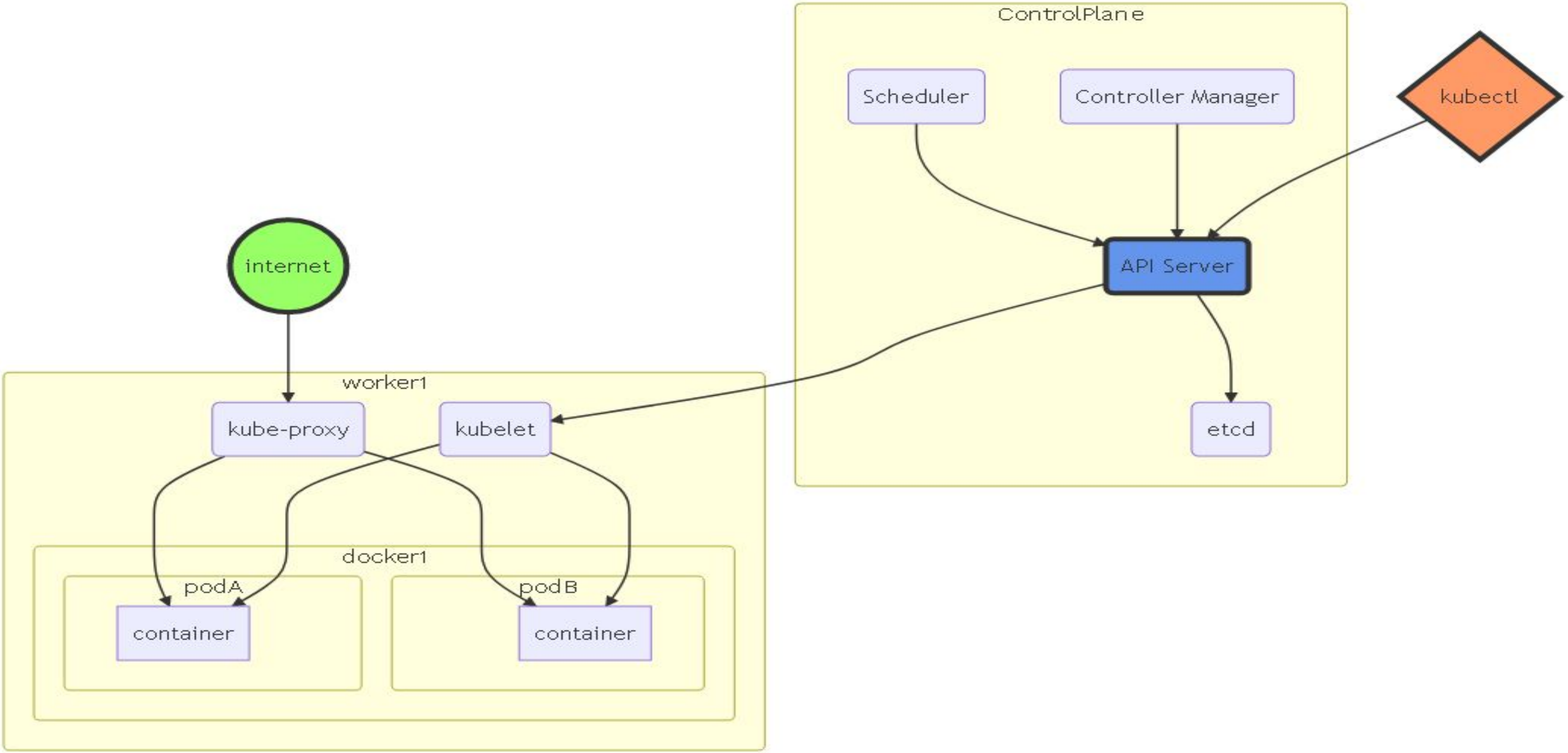
Kubernetes : Overview

- What is Kubernetes?
- Kubernetes Nodes
- Kubernetes Architecture:
- Kubernetes Components:
- Running and managing containers using Kubernetes
- Kubernetes Cluster
- Application Deployment Model
- Amazon EKS
- Companies adopting Amazon EKS
- Kubernetes Provides :
- K8s : Customer Case Studies

What is Kubernetes?

- **Docker** is a container runtime while **Kubernetes** is a platform for running and managing containers.
- It is an open source **orchestration** platform for automating deployment, scaling and the operations of application **containers** across *clusters of hosts*.
- It is also defined as a platform for **creating, deploying and managing various distributed applications.**
- These applications may be of different sizes and shapes.
- Kubernetes was originally **developed by Google** to deploy scalable, reliable systems in containers via application-oriented APIs.
- Kubernetes is suitable not only for Internet-scale companies but also for cloud-native companies, of all sizes.

Kubernetes Architecture:



Kubernetes Nodes

- Kubernetes follows **master-slave** architecture.
- Kubernetes architecture has a **master node** and **worker nodes**.

There are four components of a **master node.**

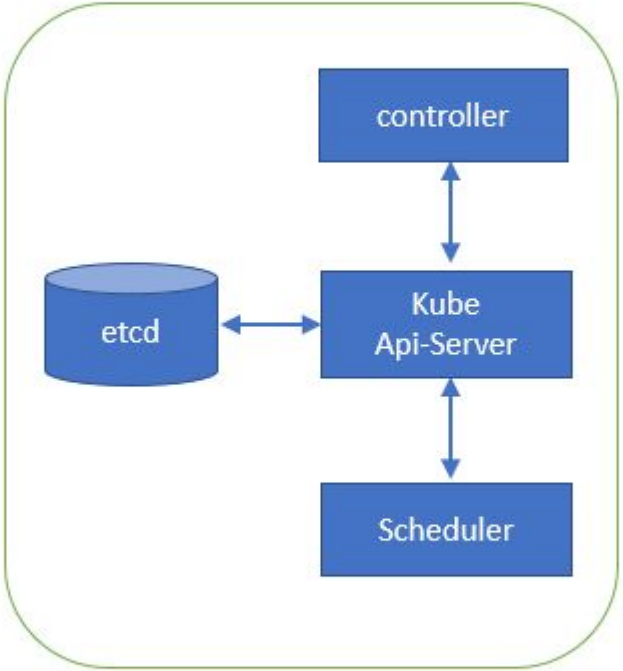
- Kube API Server
- controller
- scheduler
- etcd

There are three components of a **worker node.**

- kubelet
- kube-proxy
- Container runtime

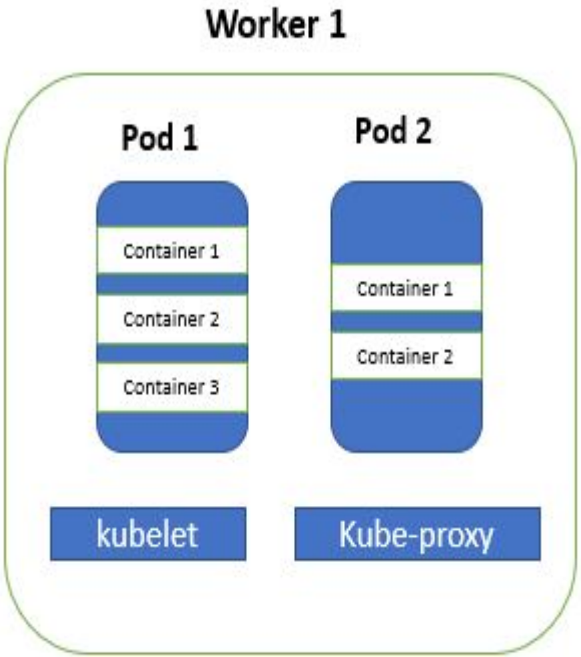
A worker node is a **virtual or physical server** that runs the applications and is controlled by the master node.
Each node has a **Kubelet**, which is an agent for managing the node and communicating with the Kubernetes control plane.

Master Node



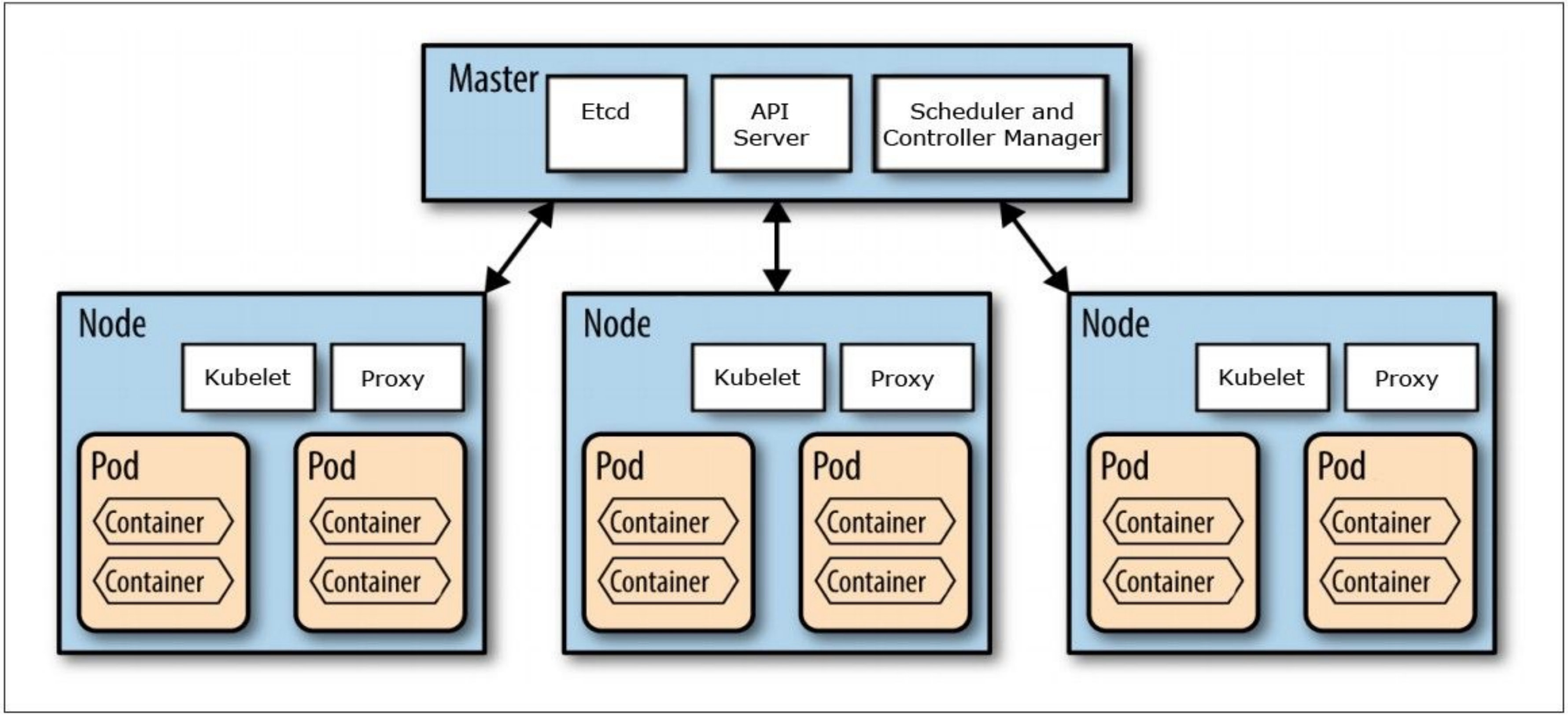
- **API Server** performs all the administrative tasks on the master node. A user sends the rest commands to the API server, which then validates the requests, then processes and executes them. **etcd** saves the resulting state of the cluster as a distributed key-value store.
- **The scheduler,** schedules the work to different worker nodes. It has the resource usage information for each worker node. The scheduler considers service requirements and **node resource parameters** and schedules the work in terms of **pods and services**.
- The **controller manager** makes sure that your **current state is the same as the desired state.** It manages the Non-terminating control loops that regulate the state of the **Kubernetes cluster**, here each one of these control loops knows about the desired state of the object it manages, and then they look at their current state through the API servers.
- The **etcd** is a **distributed key-value store** that is used to store the cluster state. It is also used to store the configuration details such as the subnets and the config maps.

Worker Node



- The **container runtime** is basically used to run and manage a continuous life cycle on the worker node.
- **Kubelet** is basically an **agent** that runs on **each worker node** and **communicates with the master node**. It receives the pod definition by various means and runs the containers associated with that port. It also makes sure that the containers which are part of the pods are always healthy.
- **Kube-proxy** runs on each worker node as the network proxy. It listens to the API server for each service point creation or deletion. For each service point, kube-proxy sets the routes so that it can reach to it.

Kubernetes Architecture:



Kubernetes Components:

NameSpaces: It is used to remove name collision within a cluster. It supports multiple virtual clusters on the same physical cluster. This provides isolation and complete access to control the degree to which other services interact with it. (**default** namespace)

Pods: Co-located group of containers that share an IP, namespace, storage volume.

A pod is a collection of application containers and volumes running in the same execution environment.

Most of the pod manifests are written using **YAML** or JSON scripts

Replica Sets: Manages the lifecycle of pods and ensures **specified number of containers is running.**

Service: Single, stable name for a set of pods, also acts as Load Balancer.

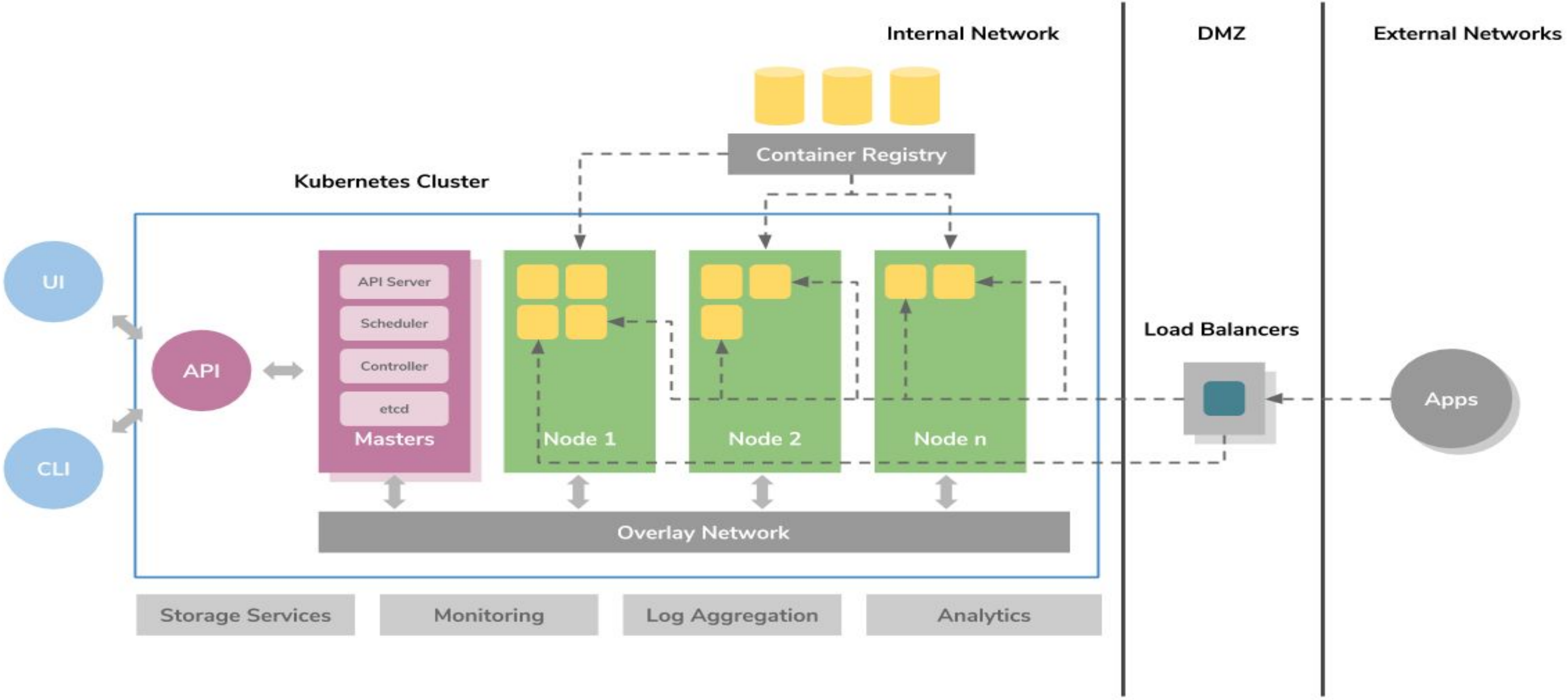
Label: Used to organize and select group of objects (Tags)

Ingress: These are objects that provide an easy-to-use front-end (externalised API surface area).

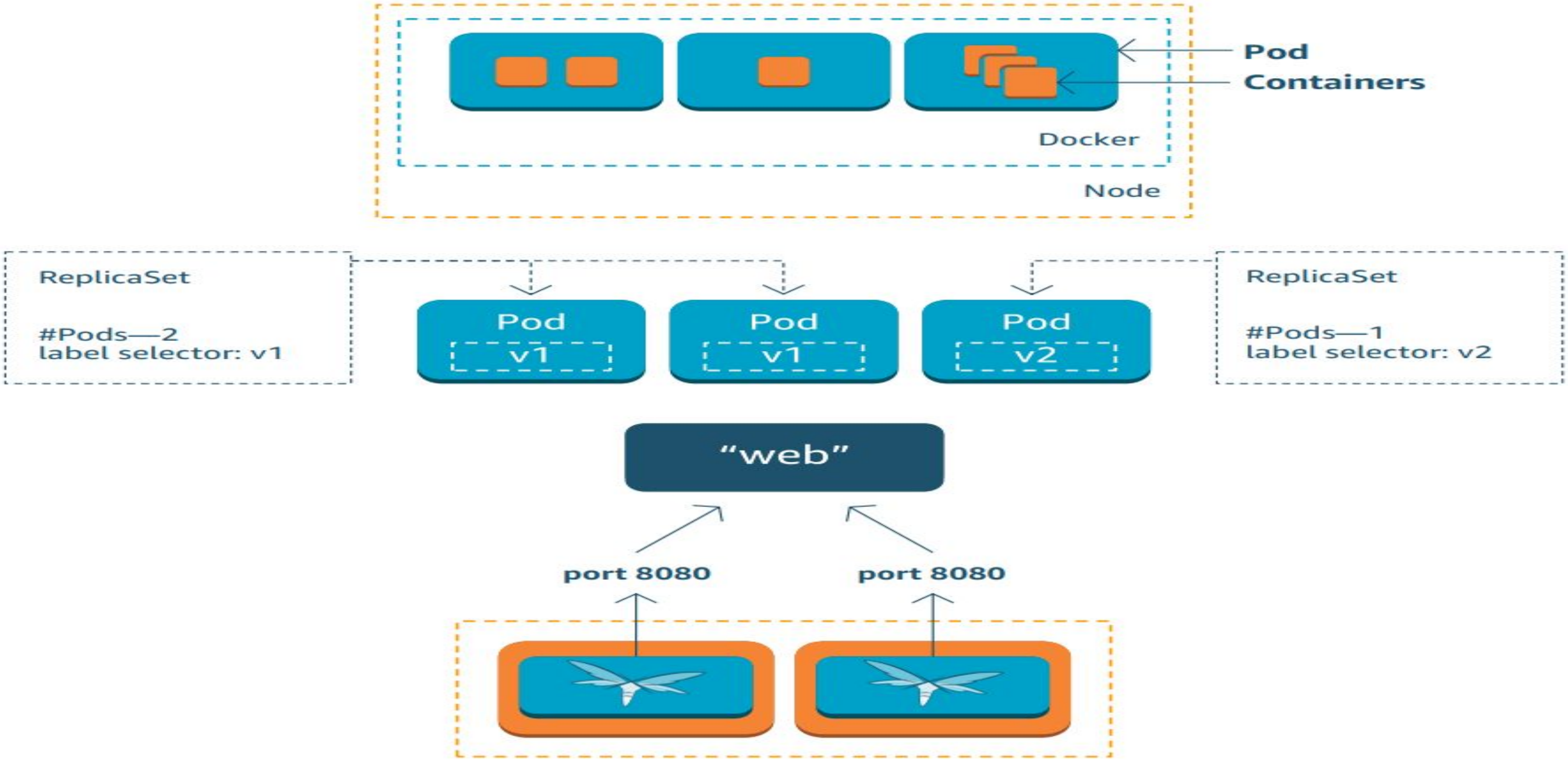
Kubernetes Cluster

- **Kubernetes coordinates a highly available cluster of computers that are connected to work as a single unit.**
- A Kubernetes cluster consists of two types of resources:
  - The **Control Plane** coordinates the cluster
  - **Nodes** are the workers that run applications

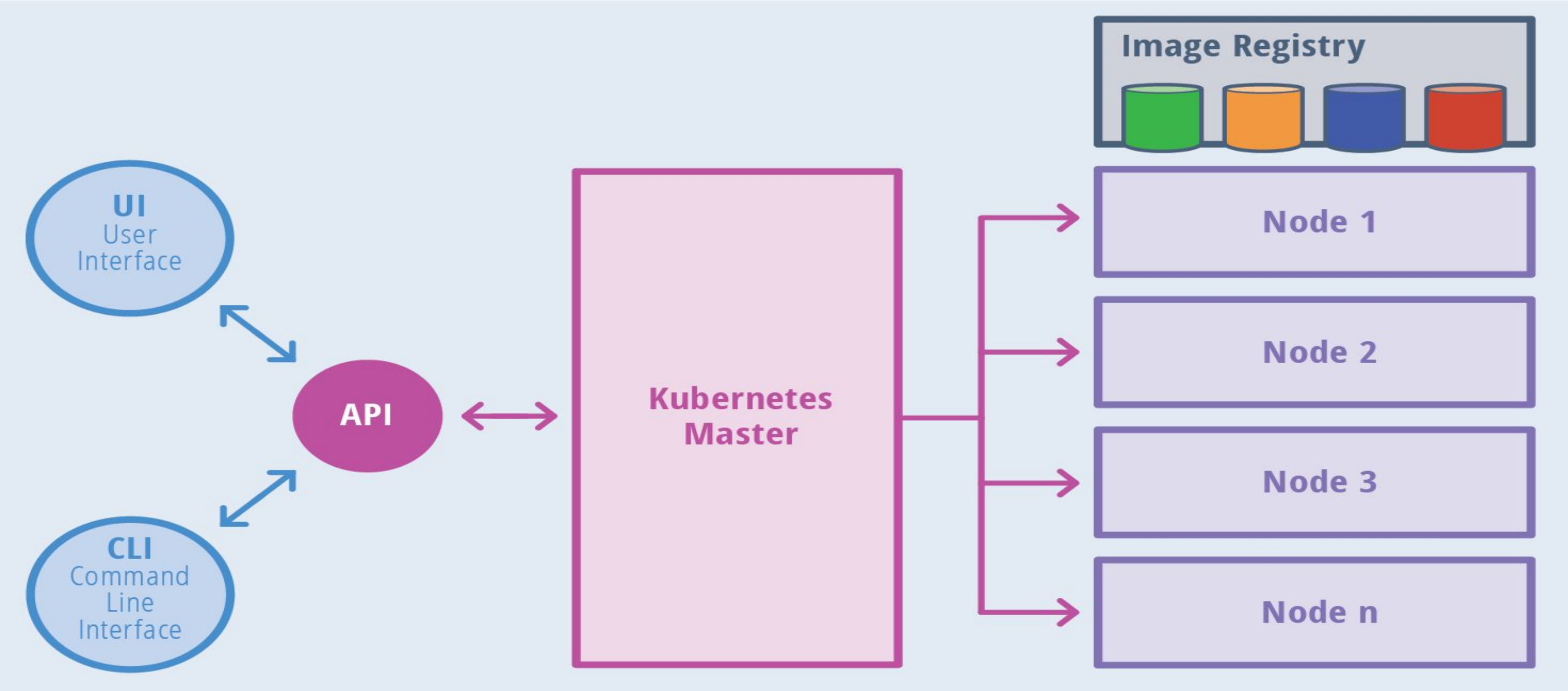
Kubernetes Architecture:



Kubernetes Components:



Kubernetes Architecture:



Managing containers using Kubernetes

Docker provides many features by exposing the underlying **'cgroups'** technology provided by the Linux kernel. With this, the following resource usage can be managed and monitored:

- Memory resources management and limitation
- CPU resources management and limitation
- Kubernetes also can be installed using **Minikube**, locally.
- **Minikube** is a simulation of the Kubernetes cluster, but the main function of this is for experimentation, local development or for learning purposes.
- https://kubernetes.io/docs/tutorials/hello-minikube/
- https://kubernetes.io/docs/tutorials/kubernetes-basics/create-cluster/cluster-intro/

- Use Commands here:
- https://kubernetes.io/docs/concepts/workloads/controllers/deployment

Container Image

- A container image is a binary package that encapsulates all of the files necessary to run an application inside an OS container.
- The Open Image (OCI) is the standard image format that’s most widely used. Container

Two Types of container categories are:

- **System containers**, which try to imitate virtual machines and may run the full boot processes
- **Application containers**, which run single applications.
- System containers, which try to imitate virtual machines and may run the full boot processes.
- Application containers, which run single applications.
- The default container runtime used by Kubernetes is **Docker**.

Kubernetes Architecture:

- Ability to deploy existing applications that run on VMs without any changes to the application code.

- On the high level, any application that runs on VMs can be deployed on Kubernetes by simply containerizing its components.

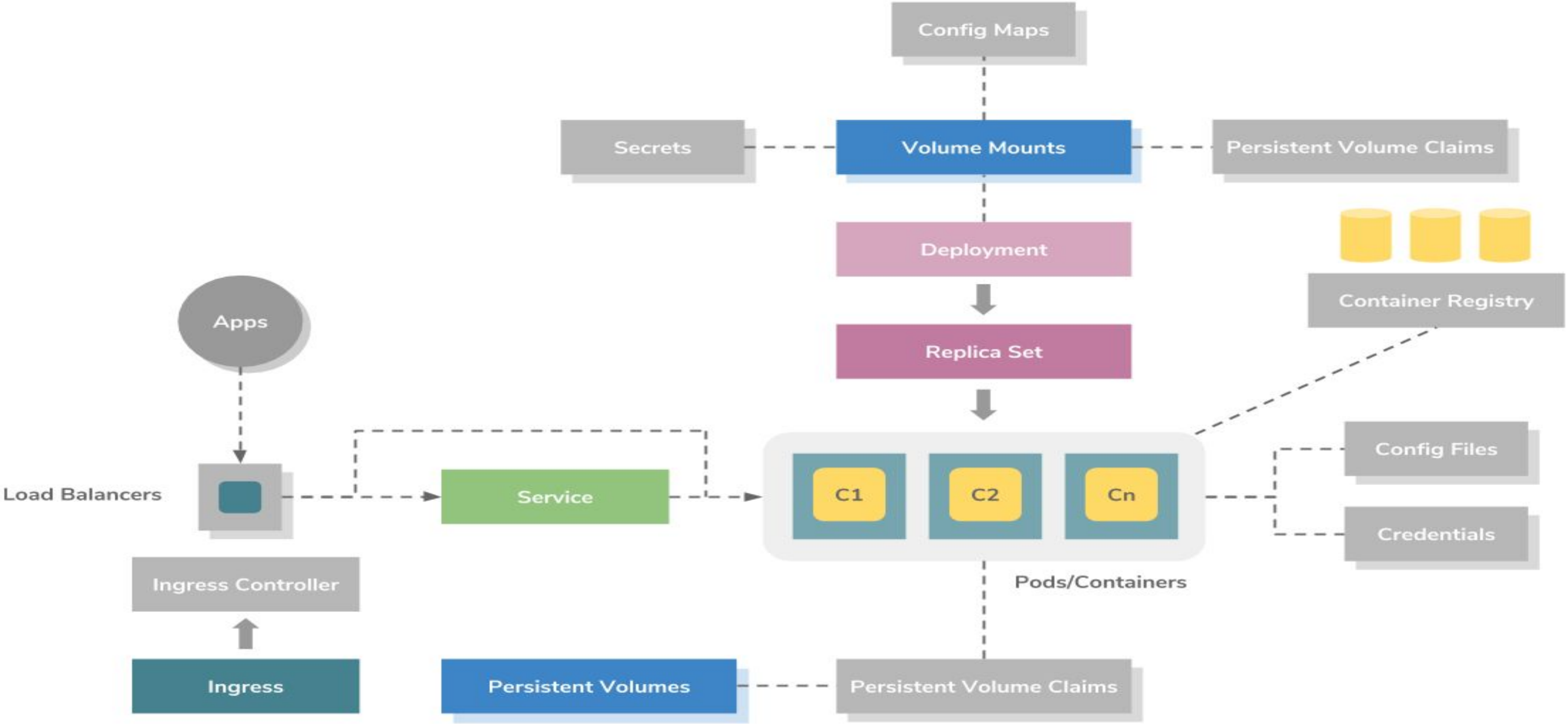
- This is achieved by its core features.

- Container grouping, container orchestration, overlay networking, container-to-container routing with layer 4 virtual IP based routing system, service discovery, support for running daemons, deploying stateful application components, and most importantly the ability to extend the container orchestrator for supporting complex orchestration requirements.

Kubernetes Architecture:

- On very high level Kubernetes provides a set of dynamically scalable hosts for running workloads using containers and uses a set of management hosts called masters for providing an API for managing the entire container infrastructure.
- The workloads could include long-running services, batch jobs and container host specific daemons.
- All the container hosts are connected together using an overlay network for providing container-to-container routing.
- Applications deployed on Kubernetes are dynamically discoverable within the cluster network and can be exposed to the external networks using traditional load balancers.
- The state of the cluster manager is stored on a highly distributed key/value store etcd which runs within the master instances.

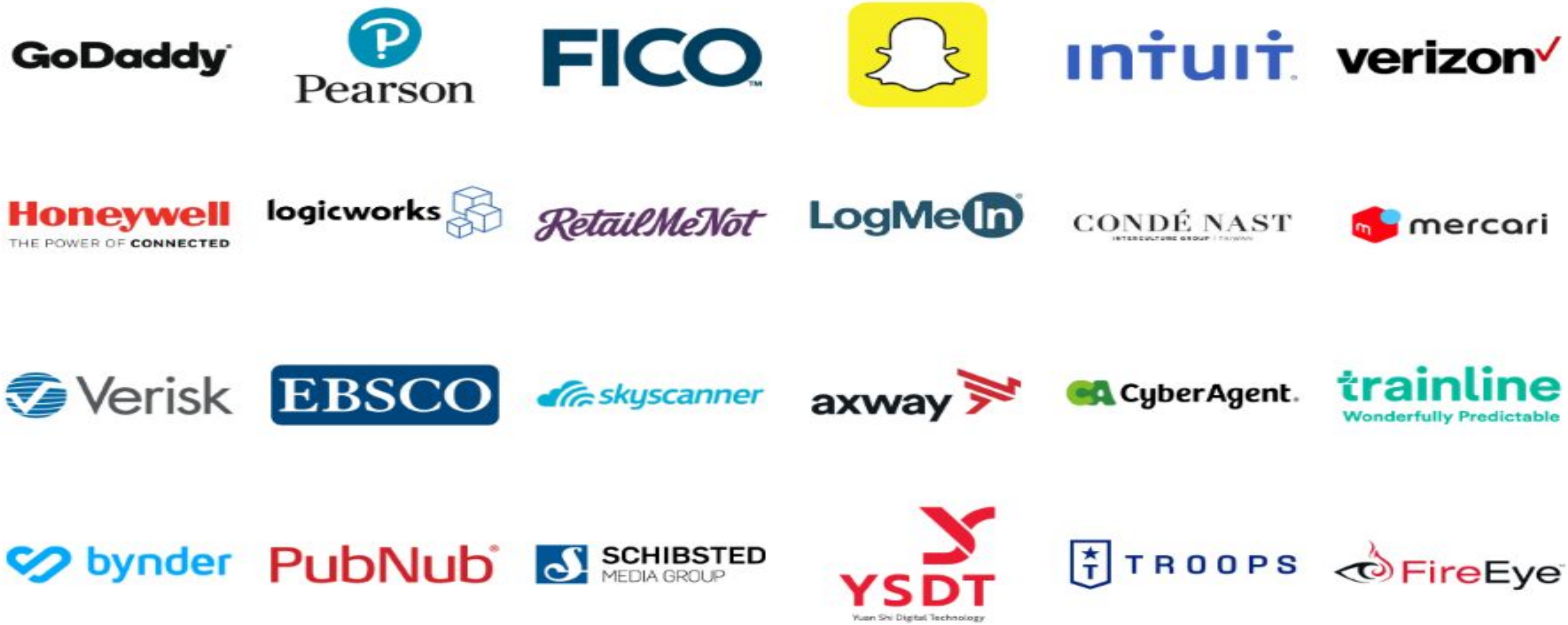
Application Deployment Model



Amazon EKS

- Amazon **Elastic Kubernetes Service** (Amazon EKS) makes it easy to deploy, manage, and scale containerized applications using Kubernetes on AWS.
- https://aws.amazon.com/eks/

Companies adopting Amazon EKS



DockerSwarm and Kubernetes

- Both Kubernetes and Docker Swarm are popular and used as **container orchestration platforms**.
- Docker Swarm is the native clustering for Docker.
- Originally, it did not provide much by way of container automation, but with the latest update to Docker Engine 1.12, container orchestration is now built into its core with first-party support.
- It takes some effort to get Kubernetes installed and running, as compared to the faster and easier Docker Swarm installation. Both have good scalability and high availability features built into them.
- Hence, one has to choose the right one based on the need of the hour.
- Do refer to https://www.upcloud.com/blog/docker-swarm-vs-kubernetes/

Reference:

https://kubernetes.io/docs/setup/production-environment/windows/intro-windows-in-kubernetes/

Kubernetes Cluster

- **Kubernetes coordinates a highly available cluster of computers that are connected to work as a single unit.**
- A Kubernetes cluster consists of two types of resources:
  - The **Control Plane** coordinates the cluster
  - **Nodes** are the workers that run applications

- **Kubernetes Proxy -** for routing network traffic for load balancing services (https://kubernetes.io/docs/getting-started-guides/scratch/)
- **Kubernetes DNS -** a DNS server for naming and discovery of the services that are defined in DNS
- **Kubernetes UI** - this is the GUI to manage the clusters