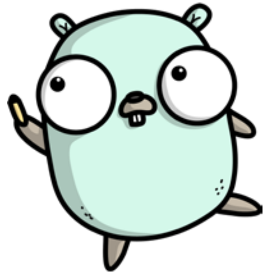
Kubernetes

A short introduction

Before We Begin

Requirements:

- **Minikube and or K3S** https://github.com/kubernetes/minikube
- **Virtualbox*:** https://www.virtualbox.org/wiki/Downloads
- **kubectl:** https://kubernetes.io/docs/tasks/tools/install-kubectl/
- **k8s-intro-tutorials repo:** https://github.com/mrbobbytables/k8s-intro-tutorials



Acknowledgements

These slides are an updated/modified version of the following ones:

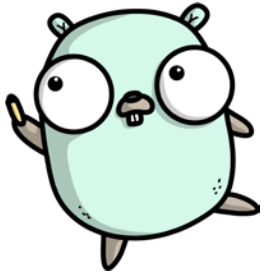
[Introduction to Kubernetes - Presentazioni Google] from Bob Killen

We (SC/ Marco Cesaratto/ Ruggero Lot) updated and modified it for the course Advanced Cloud computing.

We thanks Bob for his great work and to allow modifications and updates under CC by 4.0

Before We Begin

Go here for all info related to the course



[Foundations-of-HPC/Cloud-advanced-2023 (github.com)]

Kubernetes

A short introduction

Project Overview

What Does “Kubernetes” Mean?

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

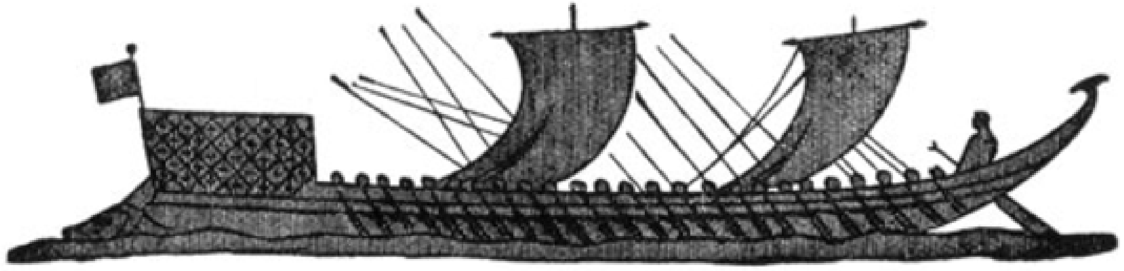


Image Source

What is Kubernetes?

- Project that was spun out of Google as an open source container orchestration platform.
- Built from the lessons learned in the experiences of developing and running Google’s Borg and Omega.
- Designed from the ground-up as a **loosely coupled** collection of components centered around deploying, maintaining and scaling workloads.

What Does Kubernetes do?

- Known as the **linux kernel of distributed systems**.
- **Abstracts away the underlying hardware** of the nodes and provides a uniform interface for workloads to be both deployed and consume the shared pool of resources.
- Works as an engine for resolving state by converging actual and the **desired state** of the system.

Decouples Infrastructure and Scaling

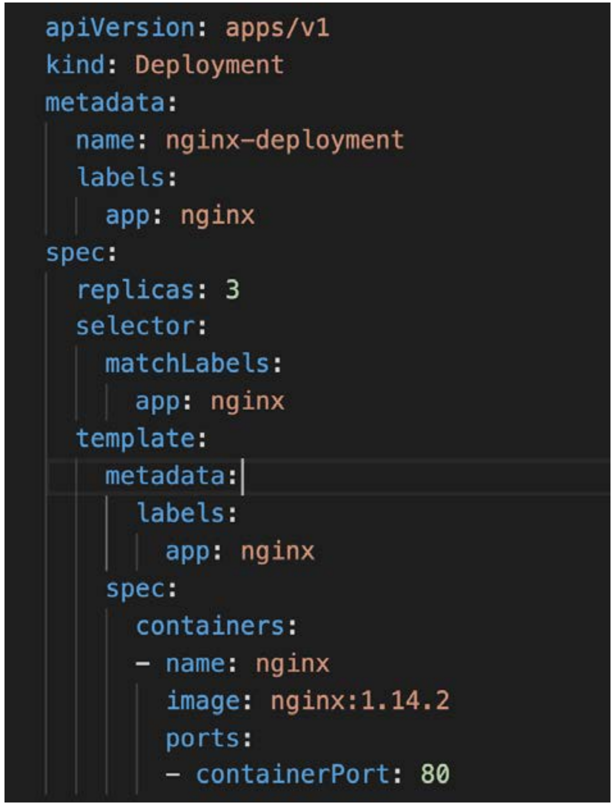
- **All services** within Kubernetes are natively Load Balanced.
- Can scale up and down dynamically.
- Used both to enable self-healing and seamless upgrading or rollback of applications.

Self Healing

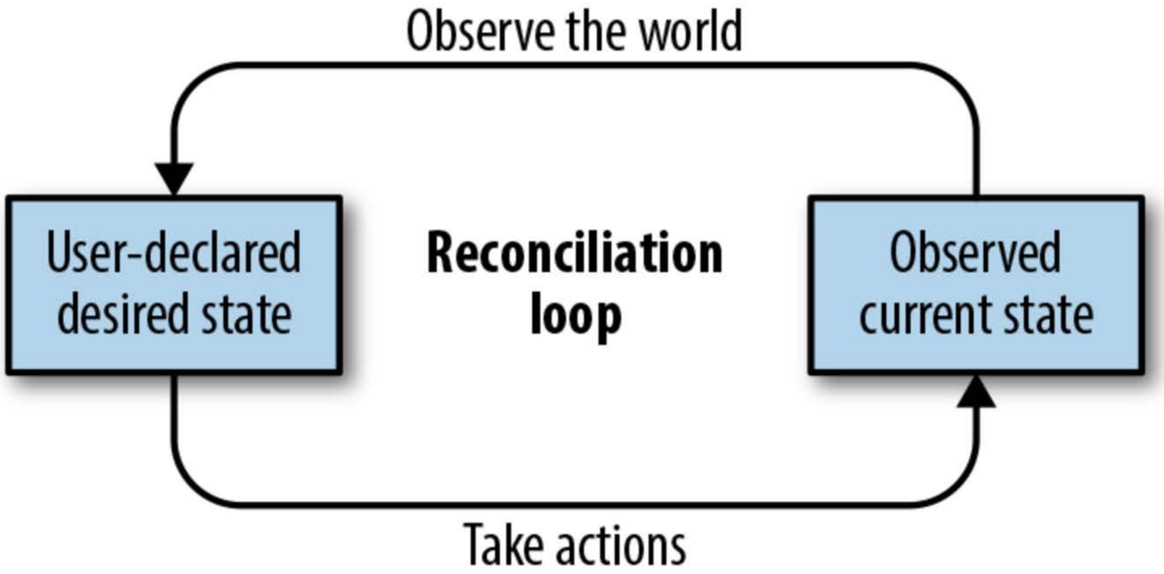
Kubernetes will **ALWAYS** try and steer the cluster to its desired state.

- **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.”

Self Healing



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```



What can Kubernetes REALLY do?

- Autoscale Workloads
- Blue/Green Deployments
- Fire off jobs and scheduled cronjobs
- Manage Stateless and Stateful Applications
- Provide native methods of service discovery
- Easily integrate and support 3rd party apps

Most Importantly...

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

Who “Manages” Kubernetes?

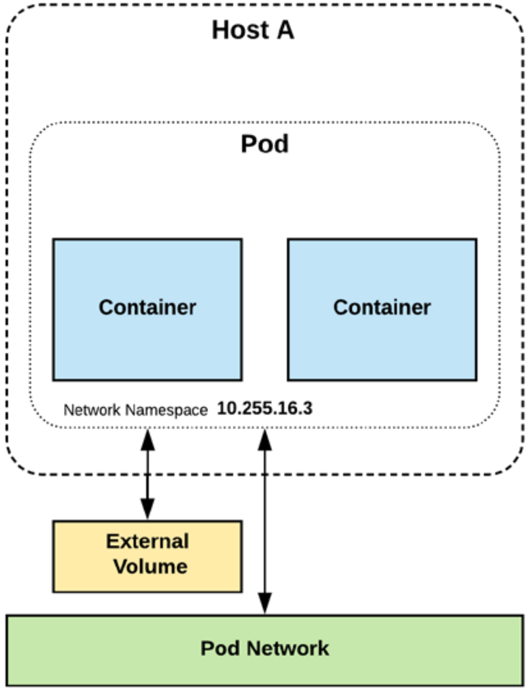
CLOUD NATIVE COMPUTING FOUNDATION

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

A Couple Key Concepts...

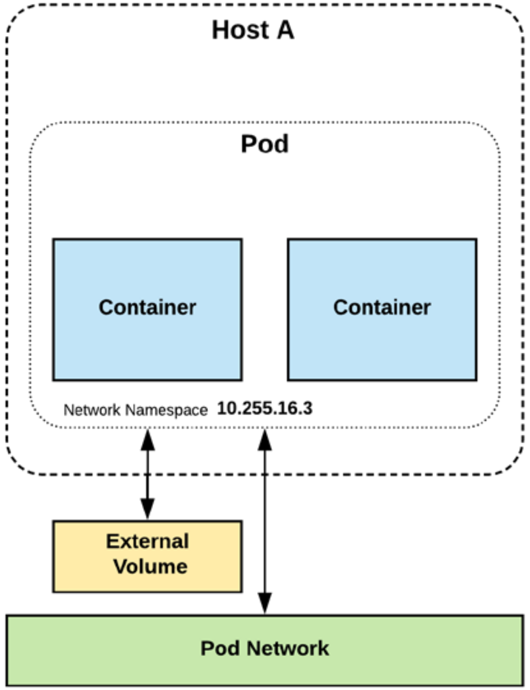
Pods

- **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**.



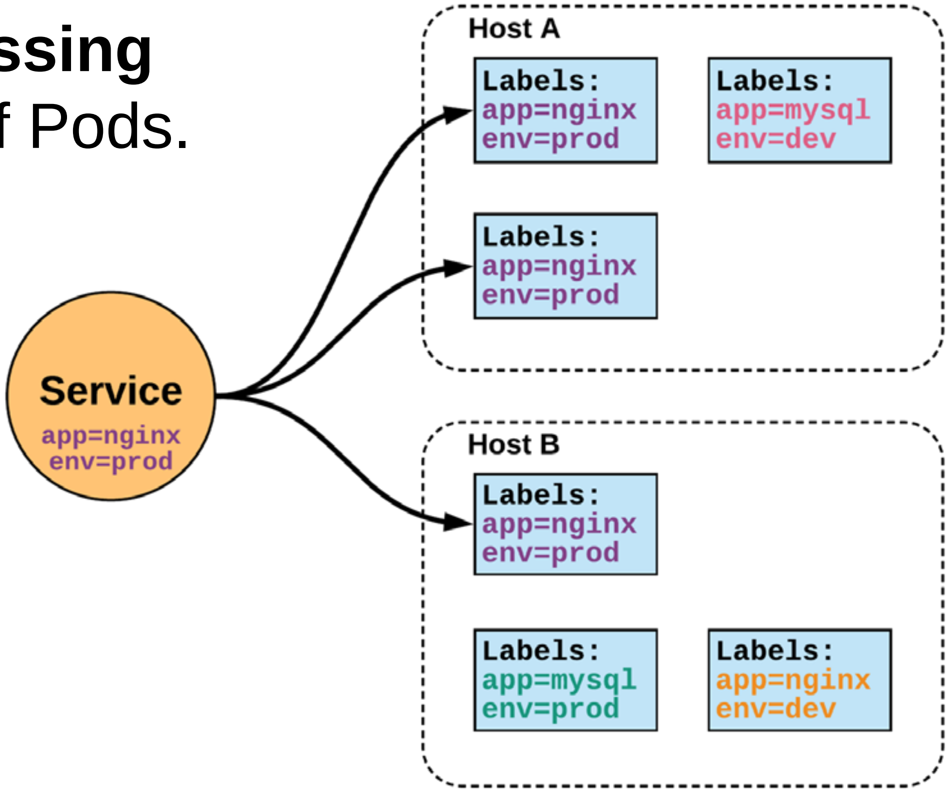
Pods

They are also Ephemeral!



Services

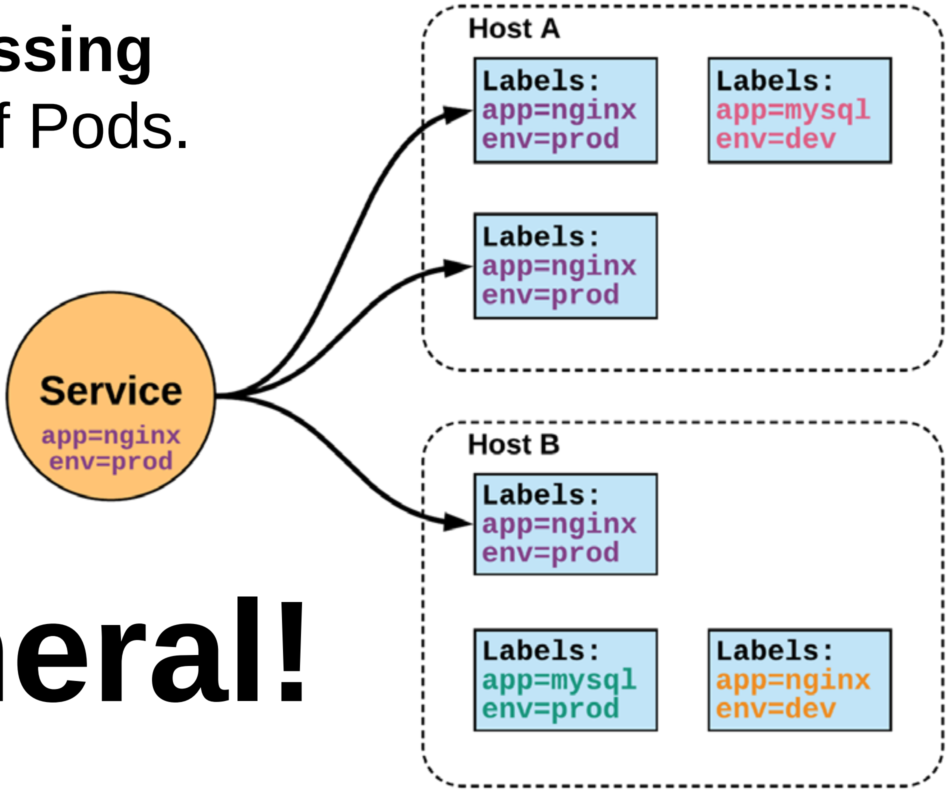
- **Unified method of accessing** the exposed workloads of Pods.
- **Durable resource**
    - static cluster IP
    - static namespaced DNS name



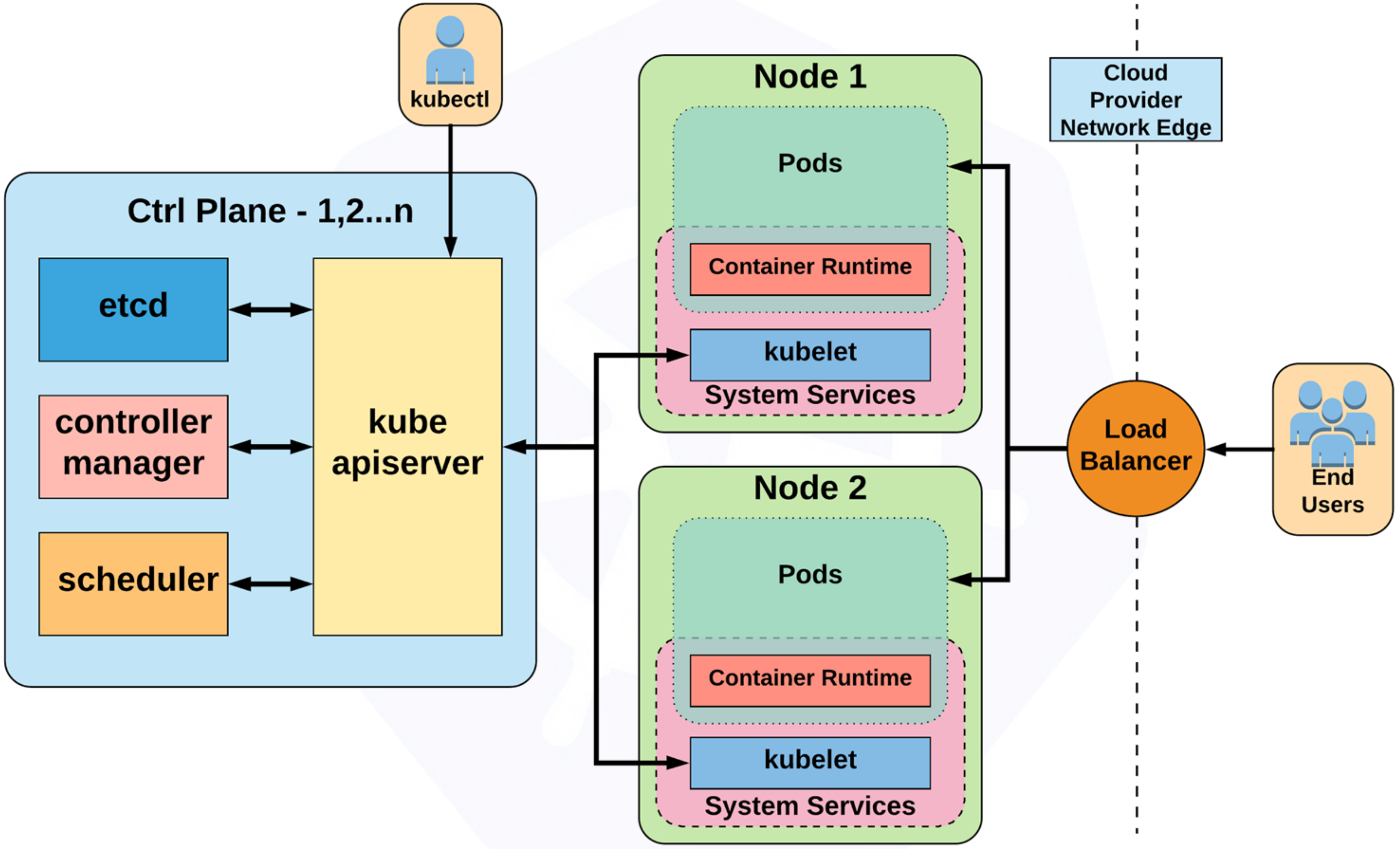
Services

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

NOT Ephemeral!



Architecture Overview

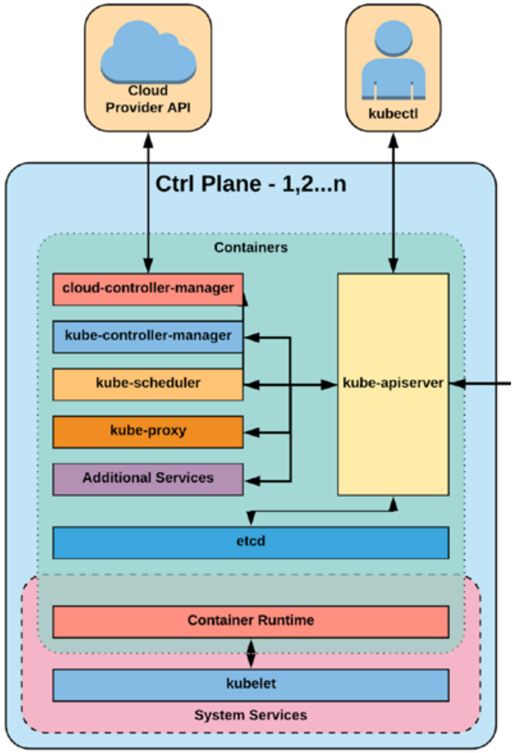

kubectl
Ctrl Plane - 1,2...n
etcd
controller manager
scheduler
kube apiserver
Node 1
Pods
Container Runtime
kubelet
System Services
Node 2
Pods
Container Runtime
kubelet
System Services
Cloud Provider Network Edge
Load Balancer
End Users

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.

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.

etcd

Uses *“Raft Consensus”* among a quorum of systems to create a fault-tolerant consistent *“view”* of the cluster.

https://raft.github.io/

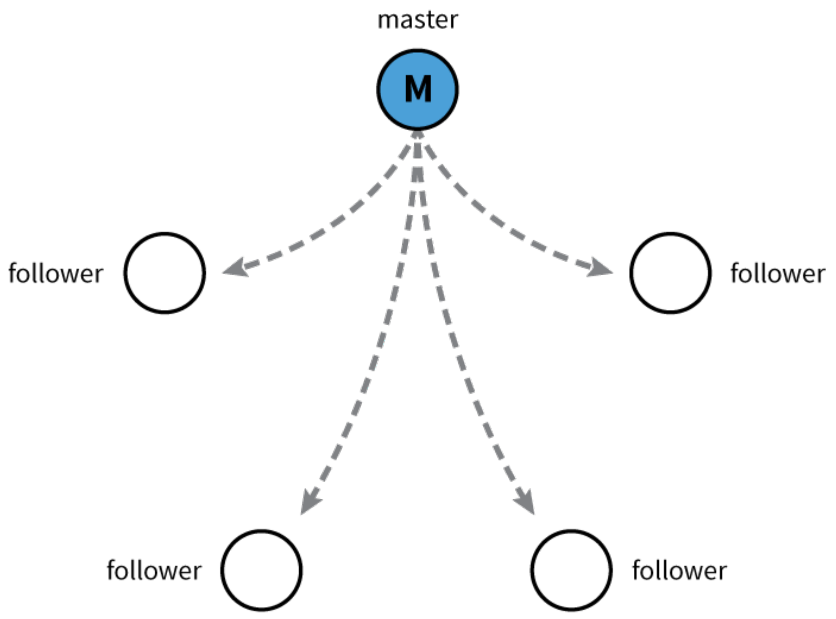


Image Source

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**.

List of core controllers: https://github.com/kubernetes/kubernetes/blob/master/cmd/kube-controller-manager/app/controllermanager.go#L344

kube-scheduler

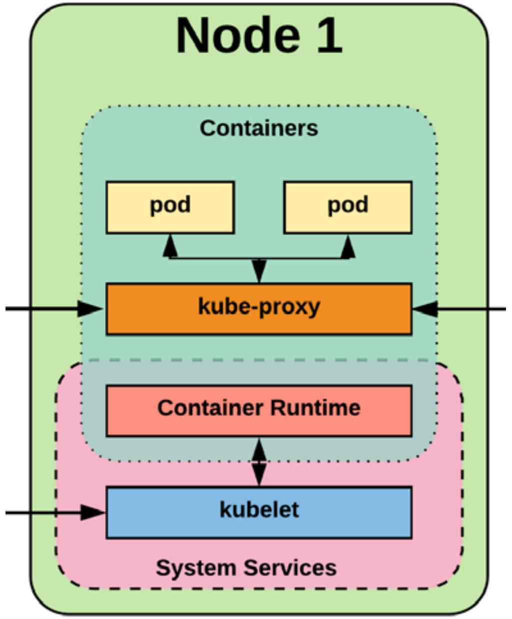
- Verbose policy-rich engine that evaluates workload requirements and attempts to place it on a matching resource.
- Default scheduler uses bin packing.
- Workload Requirements can include: general hardware requirements, affinity/anti-affinity, labels, and other various custom resource requirements.

Node Components

Architecture Overview

Node Components

- kubelet
- kube-proxy
- Container Runtime Engine



kubelet

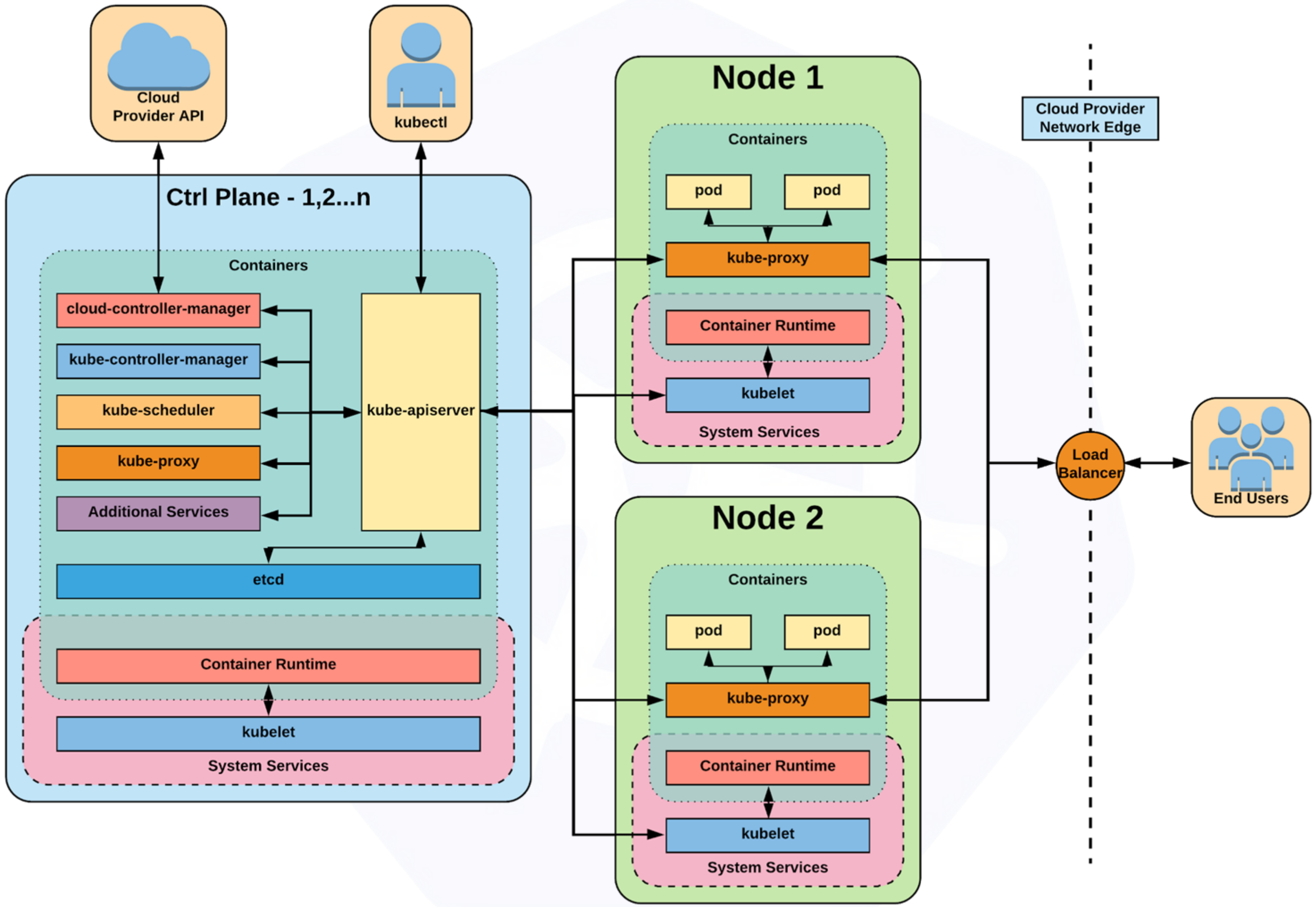
- Acts as the node agent responsible for managing the lifecycle of every pod on its host.
- Kubelet understands YAML container manifests that it can read from several sources:
  - file path
  - HTTP Endpoint
  - etcd watch acting on any changes
  - HTTP Server mode accepting container manifests over a simple API.

kube-proxy

- Manages the network rules on each node.
- Performs connection forwarding or load balancing for Kubernetes cluster services.
- Available Proxy Modes:
  - Userspace
  - iptables
  - ipvs (default if supported)

Container Runtime Engine

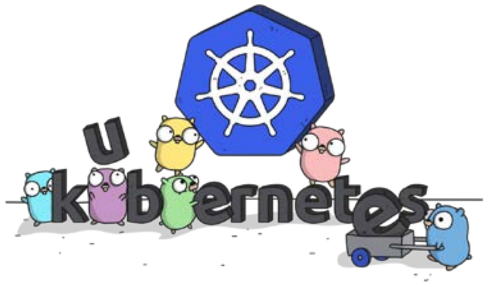
- A container runtime is a CRI (Container Runtime Interface) compatible application that executes and manages containers.
    - Containerd (docker)
    - Cri-o
    - Rkt
    - Kata (formerly clear and hyper)
    - Virtlet (VM CRI compatible runtime)


Cloud Provider API
kubectl
Ctrl Plane - 1,2...n
Containers
cloud-controller-manager
kube-controller-manager
kube-scheduler
kube-proxy
Additional Services
kube-apiserver
etcd
Container Runtime
kubelet
System Services
Node 1
Containers
pod
pod
kube-proxy
Container Runtime
kubelet
System Services
Node 2
Containers
pod
pod
kube-proxy
Container Runtime
kubelet
System Services
Cloud Provider Network Edge
Load Balancer
End Users

End of the first part

Links

- **Free Kubernetes Courses**
  https://www.edx.org/
- **Interactive Kubernetes Tutorials**
  https://www.katacoda.com/courses/kubernetes
- **Learn Kubernetes the Hard Way**
  https://github.com/kelseyhightower/kubernetes-the-hard-way
- **Official Kubernetes Youtube Channel**
  https://www.youtube.com/c/KubernetesCommunity
- **Official CNCF Youtube Channel**
  https://www.youtube.com/c/cloudnativefdn
- **Track to becoming a CKA/CKAD (Certified Kubernetes Administrator/Application Developer)**
  https://www.cncf.io/certification/expert/
- **Awesome Kubernetes**
  https://ramitsurana.gitbooks.io/awesome-kubernetes/content/



Questions?