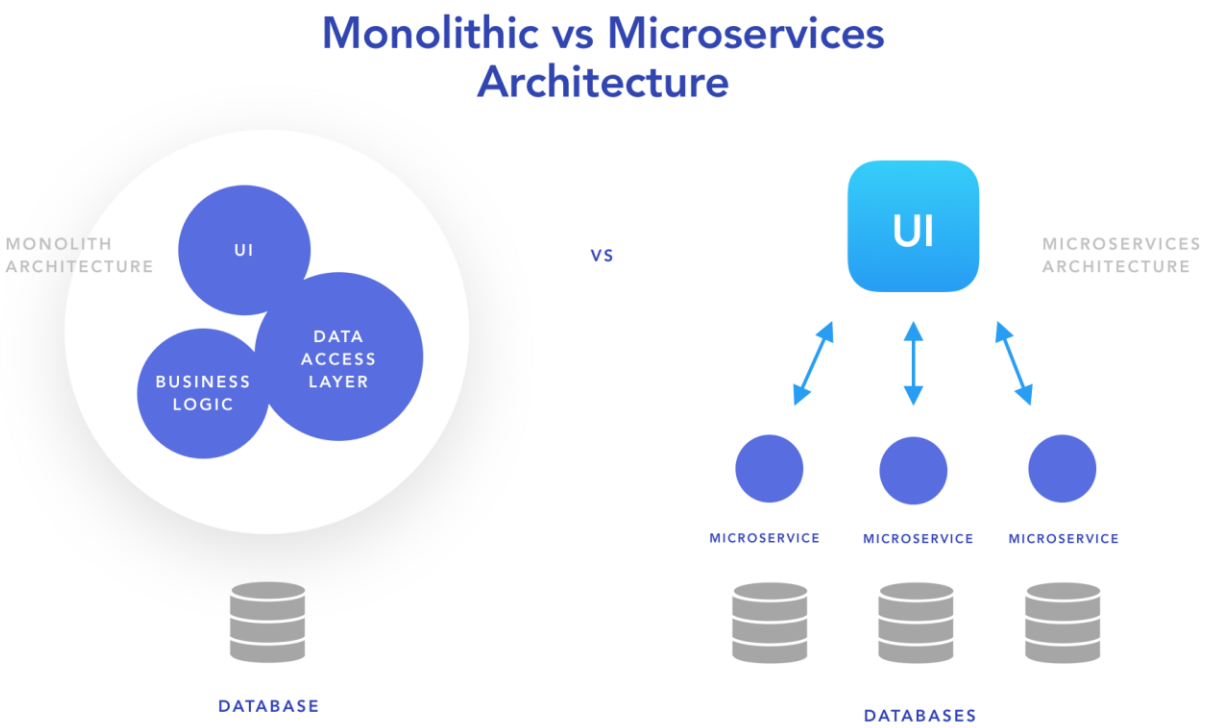
Kubernetes Essential

Agenda

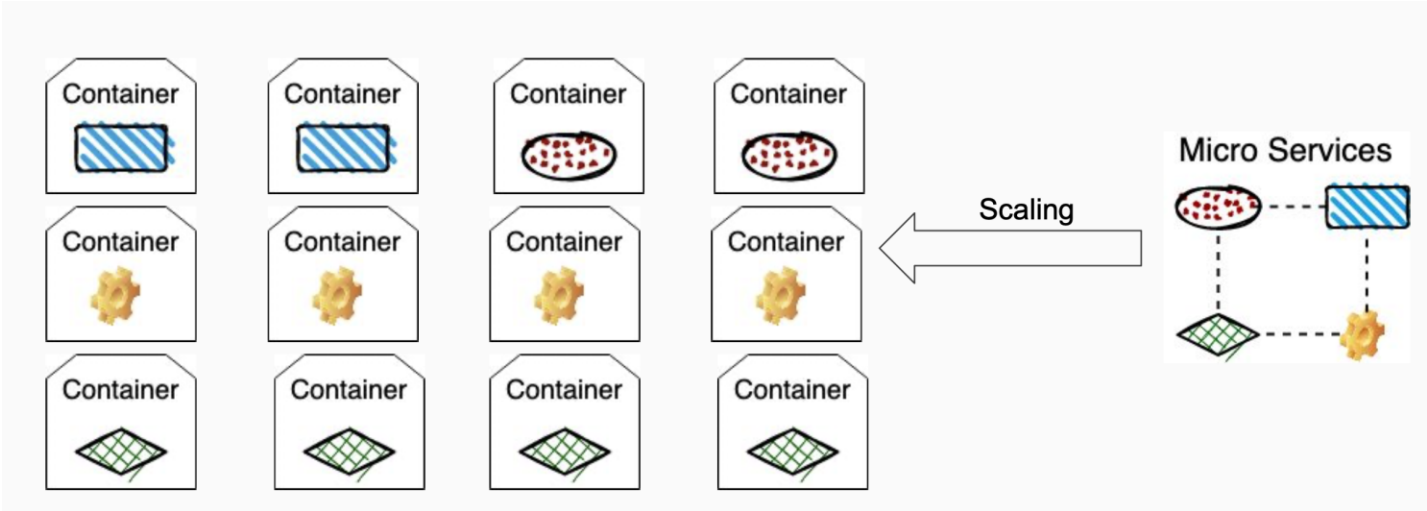
- Assignment Review & Guides
- Trend from Monolithic to Microservices
- What is Kubernetes?
- Kubernetes architecture

Trend from Monolithic to Microservices



Benefits of microservices

- Microservices are a type of application architecture that involves splitting the application into a series of small, independent services.
- Microservices can be built, modified, and scaled separately, with relatively little impact on one another.



Linux Containers and Microservices

Linux Containers excel when it comes to managing a large number of small, independent workloads.

Containers and orchestration make it easier to manage and automate the process of deploying, scaling, and connecting lots of microservice instances.

For example, I may have one microservice that needs additional resources.

With containers, all I need to do is create more containers for that service to handle the load.

With orchestration, that can even be done automatically and in real-time!

Kubernetes Overview

What is Orchestration?

- Container Orchestration simply refers to processes used to manage containers and to automate the management of containers.
- For example:
  - I want to start up a set of five containers in production.
  - I could spin up each container manually.
  - Or, I could tell an orchestration tool like Kubernetes that I want five containers, and let the tool do it.

Kubernetes - k8s

- Kubernetes is a container orchestration tool.
- Open-source container orchestration tool
- Developed by Google
- Manage containerize applications on different environments
  - Physical Machines
  - Virtual Machines
  - Cloud Environment
  - Hybrid Environment

- Check out the official Kubernetes site for documentation and additional info!

https://kubernetes.io
https://github.com/kubernetes/kubernetes

What features does Kubernetes provide ?

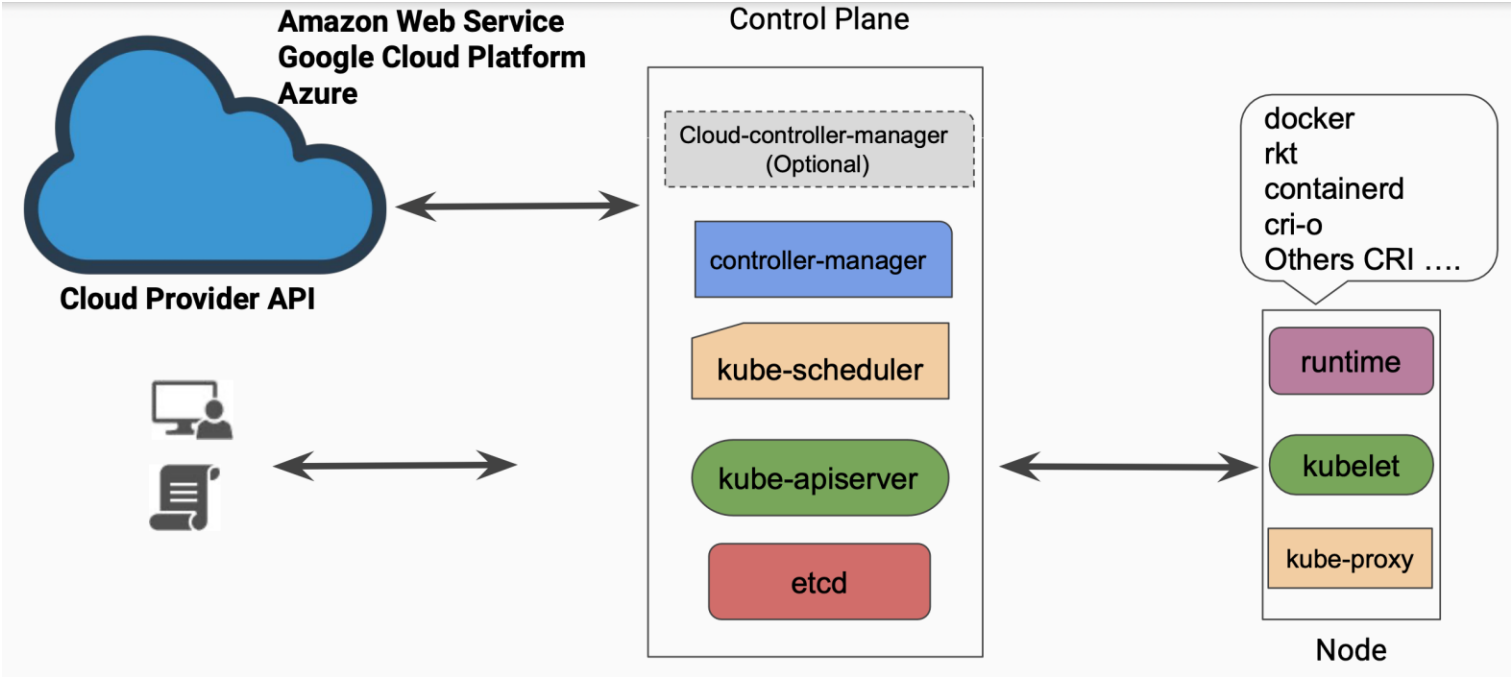
- High Availability
- Scalability
- Self-healing
- Service Discovery
- Load Balancing
- Rollout & Rollback
- Extensibility

Cloud Orchestration Solutions

Cloud providers such as Amazon Web Services, Microsoft Azure, and Google Cloud the platform also offers built-in container orchestration solutions, including cloud-native Kubernetes implementations!

- Amazon ECS for Kubernetes - EKS
- Azure Kubernetes Service - AKS
- Google Kubernetes Engine - GKE
- DigitalOcean Kubernetes - DOKS
- IBM Cloud Kubernetes Service

Cluster Architecture



Master Components

The Kubernetes Master is a collection of the following processes that run on a single node in your cluster, which is designated as the master node.

- Kube-apiserver
- Etcd
- Kube-controller-manager
- Cloud-controller-manager
- Kube-scheduler

Node Components

Node components run on every node, maintaining running pods and providing the Kubernetes runtime environment.

- Kubelet

  An agent that runs on each node in the cluster. It makes sure that containers are running in a Pod

- Kube-proxy

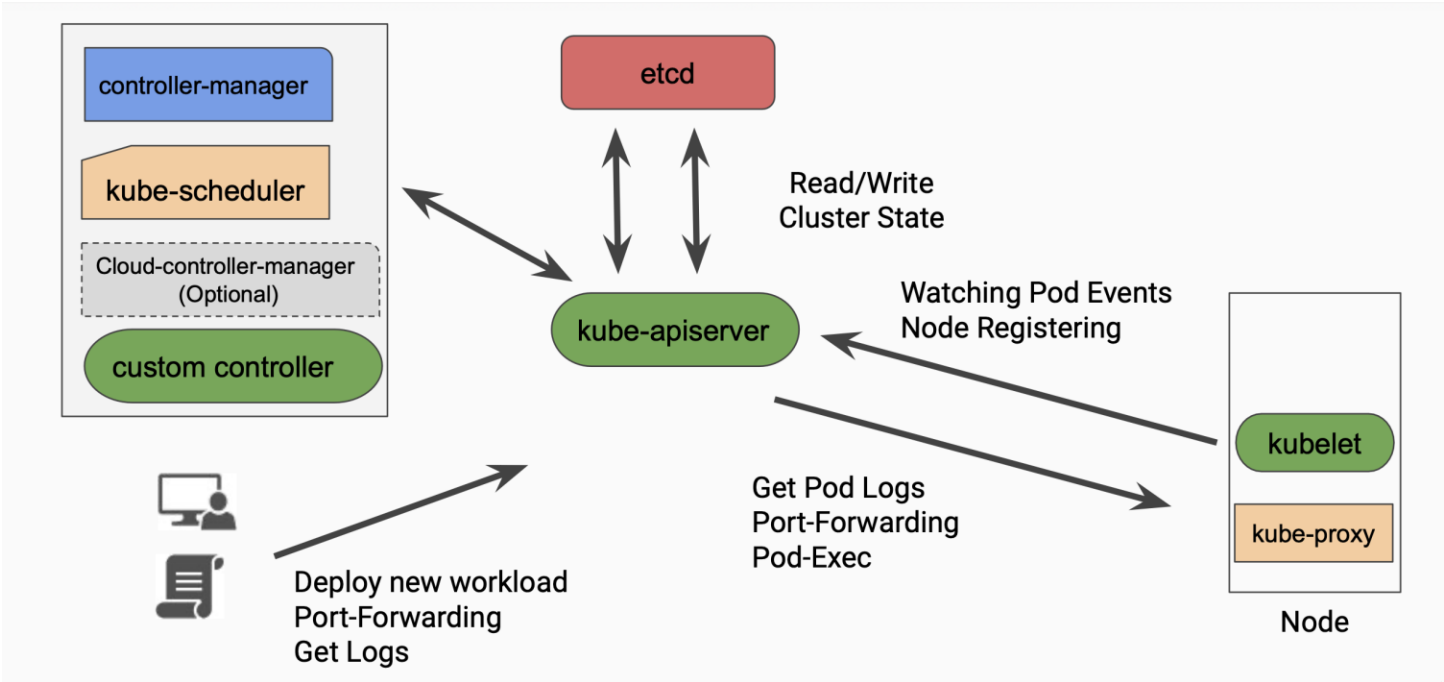
  A network proxy that runs on each node in your cluster, implementing part of the Kubernetes Service

- Container runtime engine

KUBE-APISERVER

- Provides a forward facing REST interface into the kubernetes control plane and datastore.
- All clients and other applications interact with kubernetes **strictly** through the API Server.
- Acts as the gatekeeper to the cluster by handling authentication and authorization, request validation, mutation, and admission control in addition to being the front-end to the backing datastore.

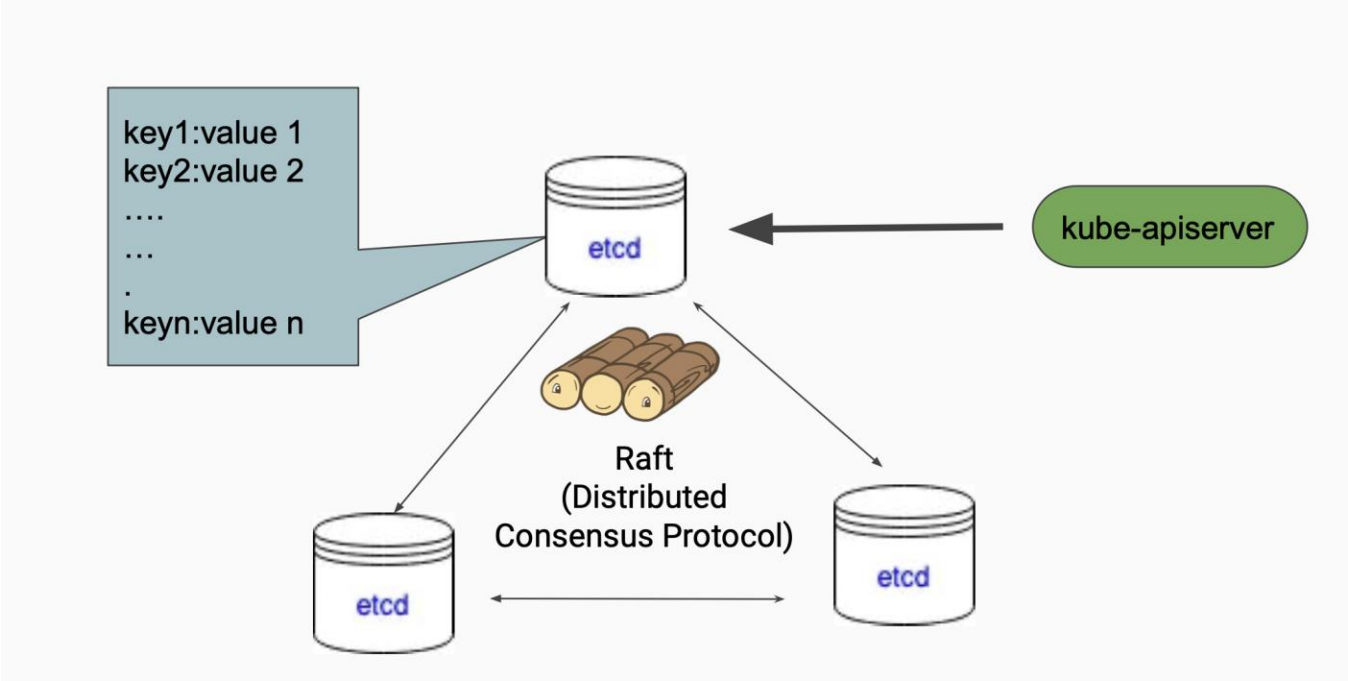
KUBE-APISERVER



ETCD

- etcd acts as the cluster datastore.
- Purpose in relation to Kubernetes is to provide a strong, consistent and highly available key-value store for persisting cluster state.
- Stores objects and config information.
- Uses "Raft Consensus" among a quorum of systems to create a fault-tolerant consistent "view" of the cluster.

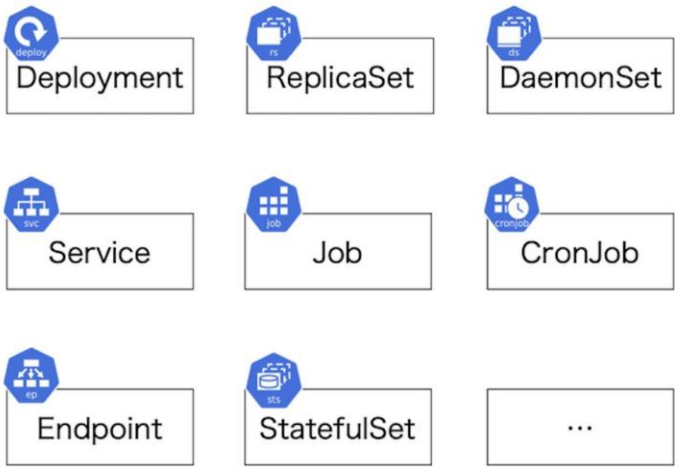
ETCD


key1:value 1
key2:value 2
….
…
.
keyn:value n
etcd
kube-apiserver
Raft
(Distributed
Consensus Protocol)
etcd
etcd

master

KUBE-CONTROLLER-MANAGER

- Serves as the primary daemon that manages all core component control loops.
- Monitors the cluster state via the apiserver and steers the cluster towards the desired state.



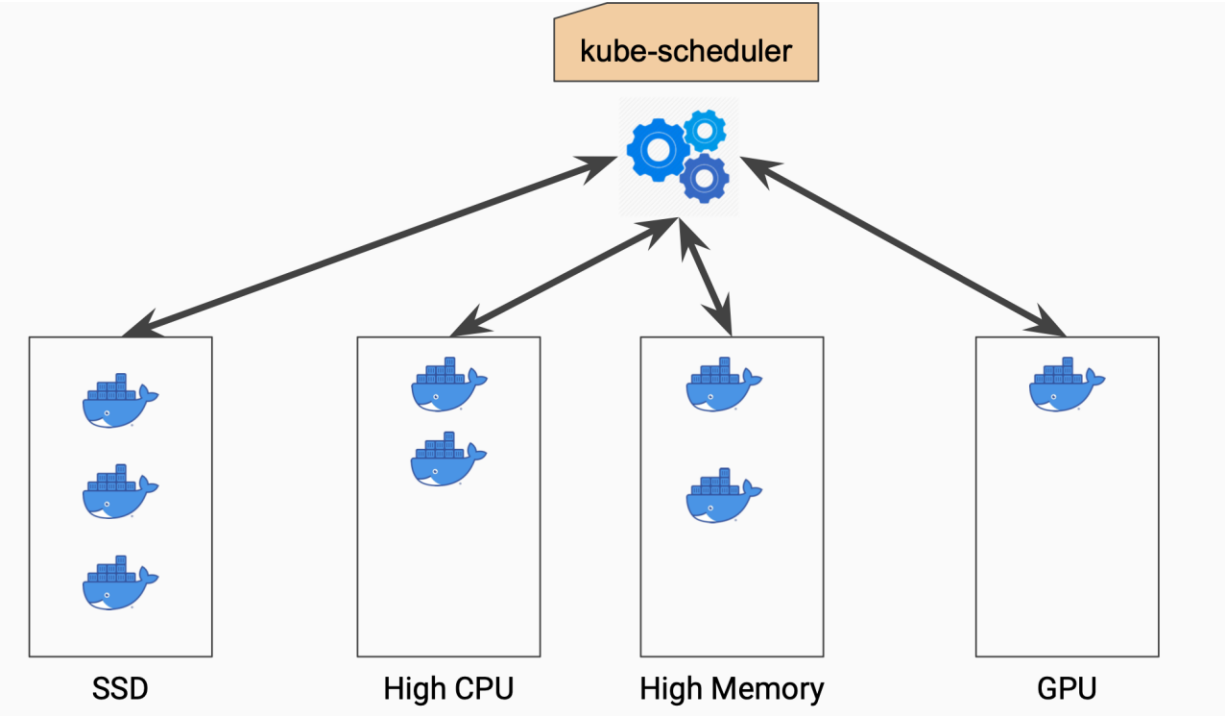
master

KUBE-SCHEDULER

- Component on the master that watches newly created pods that have no node assigned, 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.

KUBE-SCHEDULER



master

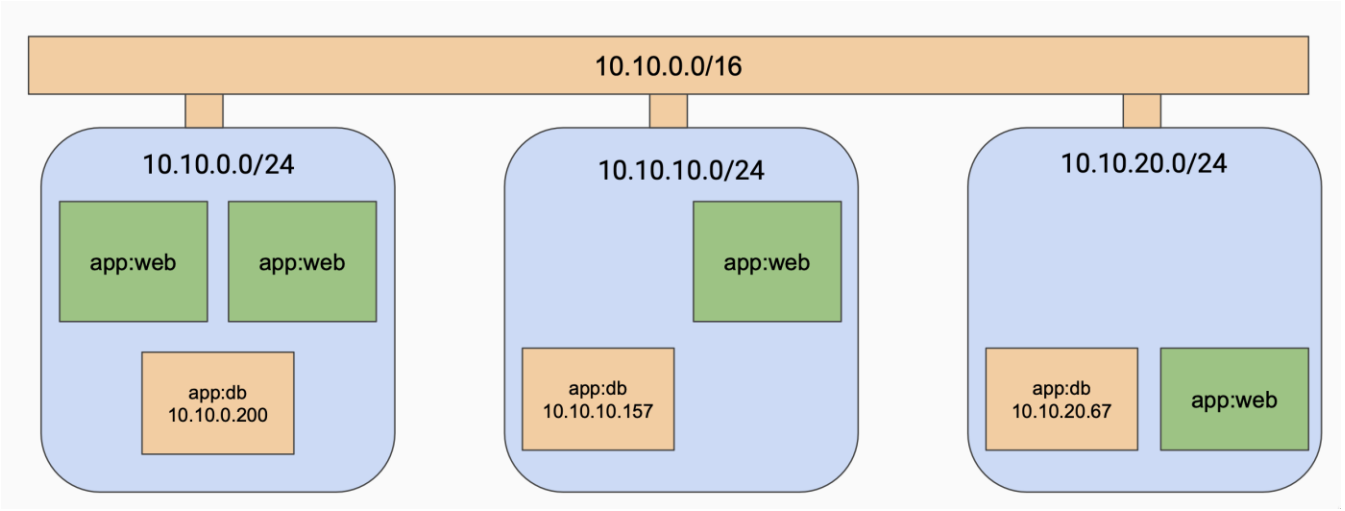
CLOUD-CONTROLLER-MANAGER

- Daemon that provides cloud-provider specific knowledge and integration capability into the core control loop of Kubernetes.

- The controllers include Node, Route, Service, and add an additional controller to handle things such as **PersistentVolume** Labels.

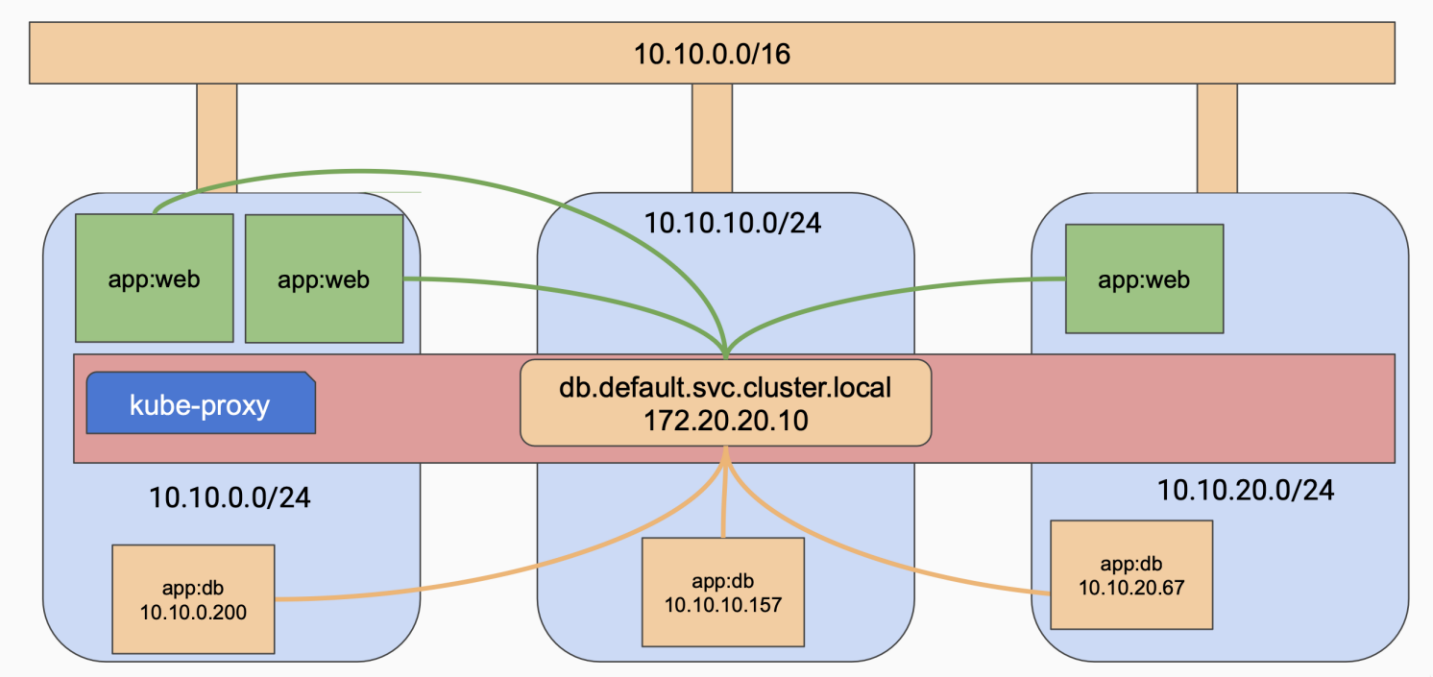
KUBE-PROXY

- How **web service** connect to **database service**?
- How to keep track of **database service** ip addresses in case of ip changing?
- How to do load-balance between many service instances?



worker

KUBE-PROXY



worker

KUBELET

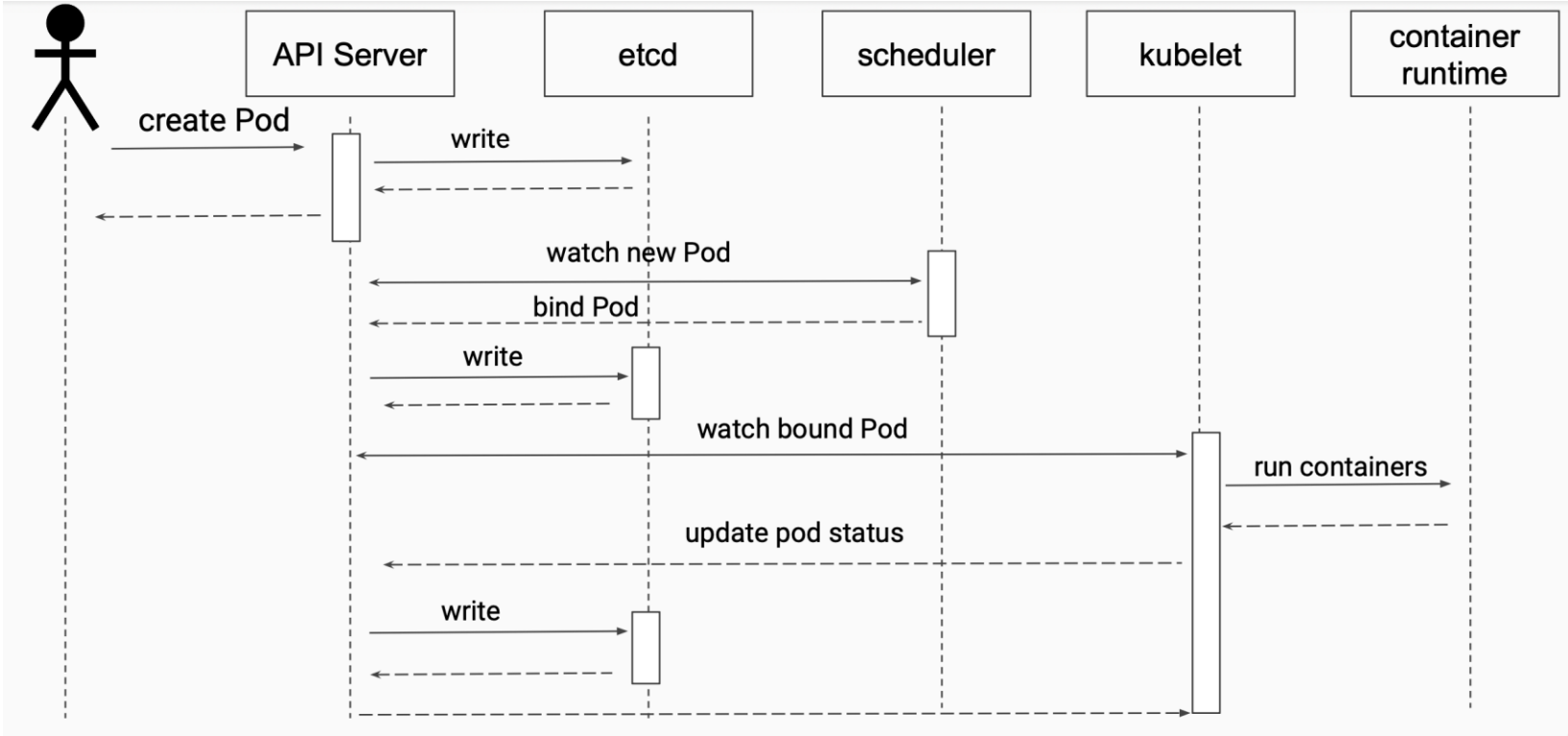
- An agent that runs on each node in the cluster. It makes sure that containers are running in a pod.

- 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.

CONTAINER RUNTIME ENGINE

- Docker is not the only option for doing containers!
- rkt - Created by CoreOS, “designed with composability and security in mind.”
- Containerd - Emphasizes “simplicity, robustness, and portability.”
- LXC/LXD

Request Flow



Q&A