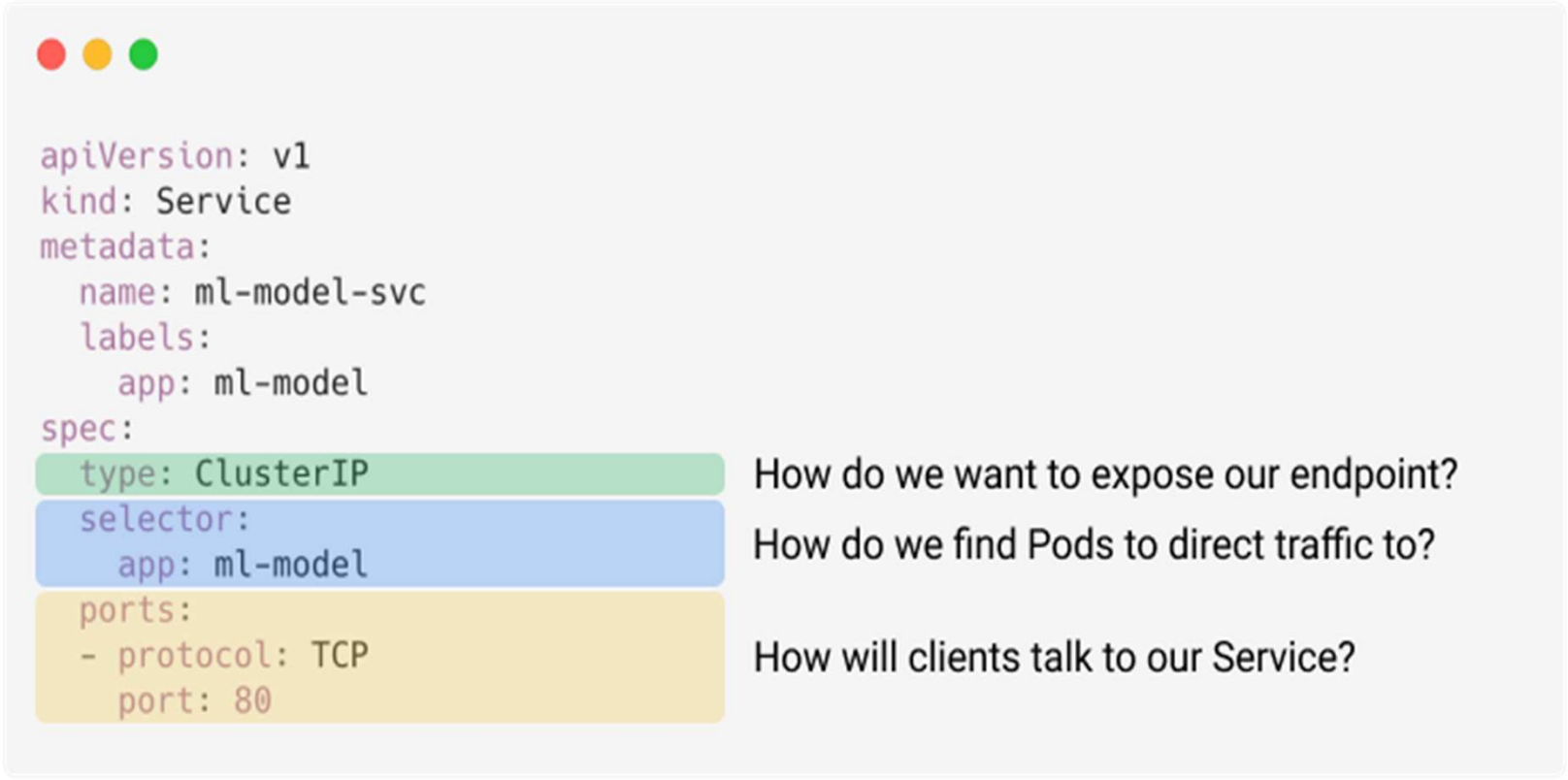
Service

- Provides with a stable endpoint which can be used to direct traffic to the desired Pods even as the exact underlying Pods change due to updates, scaling, and failures
- Services know which Pods they should send traffic to based on labels (key-value pairs) which is defined in the Pod metadata.



In this example, Service sends traffic to all healthy Pods with the label app="ml-model"

Service Example



Some other objects , include:

- **ReplicaSet (RS)**. Ensures the desired amount of pod is what's running.
- **StatefulSet.** A workload API object that manages stateful applications, such as databases.
- **DaemonSet.** Ensures that all or some worker nodes run a copy of a pod. This is useful for daemon applications like Fluentd.
- **Job.** Creates one or more pods, runs a certain task(s) to completion, then deletes the pod(s).
- **Volume.** An abstraction that lets us persist data. (This is necessary because containers are ephemeral—meaning data is deleted when the container is deleted.)
- **Namespace.** A segment of the cluster dedicated to a certain purpose, for example a certain project or team of devs.

Benefits of Kubernetes

- **Horizontal scaling**. Scale your application as needed from command line or UI.
- **Automated rollouts and rollbacks.** Roll out changes that monitor the health of your application—ensuring all instances don't fail or go down simultaneously. If something goes wrong, K8S automatically rolls back the change.
- **Service discovery and load balancing.** Containers get their own IP so you can put a set of containers behind a single DNS name for load balancing.
- **Storage orchestration.** Automatically mount local or public cloud or a network storage.
- **Secret and configuration management.** Create and update secrets and configs without rebuilding your image.
- **Self-healing.** The platform heals many problems: restarting failed containers, replacing and rescheduling containers as nodes die, killing containers that don't respond to your user-defined health check, and waiting to advertise containers to clients until they're ready.
- **Automatic binpacking.** Automatically schedules containers based on resource requirements and other constraints.

Kubectl

Kube control tool/Kube command line tool

- Kubernetes CLI tool
- Used to deploy and manage applications on a kubernetes cluster
- Gets cluster related information
- Gets status of the nodes in the cluster and many others

Kubectl Basic Commands

- kubectl get - list resources
- kubectl describe - show detailed information about a resource
- kubectl logs - print the logs from a container in a pod
- kubectl exec - execute a command on a container in a pod
- Kubectl run hello-minikube - Deploys application in a cluster
- Kubectl cluster-info - View information about the cluster
- Kubectl get nodes - View all the nodes part of a cluster
- Kubectl get deployments - View information about the deployments

Thank you