Kubernetes

An Introduction

Pre-Requisites

Docker

YAML

Virtual Box to Setting up Lab environment,AWS

Linux

$ whoami - Aneesh

Aneesh Ansari

er.aneeshansari@gmail.com

Sr. Devops Eng. @ MNC

Part time Freelancer

Trainer in following technology

Linux, Docker, Kubernetes, DevOps

Overview

What Does “Kubernetes” Mean?

Greek for “pilot” or
“Helmsman of a ship”

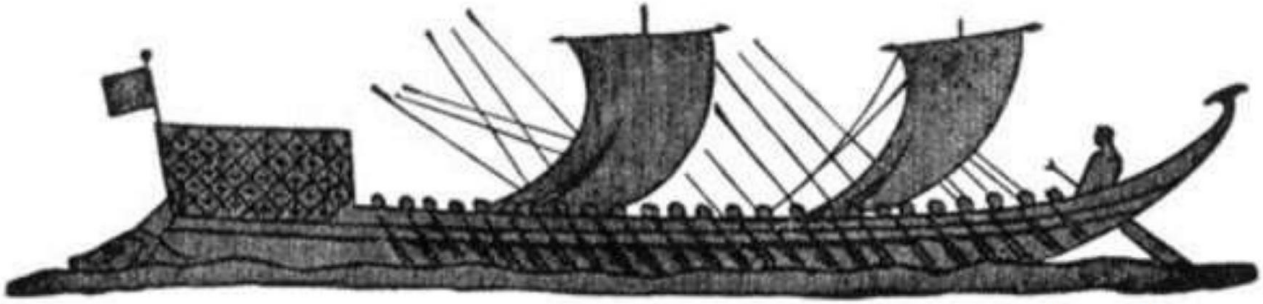


Image Source

What is Kubernetes?

- Kubernetes is an open-source Container Management tool which automates container deployment, container (de)scaling & container load balancing.
- Written on Golang, it has a huge community because it was first developed by Google & later donated to **CNCF**
- Can group ‘n’ no of containers into one logical unit for managing & deploying them
- Built from the lessons learned in the experiences of developing and running Google’s Borg and Omega.

Decouples Infrastructure and Scaling

- **All services** within Kubernetes are natively Load Balanced.
- Can scale up and down dynamically.
- Used both to enable self-healing and seamless upgrading or rollback of applications.

Self Healing

Kubernetes will **ALWAYS** try and steer the cluster to its desired state.

- **Me:** “I want 3 healthy instances of redis to always be running.”
- **Kubernetes:** “Okay, I’ll ensure there are always 3 instances up and running.”
- **Kubernetes:** “Oh look, one has died. I’m going to attempt to spin up a new one.”

Most Importantly...

Use the **SAME** API
across bare metal and
EVERY cloud provider!!!

Who “Manages” Kubernetes?

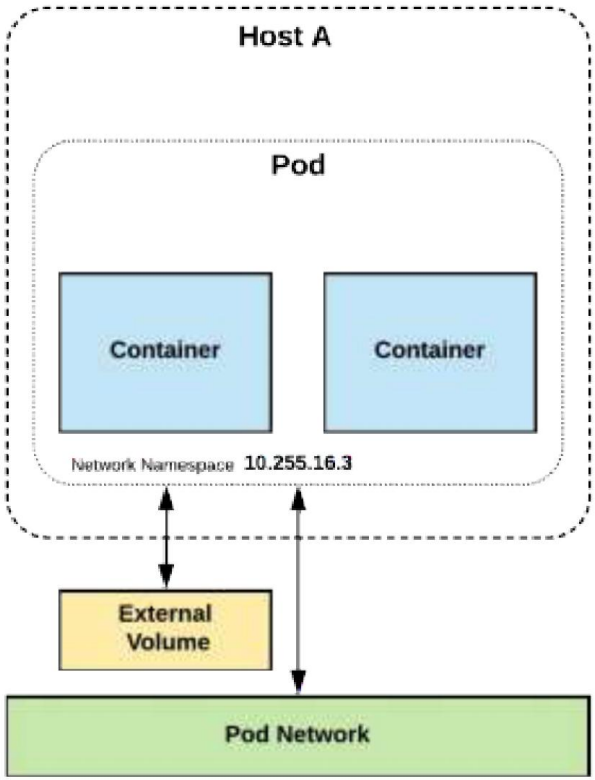
CLOUD NATIVE
COMPUTING FOUNDATION

The CNCF is a child entity of the Linux Foundation and operates as a vendor neutral governance group.

A Couple Key Concepts...

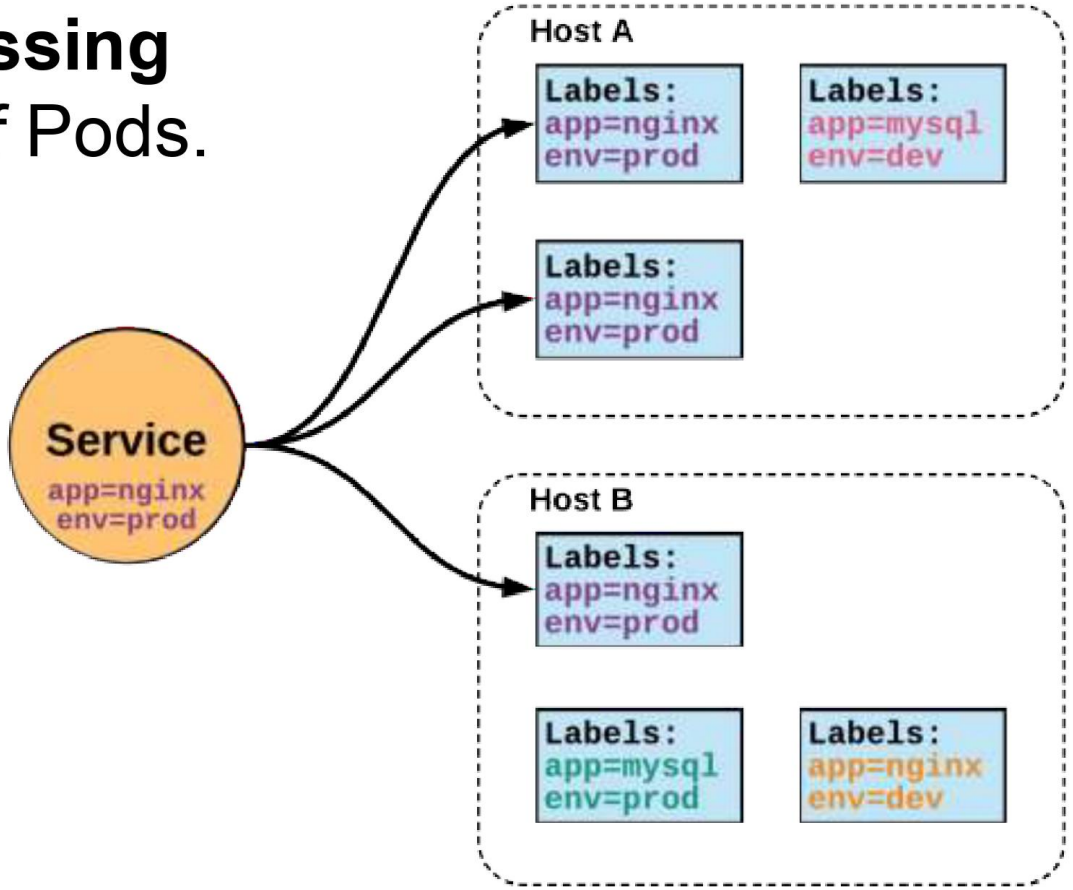
Pods

- **Atomic unit** or smallest "*unit of work*"of Kubernetes.
- Pods are **one or MORE containers** that share volumes, a network namespace, and are a part of a **single context**.

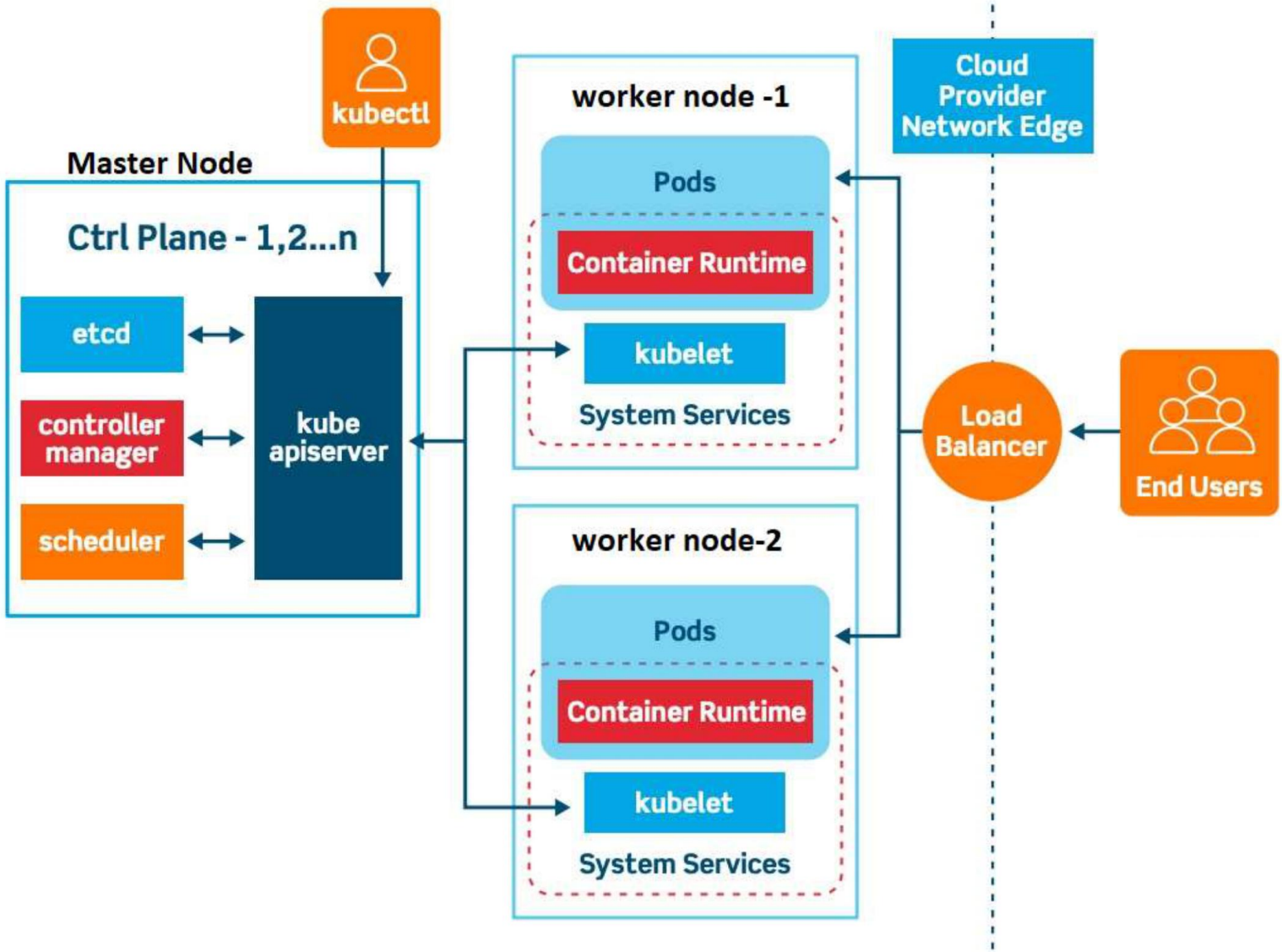


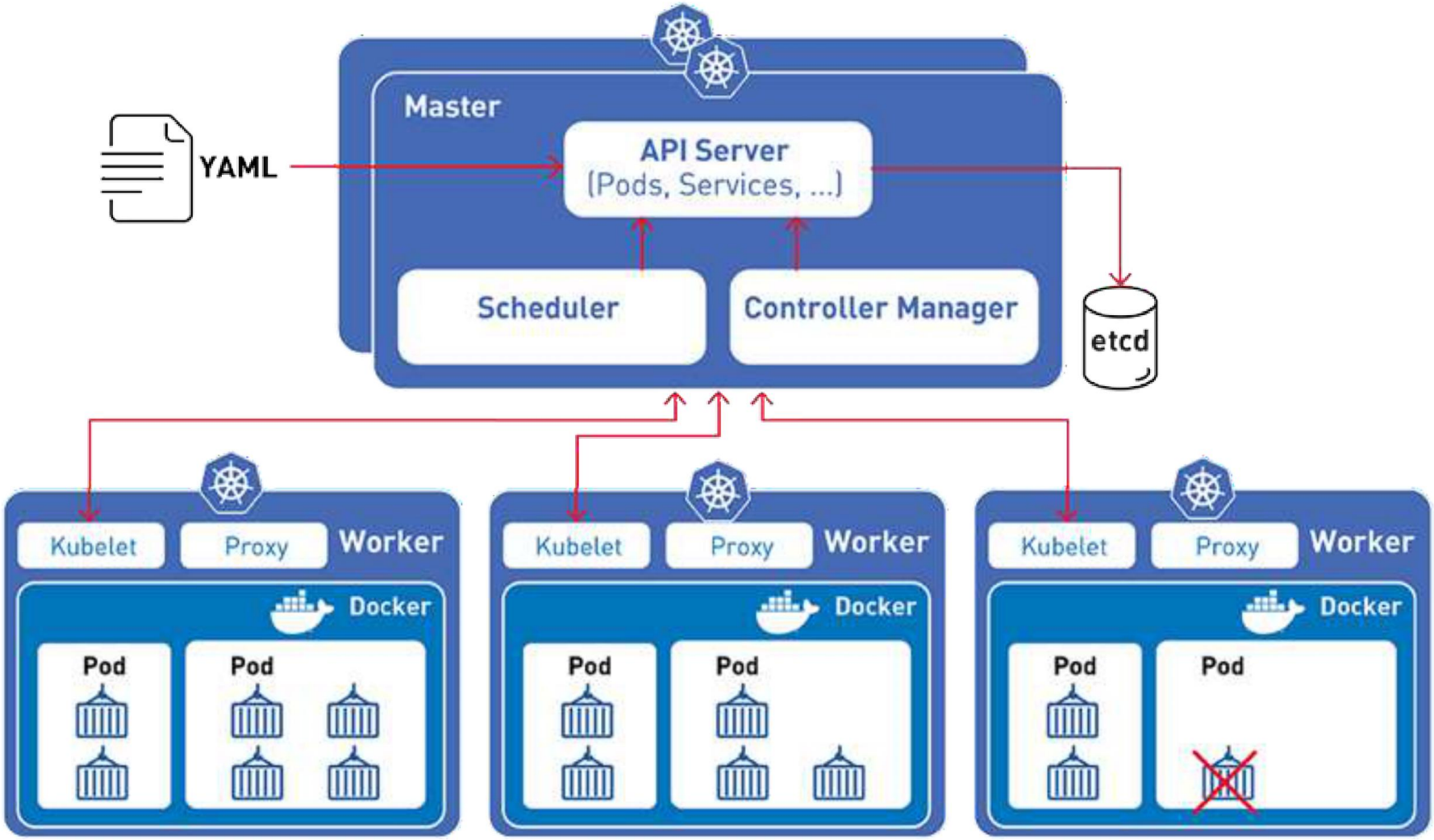
Services

- **Unified method of accessing** the exposed workloads of Pods.
- **Durable resource**
  - static cluster IP
  - static namespaced DNS name



Architecture Overview


kubectl
Master Node
Ctrl Plane - 1,2...n
etcd
controller manager
scheduler
kube apiserver
worker node -1
Pods
Container Runtime
kubelet
System Services
worker node-2
Pods
Container Runtime
kubelet
System Services
Cloud Provider Network Edge
Load Balancer
End Users

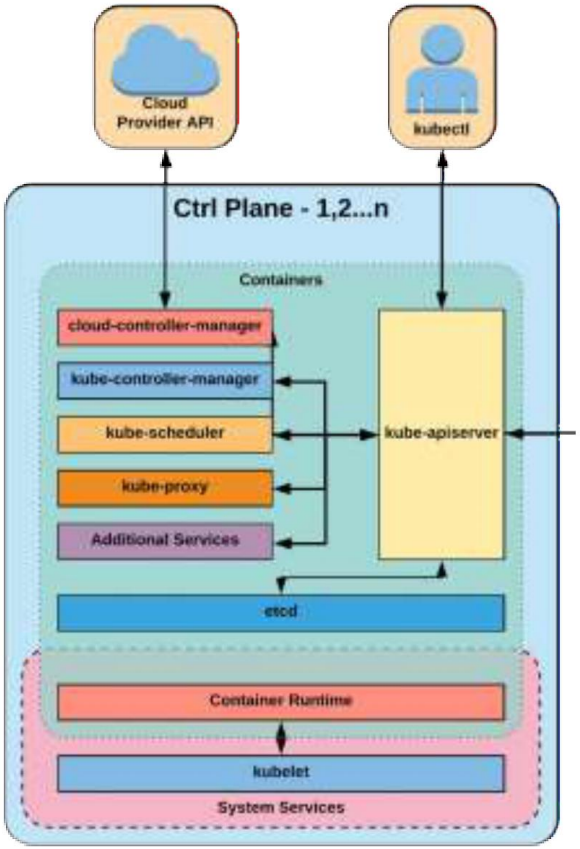

YAML
Master
API Server
(Pods, Services, ...)
Scheduler
Controller Manager
etcd
Kubelet
Proxy
Worker
Docker
Pod
Pod
Kubelet
Proxy
Worker
Docker
Pod
Pod
Kubelet
Proxy
Worker
Docker
Pod
Pod

Control Plane Components

Architecture Overview

Control Plane Components

- kube-apiserver
- etcd
- kube-controller-manager
- kube-scheduler



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

1. Authenticate User
2. Validate Request
3. Retrieve data
4. Update ETCD
5. Scheduler
6. Kubelet

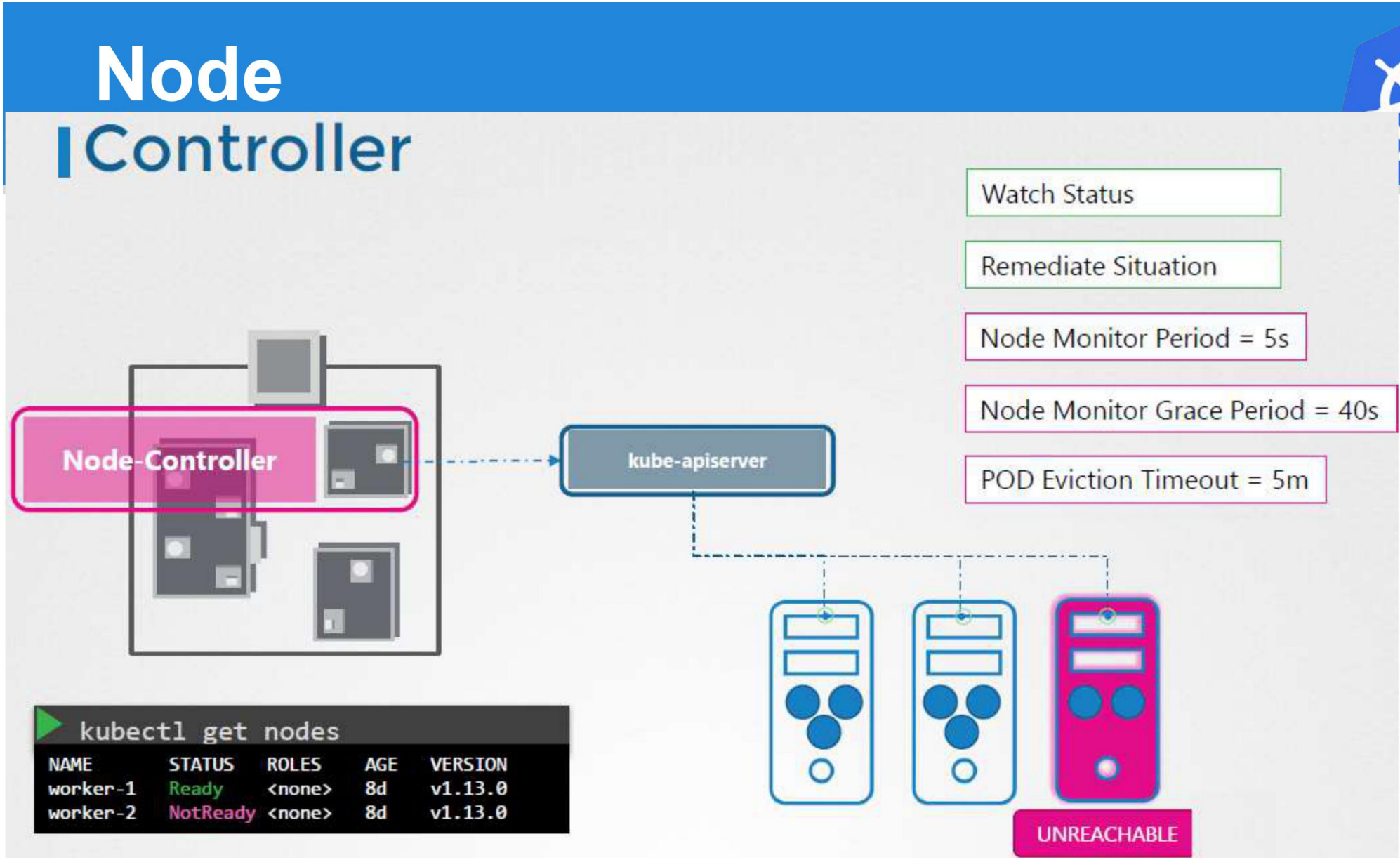
etcd

- 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.
- If you setup cluster via kubeadm , than kubeadm will set a etcd in pod in kube-system namespace.

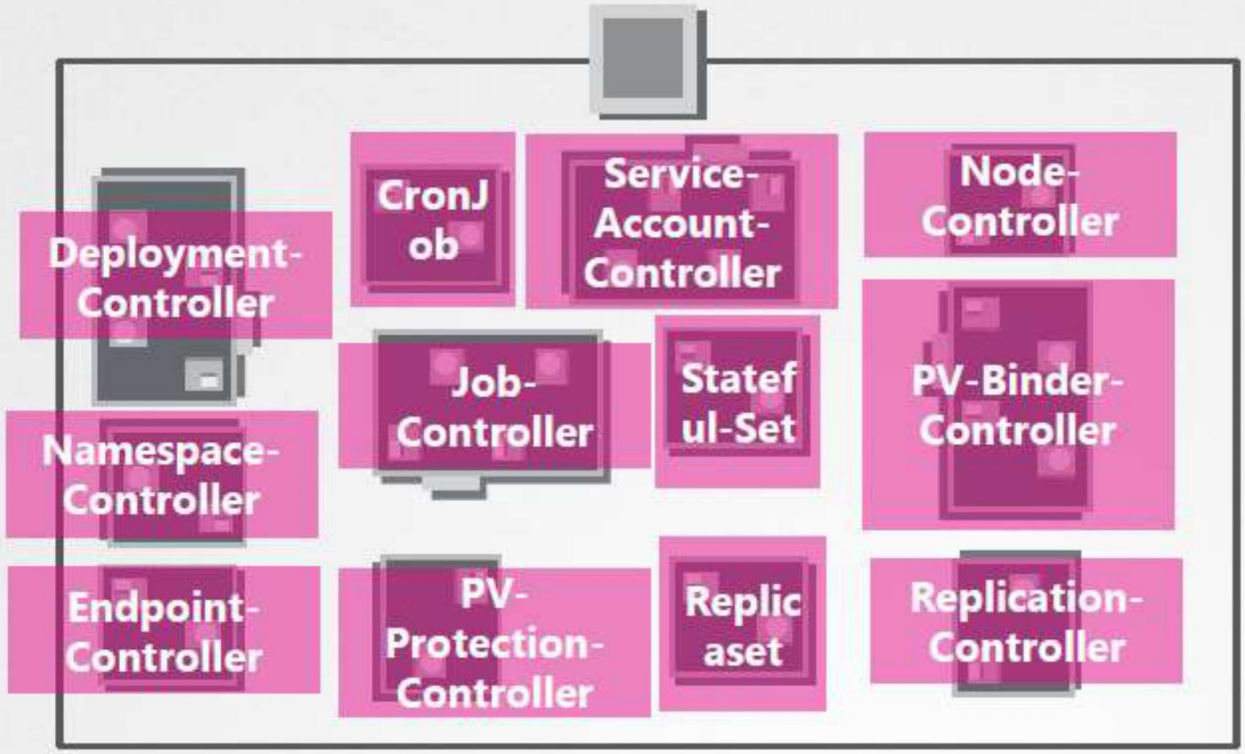
kube-controller-manager

- 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**.
- Node and Replication Controller

Node Controller

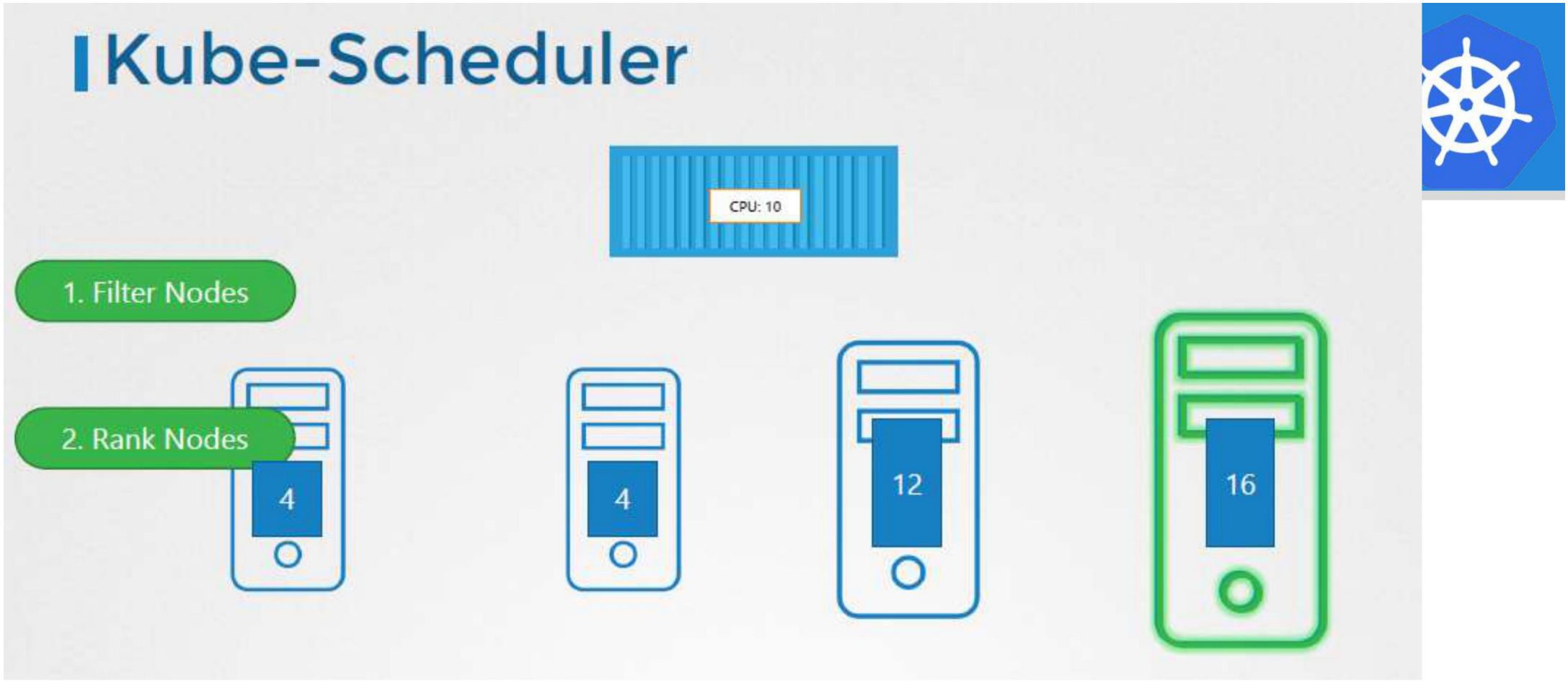


Controller


Deployment-Controller
CronJob
Service-Account-Controller
Node-Controller
Namespace-Controller
Job-Controller
Stateful-Set
PV-Binder-Controller
Endpoint-Controller
PV-Protection-Controller
Replicaset
Replication-Controller

kube-scheduler

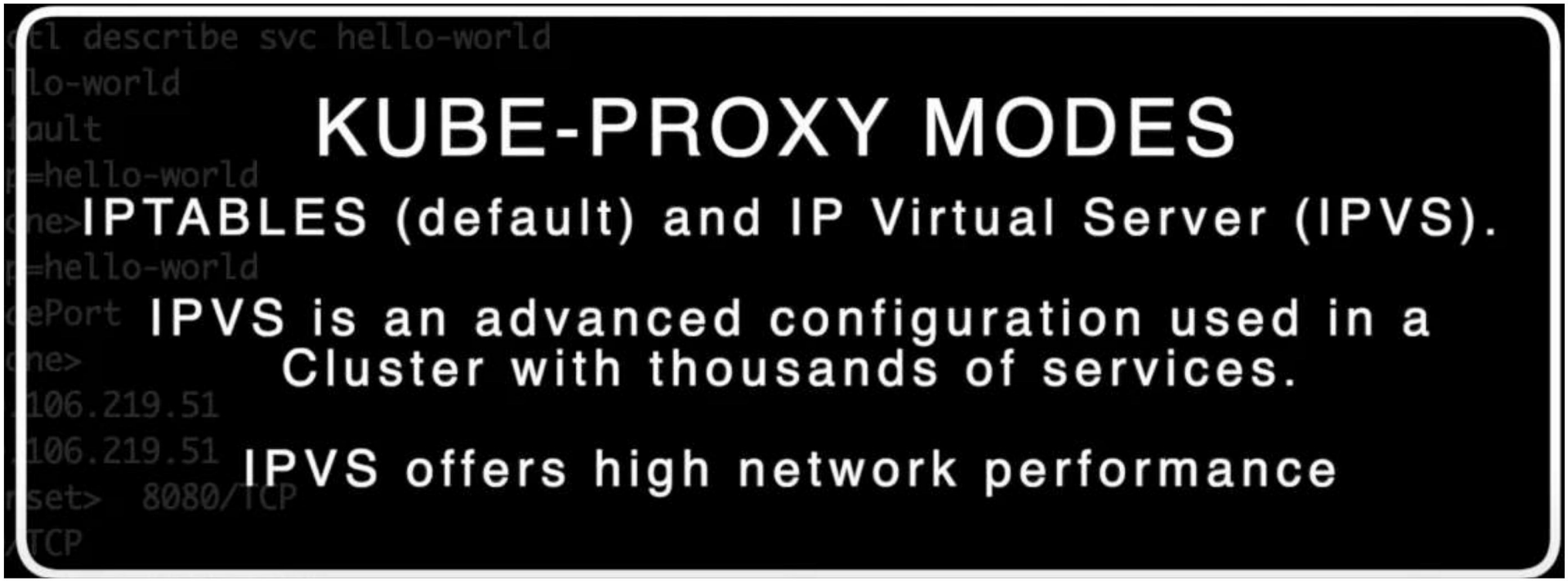
- Verbose policy-rich engine that evaluates workload requirements and attempts to place it on a matching resource.
- Scheduler only decides where pod will be deployed, than kubelet creates pod
- Workload Requirements can include: general hardware requirements, affinity/anti-affinity, labels, and other various custom resource requirements.



View kube-scheduler options -kubeadm

```
#cat /etc/kubernetes/manifests/kube-scheduler.yaml
```

#ps-aux | grep kube-scheduler


KUBE-PROXY MODES
IPTABLES (default) and IP Virtual Server (IPVS).
IPVS is an advanced configuration used in a Cluster with thousands of services.
IPVS offers high network performance

kube-dns/proxy

Here's how Kubernetes services work! A service is a collection of pods, which each have their own IP address (like 10.1.0.3, 10.2.3.5, 10.3.5.6)

1. Every Kubernetes service gets an IP address (like 10.23.1.2)
2. `kube-dns` resolves Kubernetes service DNS names to IP addresses (so my-svc.my-namespace.svc.cluster.local might map to 10.23.1.2)
3. `kube-proxy` sets up iptables rules in order to do random load balancing between them. Kube-proxy also has a userspace round-robin load balancer but my impression is that they don't recommend using it.

So when you make a request to `my-svc.my-namespace.svc.cluster.local`, it resolves to 10.23.1.2, and then iptables rules on your local host (generated by kube-proxy) redirect it to one of 10.1.0.3 or 10.2.3.5 or 10.3.5.6 at random.