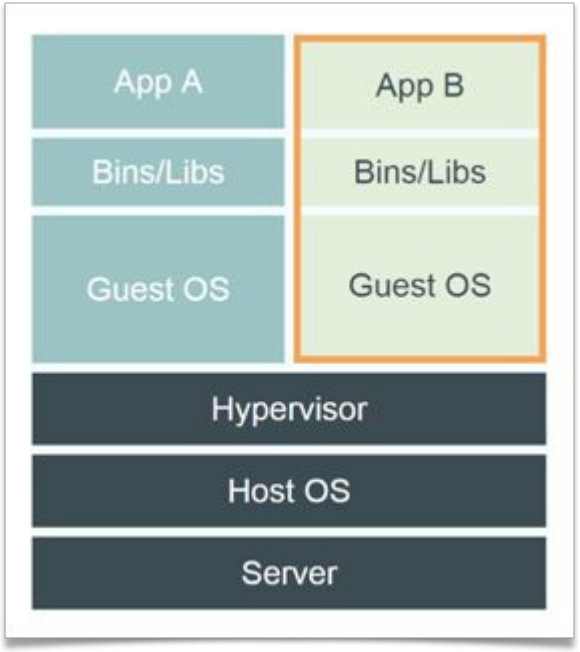
# Kubernetes 101

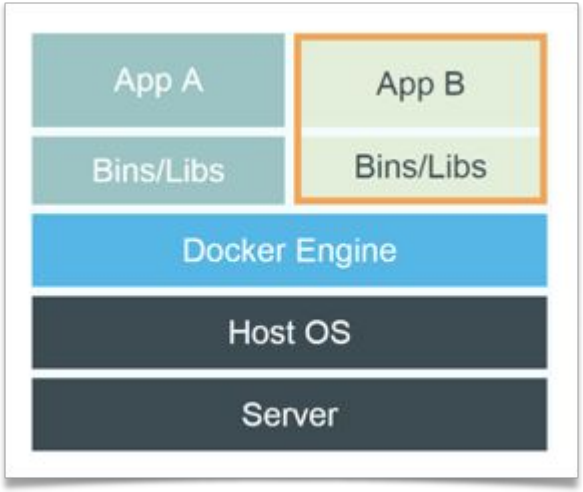
Vinayak Shinde
CKA | MTS @Portworx

# Agenda

1. What are Containers ?
2. What is Kubernetes ?
3. Key Concepts
4. Architecture
5. Summery

# What are containers ?




kubernetes

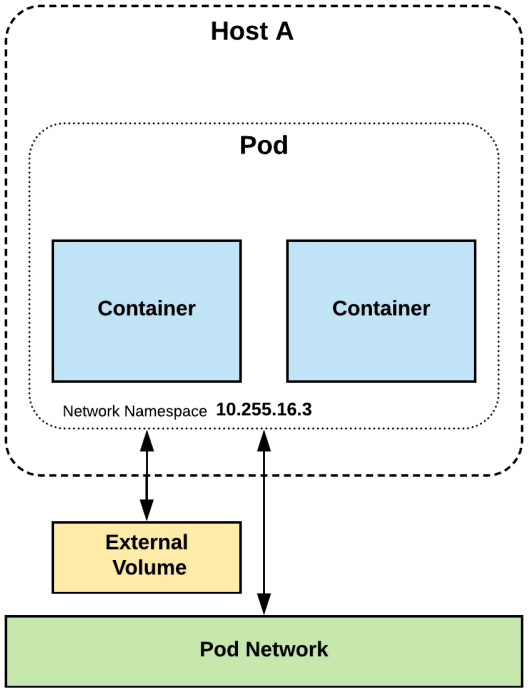
# What is Kubernetes ?

- A **Production-Grade Container Orchestration System** Google-grown, based on Borg and Omega, systems that run inside of Google right now and are proven to work at Google for over 10 years.
- Google spawns billions of containers per week with these systems.
- Created by three Google employees initially during the summer of 2014; grew exponentially and became the first project to get donated to the CNCF.

# Key Concepts

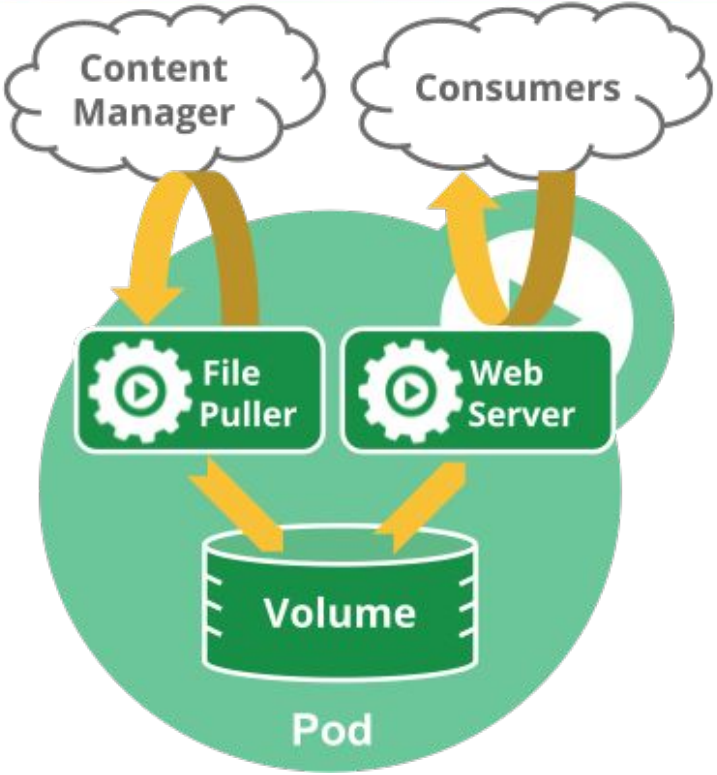
# 1. Pod

- **Atomic unit** or smallest "*unit of work*"of Kubernetes.
- **One or MORE containers** that share
  - Storage: Have access to shared volumes.
  - Network: An IP address and port space, and can find each other via **localhost**.
- **Ephemeral**




kubernetes

# 1. Pod




kubernetes

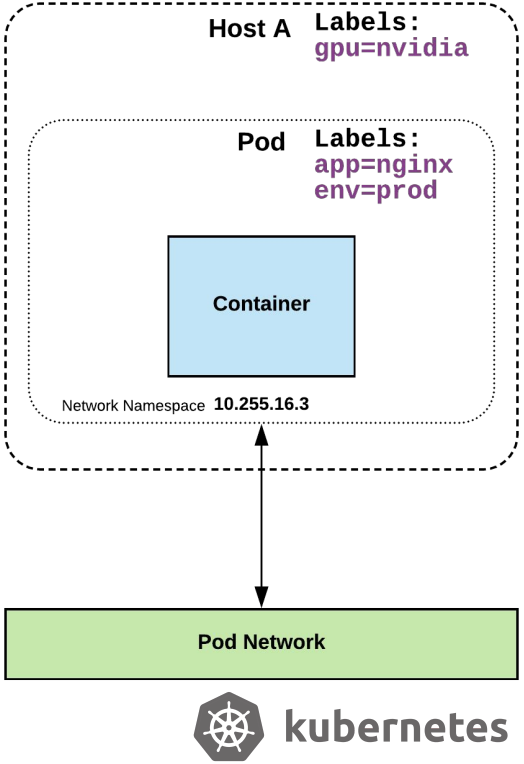
# Pod Example

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    resources: {}
```

kubernetes

# 2. Labels

- key-value pairs that are used to identify, describe and group together related sets of objects or resources.
- **NOT** characteristic of uniqueness.
- Have a strict syntax with a slightly limited character set*.




kubernetes

# 3. Selectors

- Selectors use labels to filter or select objects, and are used throughout Kubernetes.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-label-example
  labels:
    app: nginx
    env: prod
spec:
  containers:
  - name: nginx
    image: nginx:stable-alpine
    ports:
    - containerPort: 80
  nodeSelector:
    gpu: nvidia
```

kubernetes

# 3. Selector Types

**Equality based** selectors allow for simple filtering (=,==, or !=).

```
selector:
  matchLabels:
    gpu: nvidia
```

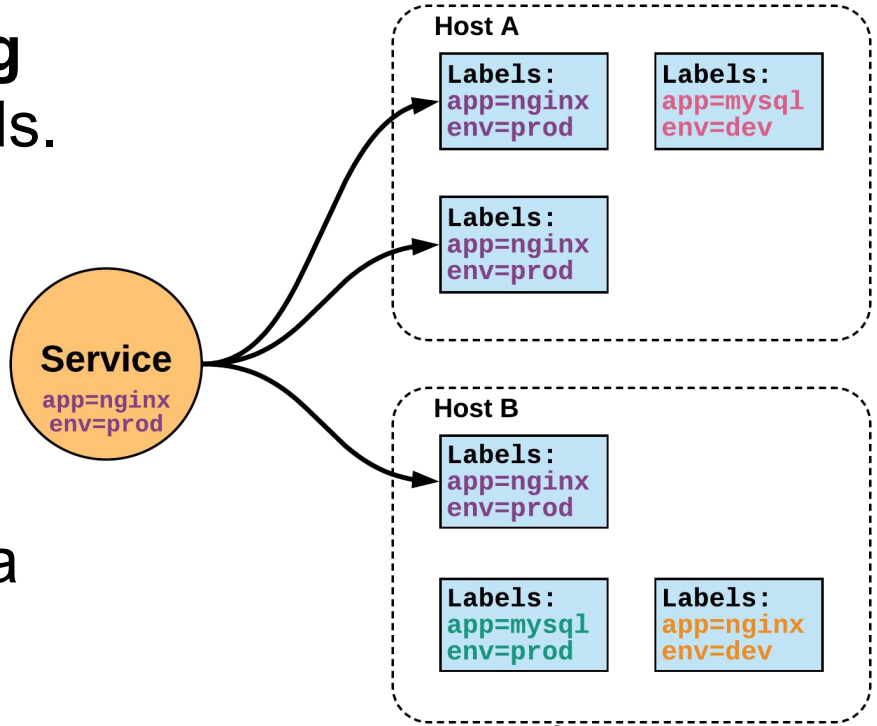
**Set-based** selectors are supported on a limited subset of objects. However, they provide a method of filtering on a set of values, and supports multiple operators including: `in`, `notin`, and `exist`.

```
selector:
  matchExpressions:
    - key: gpu
      operator: in
      values: [“nvidia”]
```

kubernetes

# 4. Services

- **Unified method of accessing** the exposed workloads of Pods.
- **Durable resource**
  - static cluster IP
  - static namespaced DNS name
- An abstraction which defines a logical set of Pods
- Uses pod readiness probes




kubernetes

# Probes

- **Readiness Probe:**
  - Indicates whether the Container is ready to service requests. Cont. may take time to load data etc.
  - Used by Services
- **Liveness Probe:**
  - Indicates whether the Container is running.
  - If it fails, kubernetes kills the Container
  - Useful to detect app deadlocks

# Readiness Probe

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    resources: {}
   readinessProbe:
      httpGet:
        path: /healthz
        port: 80
      initialDelaySeconds: 100
      periodSeconds: 10
      timeoutSeconds: 4
      failureThreshold: 2
```

kubernetes

# Liveness Probe

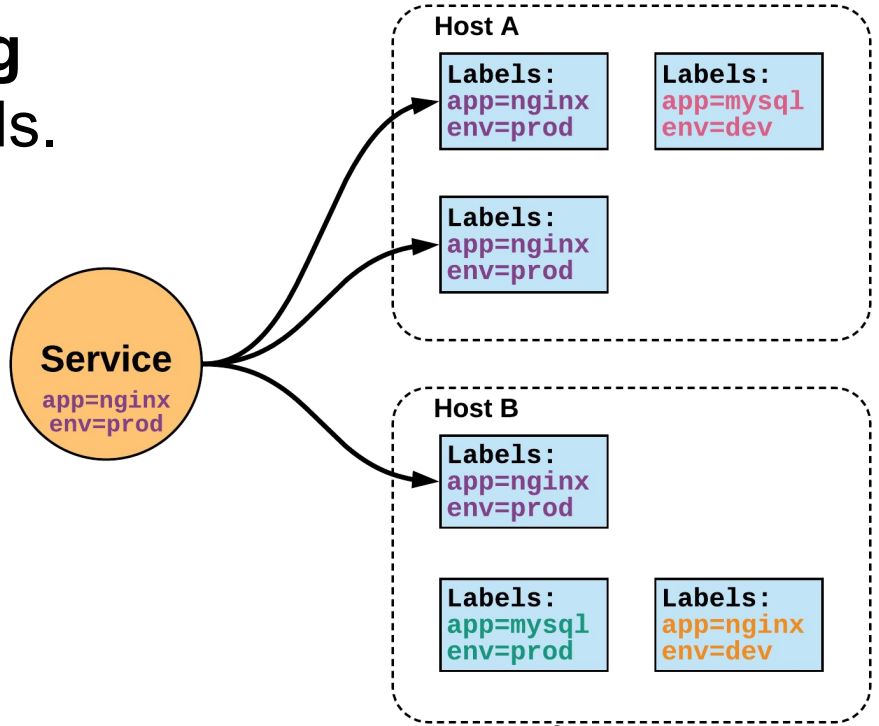
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
  livenessProbe:
      httpGet:
        path: /healthz
        port: 80
      initialDelaySeconds: 100
      periodSeconds: 10
      timeoutSeconds: 4
      failureThreshold: 2
```

kubernetes

# 3. Services

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

**NOT Ephemeral !!!**




kubernetes

# 3. Services

There are 4 major service types:

- **ClusterIP** (default)
- **NodePort**
- **LoadBalancer**
- **ExternalName**

# ClusterIP Service

**ClusterIP** services exposes a service on a strictly cluster internal virtual IP.

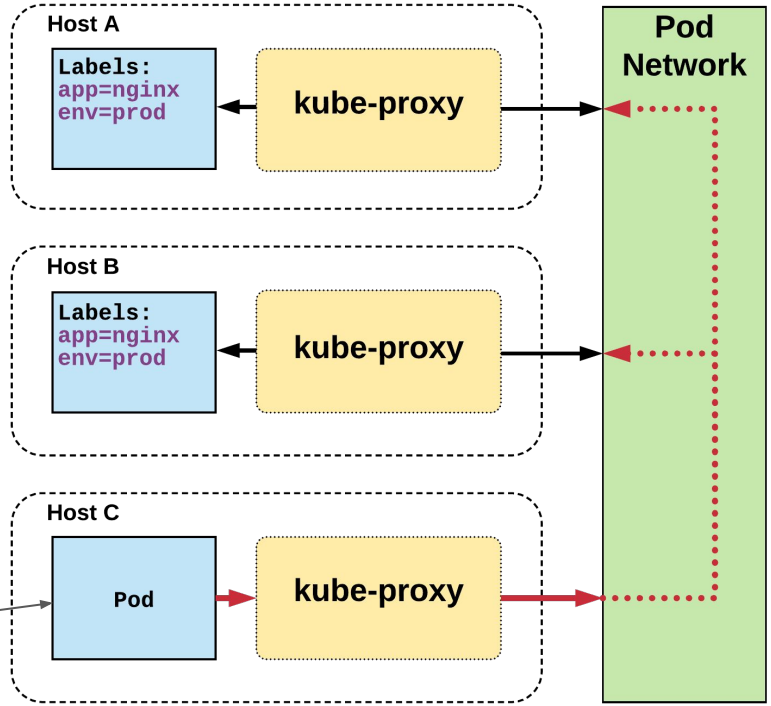
```yaml
apiVersion: v1
kind: Service
metadata:
  name: example-prod
spec:
  selector:
    app: nginx
    env: prod
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```

# ClusterIP Service

```
Name:           example-prod
Selector:       app=nginx,env=prod
Type:           ClusterIP
IP:             10.96.28.176
Port:           <unset>  80/TCP
TargetPort:     80/TCP
Endpoints:      10.255.16.3:80,
                10.255.16.4:80
```

```
/ # nslookup example-prod.default.svc.cluster.local

Name:      example-prod.default.svc.cluster.local
Address 1: 10.96.28.176 example-prod.default.svc.cluster.local
```



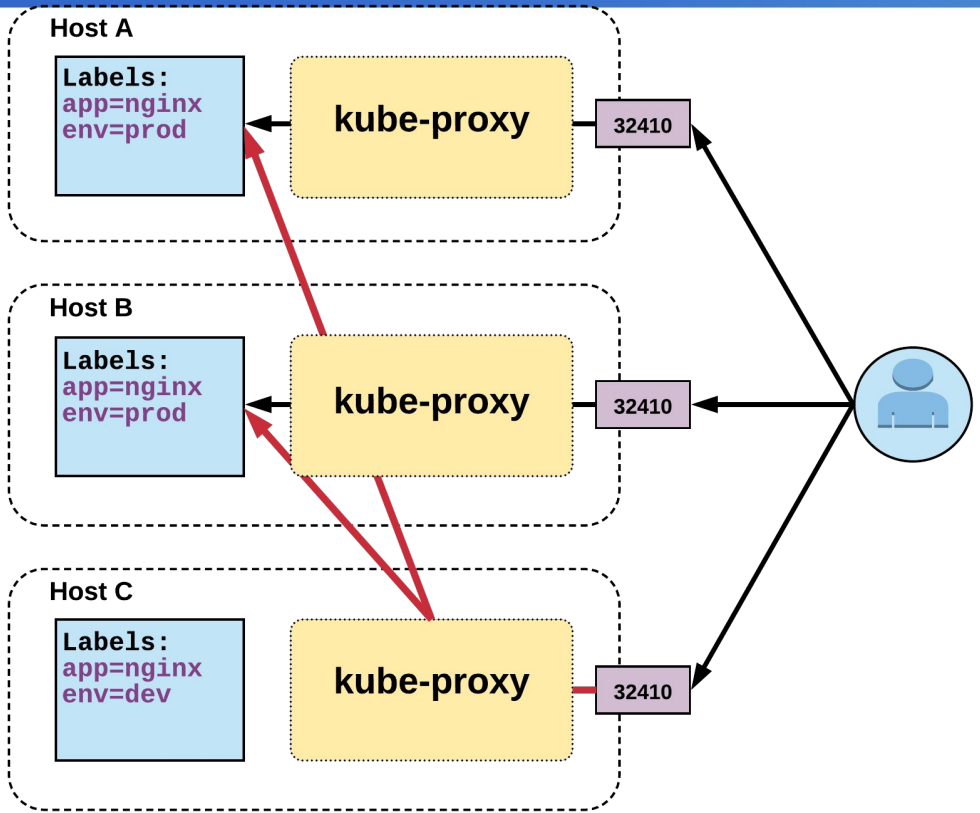

kubernetes

# Node Port Service

- **NodePort** services extend the **ClusterIP** service.
- Exposes a port on every node's IP.
- Port can either be statically defined, or dynamically taken from a range between 30000-32767.

```yaml
apiVersion: v1
kind: Service
metadata:
  name: example-prod
spec:
  type: NodePort
  selector:
    app: nginx
    env: prod
  ports:
  - nodePort: 32410
    protocol: TCP
    port: 80
    targetPort: 80
```

# Node Port Service




```
Name:          example-prod
Selector:      app=nginx,env=prod
Type:          NodePort
IP:            10.96.28.176
Port:          <unset>  80/TCP
TargetPort:    80/TCP
NodePort:      <unset>  32410/TCP
Endpoints:     10.255.16.3:80,
               10.255.16.4:80
```

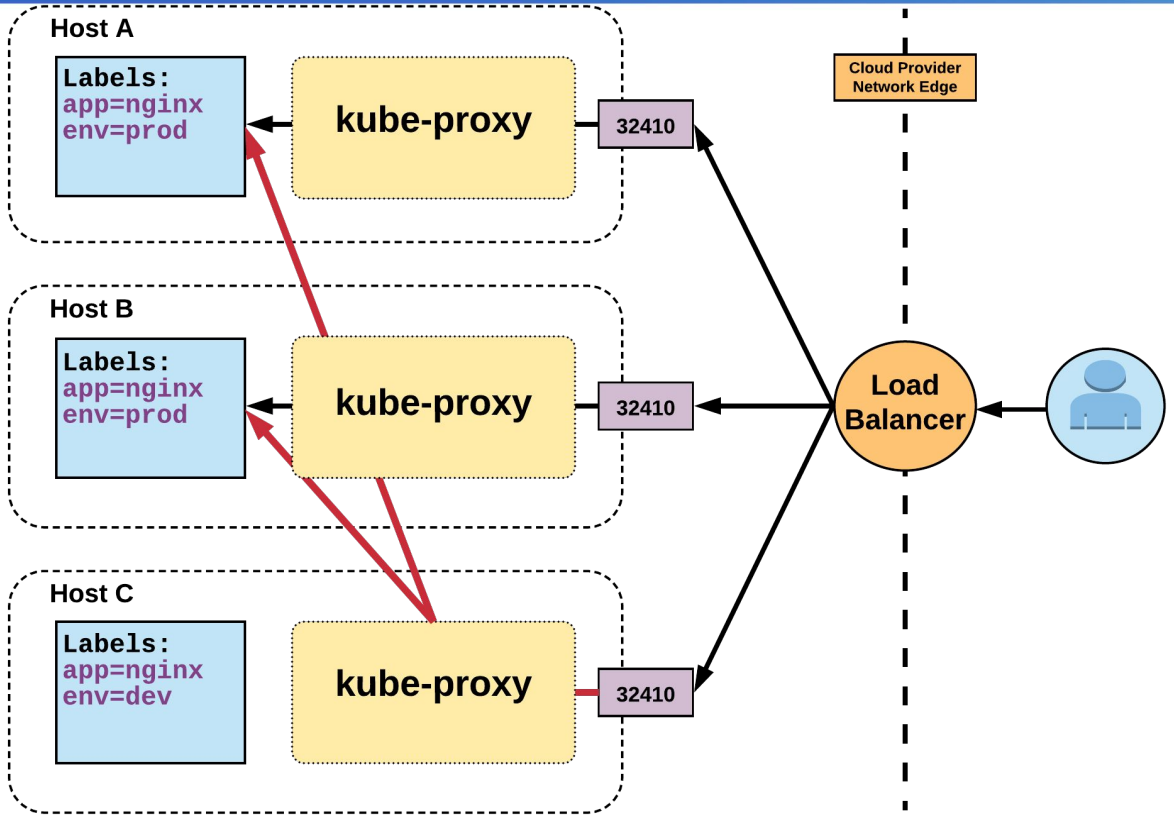
kubernetes

# Load Balancer Service

- **LoadBalancer** services extend **NodePort.**
- Works in conjunction with an external system to map a cluster external IP to the exposed service.

```
apiVersion: v1
kind: Service
metadata:
  name: example-prod
spec:
  type: LoadBalancer
  selector:
    app: nginx
    env: prod
  ports:
    protocol: TCP
    port: 80
    targetPort: 80
```

# Load Balancer Service




```
Name:          example-prod
Selector:      app=nginx,env=prod
Type:          LoadBalancer
IP:            10.96.28.176
LoadBalancer
Ingress:       172.17.18.43
Port:          <unset>  80/TCP
TargetPort:    80/TCP
NodePort:      <unset>  32410/TCP
Endpoints:     10.255.16.3:80,
               10.255.16.4:80
```

kubernetes

# ExternalName Service

- **ExternalName** is used to reference endpoints **OUTSIDE** the cluster.
- Creates an internal **CNAME** DNS entry that aliases another.

```
apiVersion: v1
kind: Service
metadata:
  name: example-prod
spec:
  type: ExternalName
spec:
  externalName:
example.com
```

# 4. ReplicaSets

- Primary method of managing pod replicas and their lifecycle.
- Includes their scheduling, scaling, and deletion.
- Their job is simple: **Always ensure the desired number of pods are running.**

# 4. ReplicaSets

- `replicas:` The desired number of instances of the Pod.
- `selector:` The label selector for the **ReplicaSet** will manage **ALL** Pod instances that it targets; whether it's desired or not.

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: rs-example
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
      env: prod
  template:
    <pod template>
```

kubernetes

# 4. ReplicaSets

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: rs-example
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
      env: prod
  template:
    metadata:
      labels:
        app: nginx
        env: prod
    spec:
      containers:
      - name: nginx
        image: nginx:stable-alpine
        ports:
        - containerPort: 80
```

kubernetes

# Architecture

# Control Plane Components

- Kube API Server
- ETCD
- Kube Controller Manager
- Kube-Scheduler

# Kube API Server

- 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.

# 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.


kubernetes

# Kube Controller Manager

- Monitors the cluster state via the API server and **steers the cluster towards the desired state**. (Reconciliation loop - Imperative vs Declarative)
- **Node Controller**: Responsible for noticing and responding when nodes go down.
- **Replication Controller:** Responsible for maintaining the correct number of pods for every replication controller object in the system.
- **Endpoints Controller:** Populates the Endpoints object (that is, joins Services & Pods).

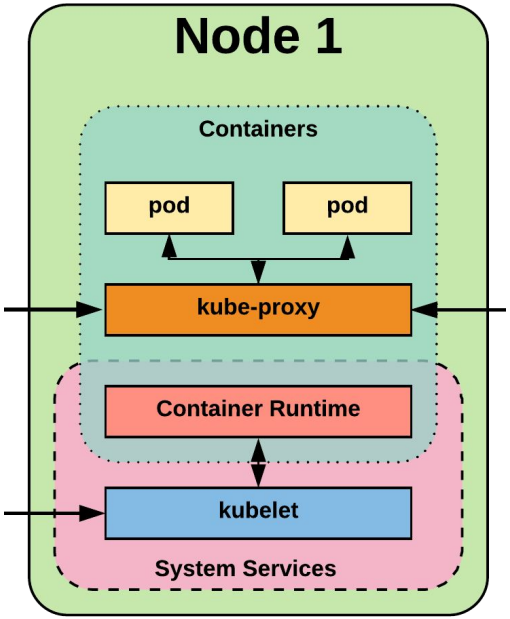
# Kube Scheduler

- 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.
- Can be **extended**.

kubernetes

# Node Components

- Kubelet
- KubeProxy
- Container Runtime Engine




kubernetes

# Kubelet

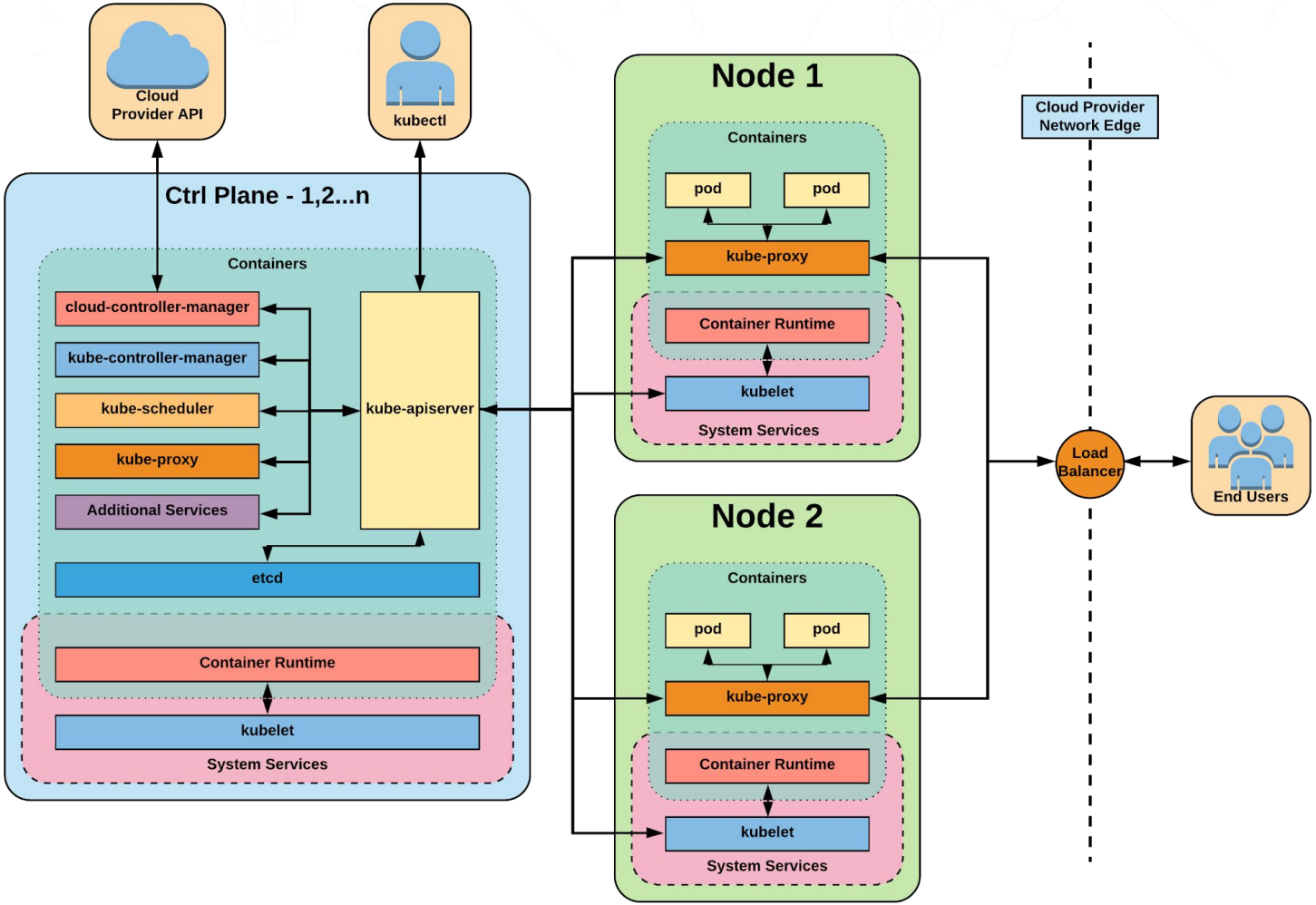
- An **agent** that runs on each node. It makes sure that containers are running in a pod.
- **Takes a set of PodSpecs** and ensures that the containers described in those PodSpecs are running and healthy.
- **On workers:** poll kube-apiserver looking for what they should run
- **On masters:** run the master services as static manifests found locally on the host (/etc/kubernetes/manifests/)

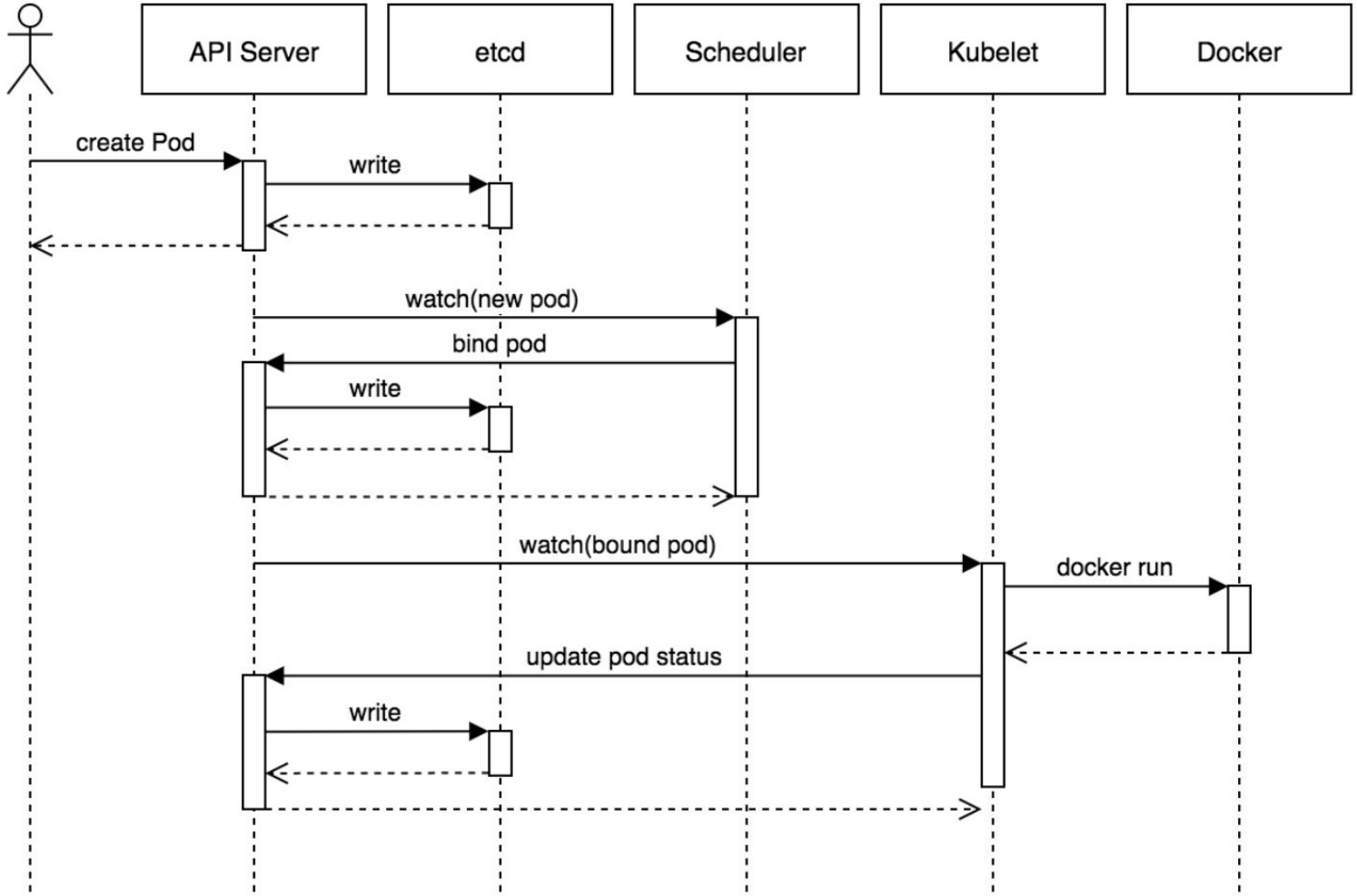
# KubeProxy

- Manages the network rules on each node.
- Performs connection forwarding or load balancing for Kubernetes cluster services.

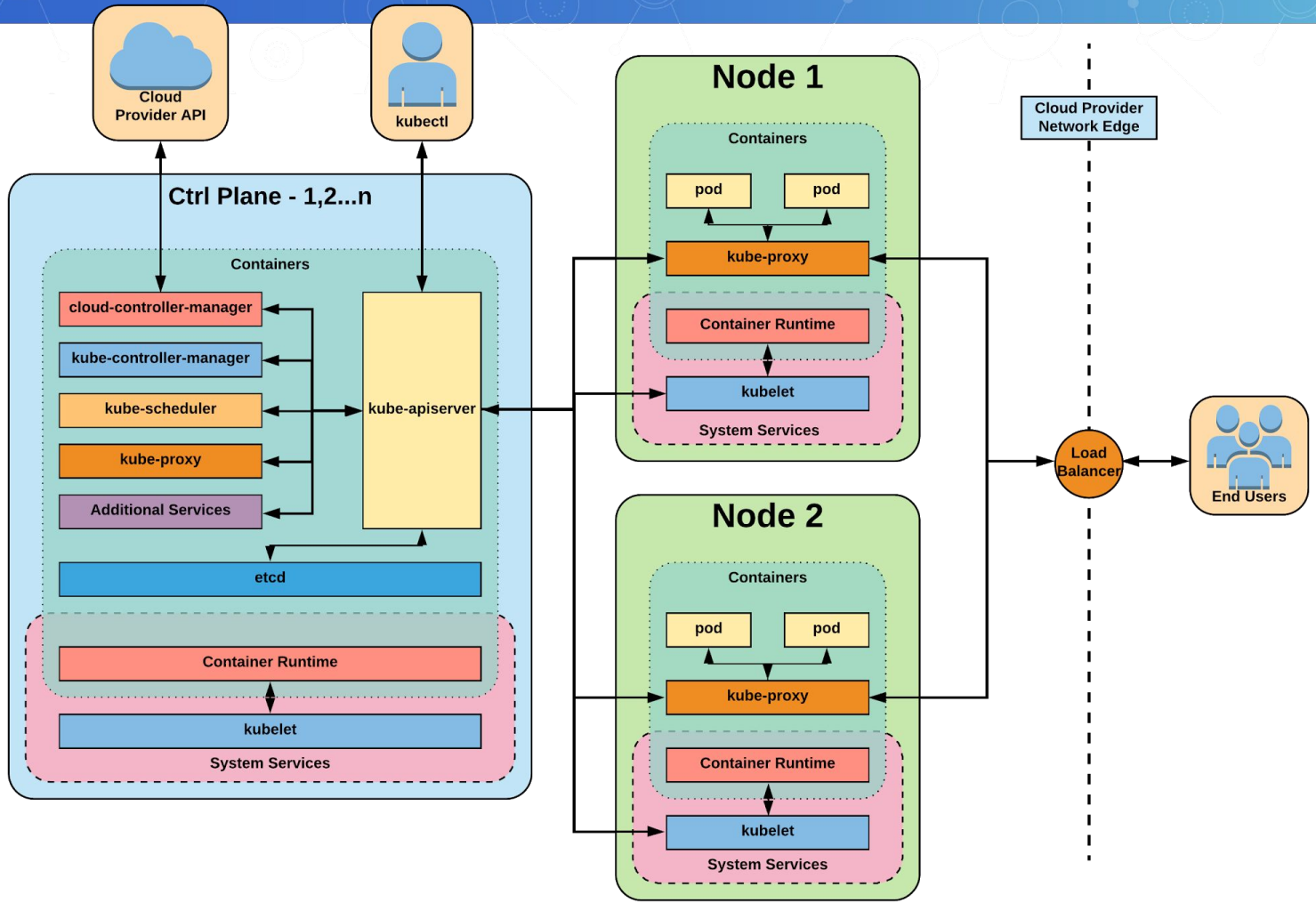
# Container Runtime Engine

- 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


API Server
etcd
Scheduler
Kubelet
Docker
create Pod
write
watch(new pod)
bind pod
write
watch(bound pod)
docker run
update pod status
write

# Summery


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

# Q&A

- by Joe Beda (Gluecon 2017)

# Patterns for composite containers

- Sidecar
- Ambassador
- Adaptor

https://kubernetes.io/blog/2015/06/the-distributed-system-toolkit-patterns/

https://storage.googleapis.com/pub-tools-public-publication-data/pdf/45406.pdf

kubernetes