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 Nodes

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

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

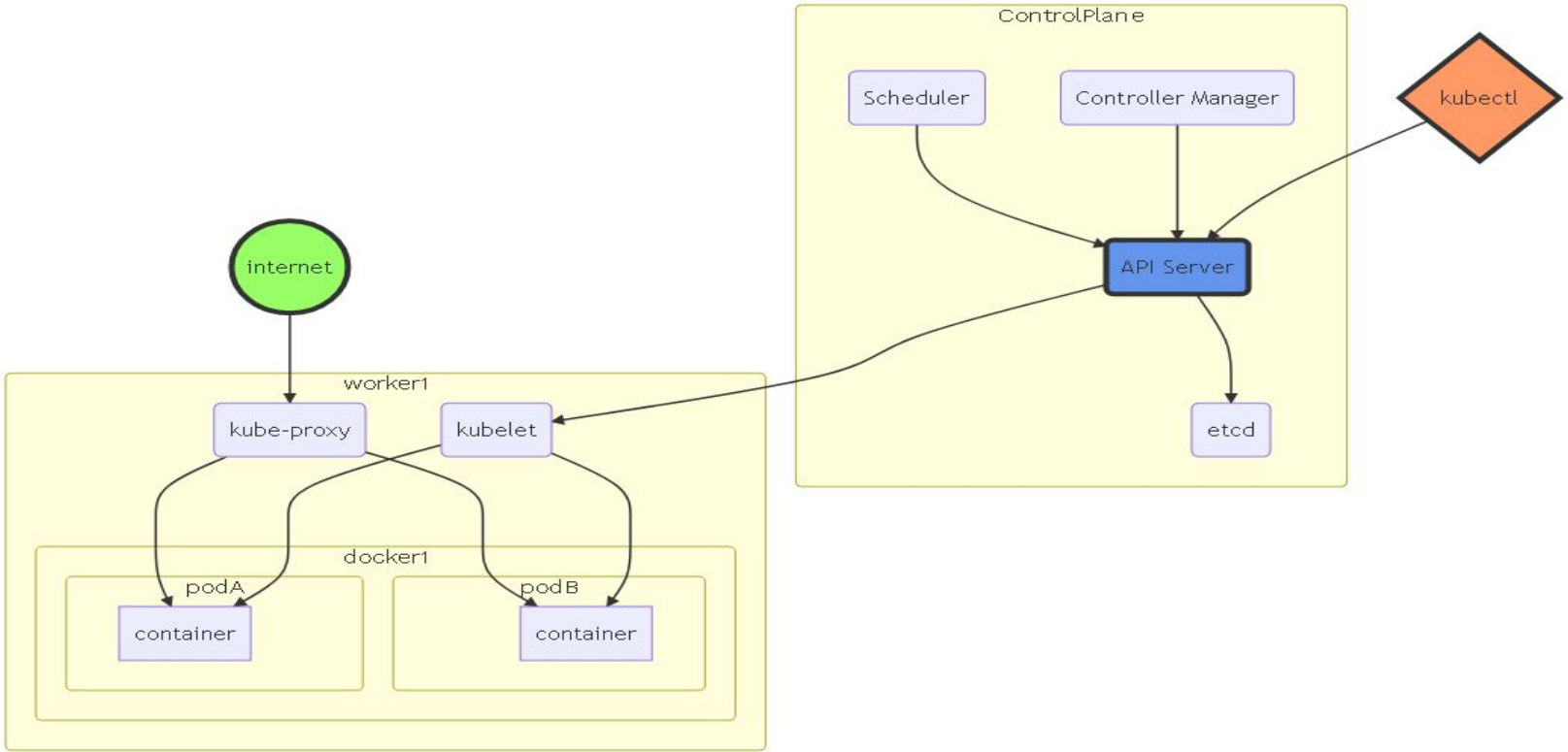
- Kube API Server
- controller
- scheduler
- etcd

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

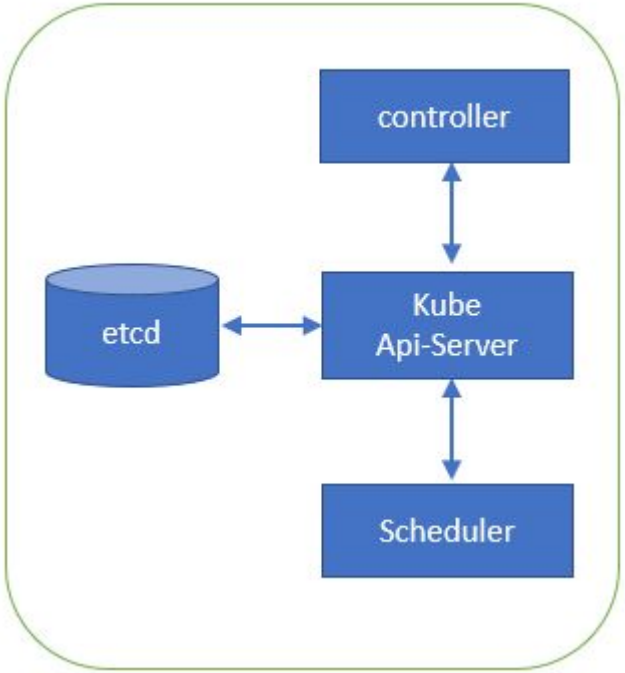
- kubelet
- kube-proxy
- Container runtime

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.

Kubernetes Architecture:

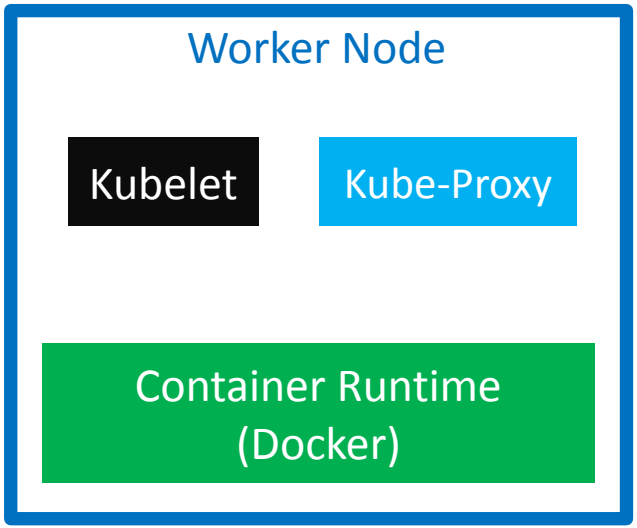


Master Node



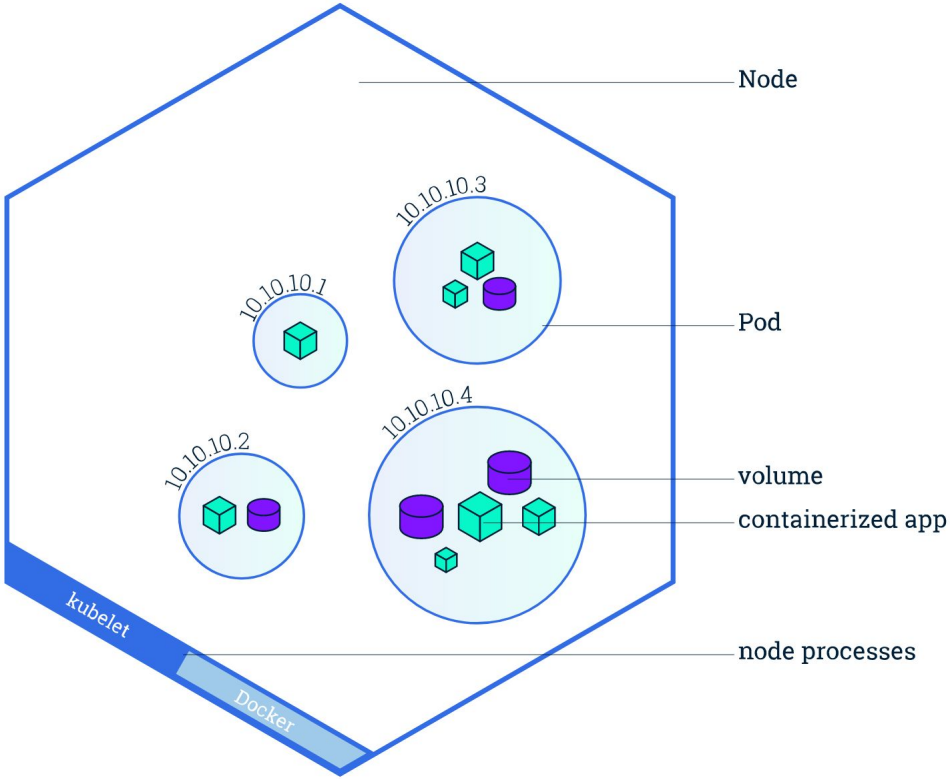
- **API Server** performs all the administrative tasks on the master node. A **user sends the rest commands to the API server,** which then validates the requests, then processes and executes them.
- **The scheduler,** schedules the work to different worker nodes. It has the **resource usage information** 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.** A **controller** is a loop that continually monitors your cluster and performs actions when certain events occur. The **controller manager** oversees all the controllers in your cluster. It starts their processes and ensures they're operational the whole time that your cluster's running.
- The **etcd** is a **key-value database store** that stores the state of the **cluster,node or pods**. It is also used to store the **configuration** details such as the **ConfigMaps** (store environment variables) and Secrets
  - **etcd** saves the resulting state of the cluster as a **key-value** store.

Worker Node

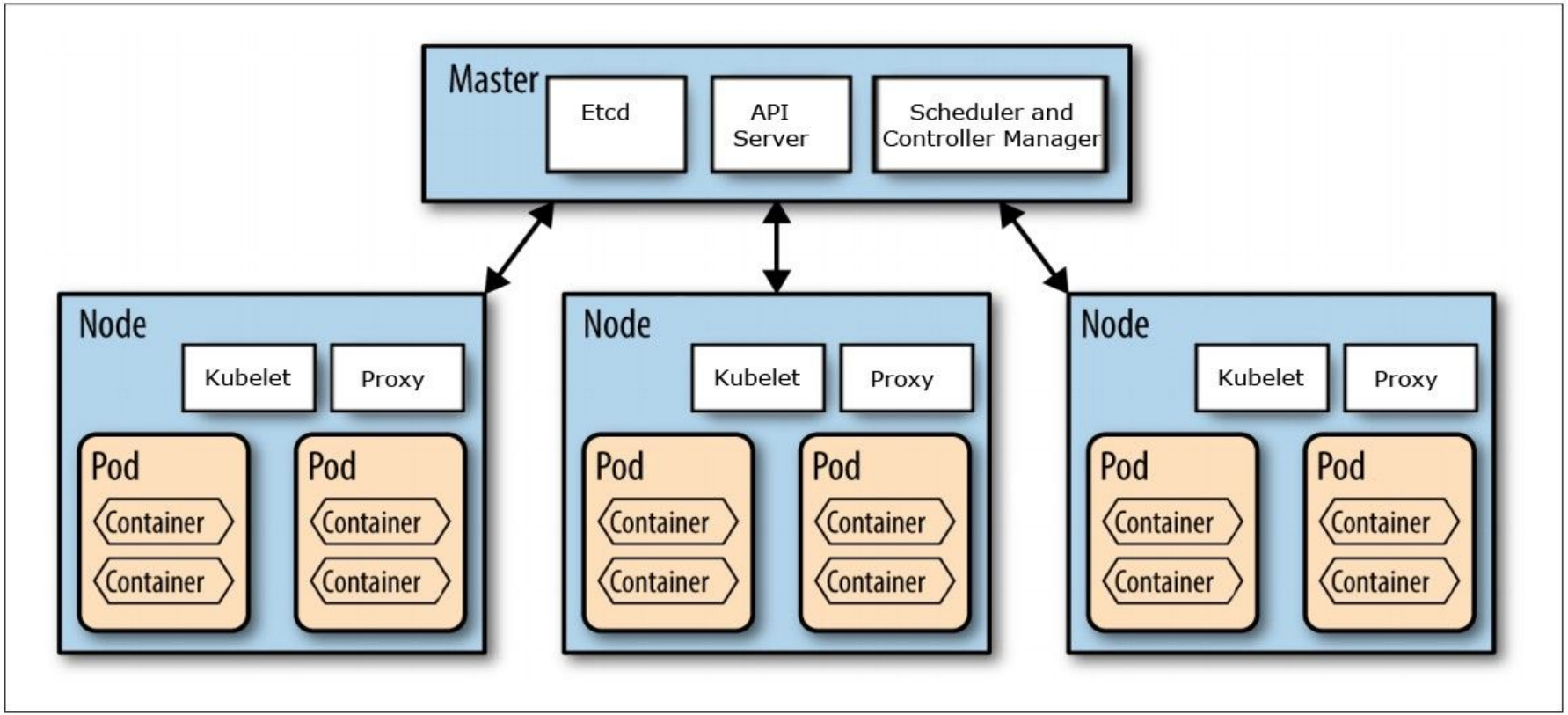


- **Kubelet** is basically an **agent/daemon** that runs on **each worker node** and **communicates with the master node**. This agent is responsible for making sure that **containers are running in a Pod on a node.**
- The **container runtime** is basically used to run and manage the continuous life cycle of the container on the worker node.
- **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.

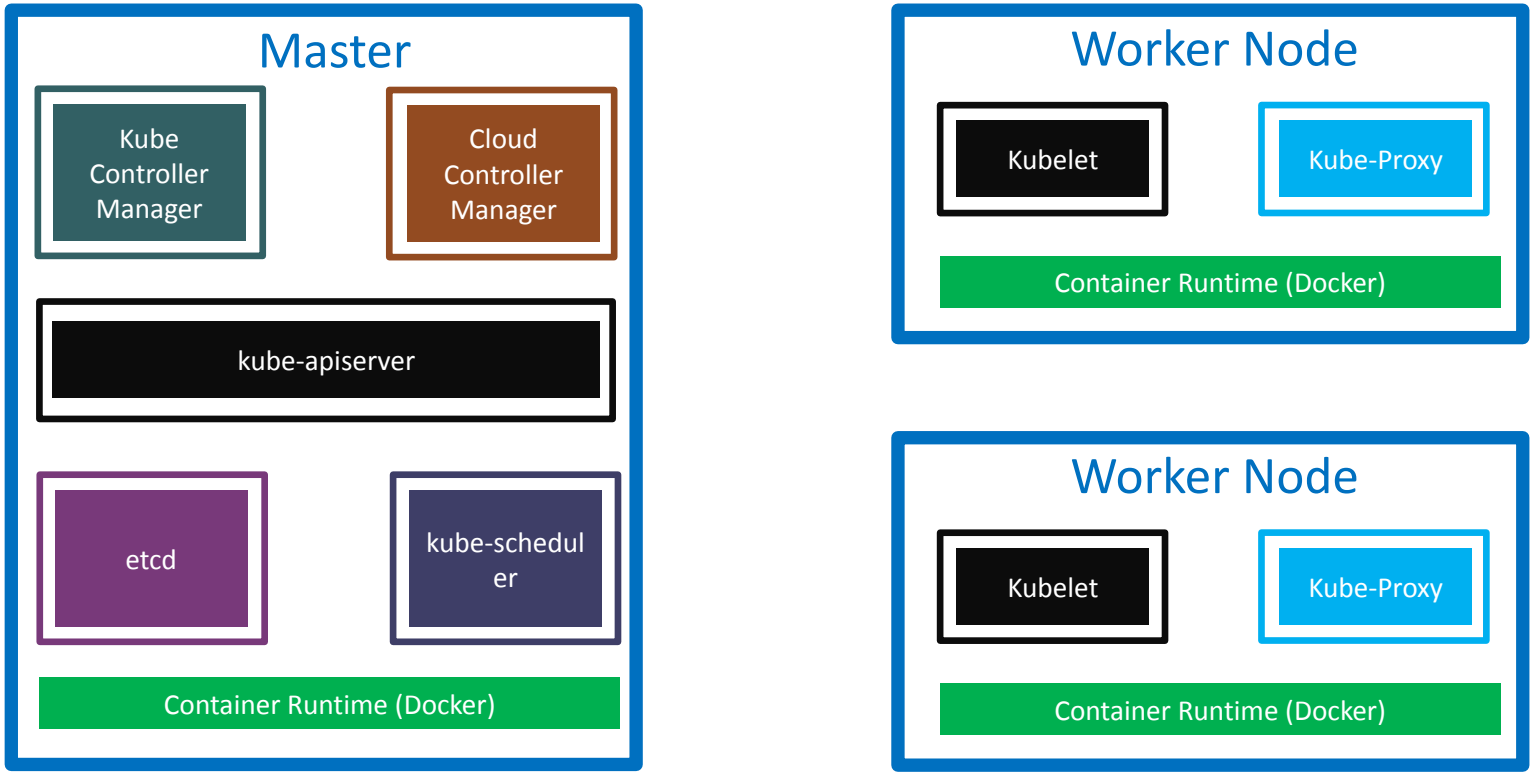
Worker Node


Node
10.10.10.3
10.10.10.1
Pod
10.10.10.4
10.10.10.2
volume
containerized app
kubelet
Docker
node processes

Kubernetes Architecture:



Kubernetes Architecture:



Kubernetes Objects/Components:

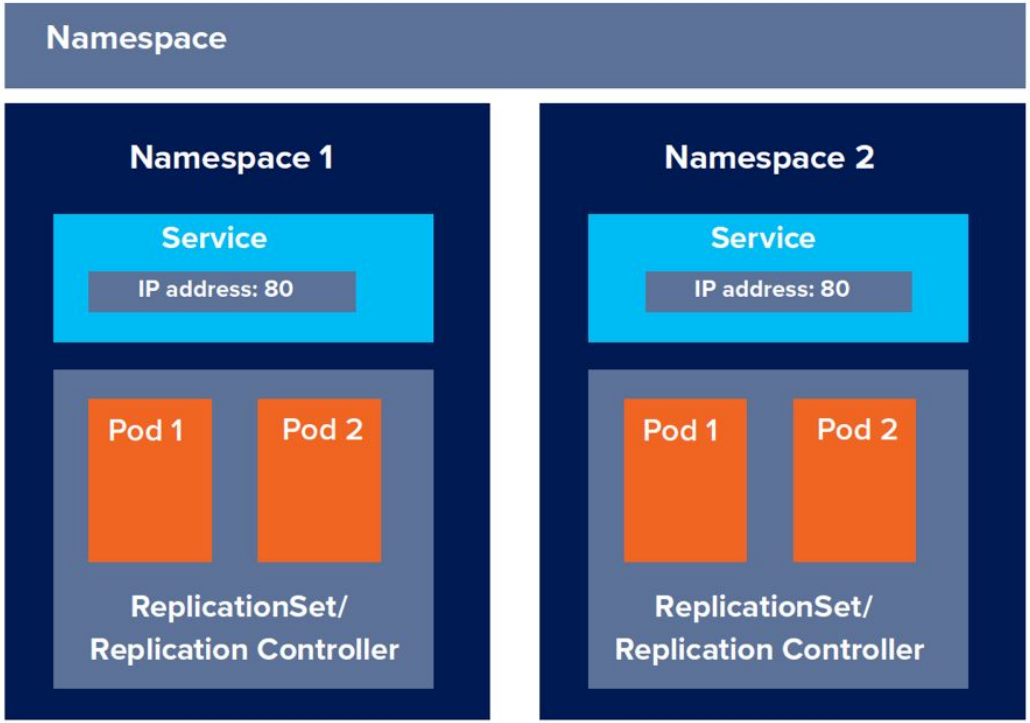
Pods: Co-located group of containers that share an **IP, namespace, storage volume**.
A pod is a collection of **application containers** and volumes running in the same **execution environment**. Most of the **pod manifests** are written using **YAML** or JSON scripts
Replica Sets: maintains a stable set of replica Pods running at any given time.
Manages the lifecycle of pods, and **If any Pod (Container) goes down, RS launches another Pod.**
Deployment: A **Deployment runs multiple replicas of your application containers** and automatically replaces any **containers** that fail or become unresponsive. **Rollout & rollback images changes to applications**. Deployments are well-suited for **stateless** applications.
Label: Key/Value pairs used for association and filtering. Also select group of objects (**Tags**), **Selectors** is used to select specific K8s Object using the Label key:values.
Service: Maps a fixed IP address to a logical group of pods.
Ingress: These are objects that provide an easy-to-use front-end (externalised API surface area).

Kubernetes Components:

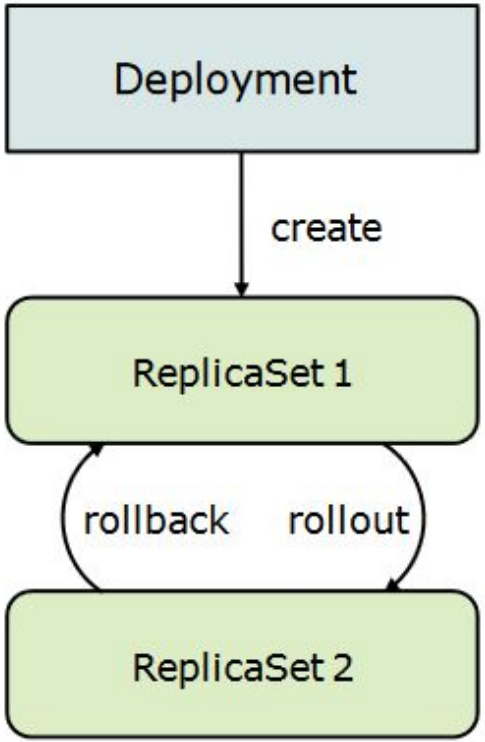
NameSpaces: It is used to remove name collision within a cluster. It supports **multiple virtual clusters** on the same **physical cluster**. This provides isolation and complete access to control the degree to which other services interact with it. (**default** namespace). Namespaces provides a mechanism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

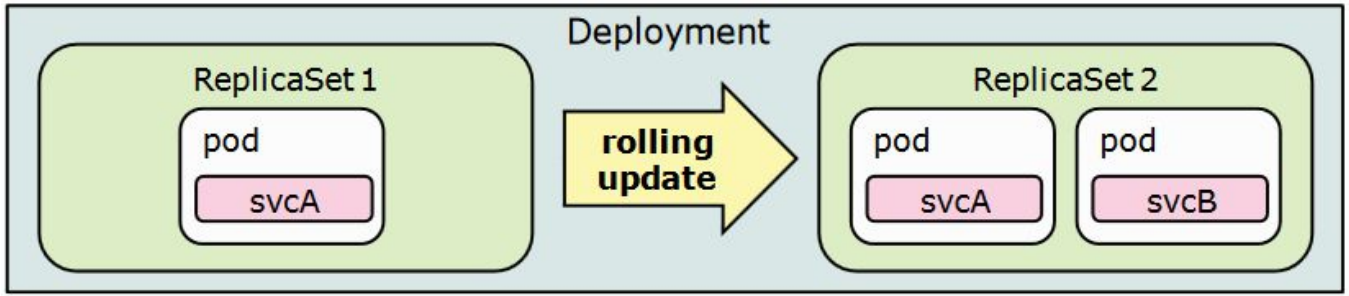
- In a new cluster, Kubernetes automatically creates the following namespaces: `**default**` (for user workloads) and for the Kubernetes **control plane**: `**kube-system**`.
- **default**
  - All pods will be launched in this default namespace.
  - **kubectl get pods -n default**
- **kube-system (do not make any changes to this namespace)**
  - All master components daemons will be running here
  - **kubectl get pods -n kube-system**

Kubernetes Components:



Kubernetes Components: Rolling update

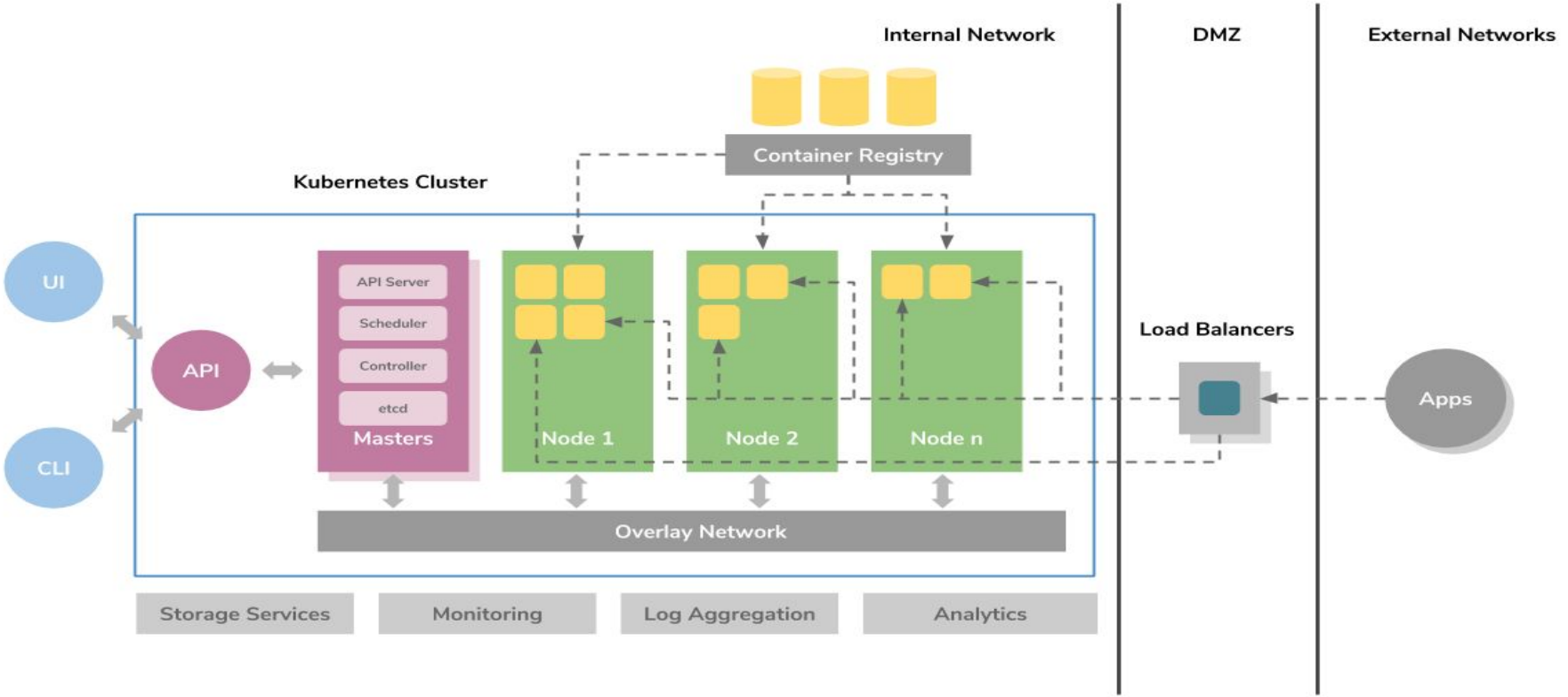




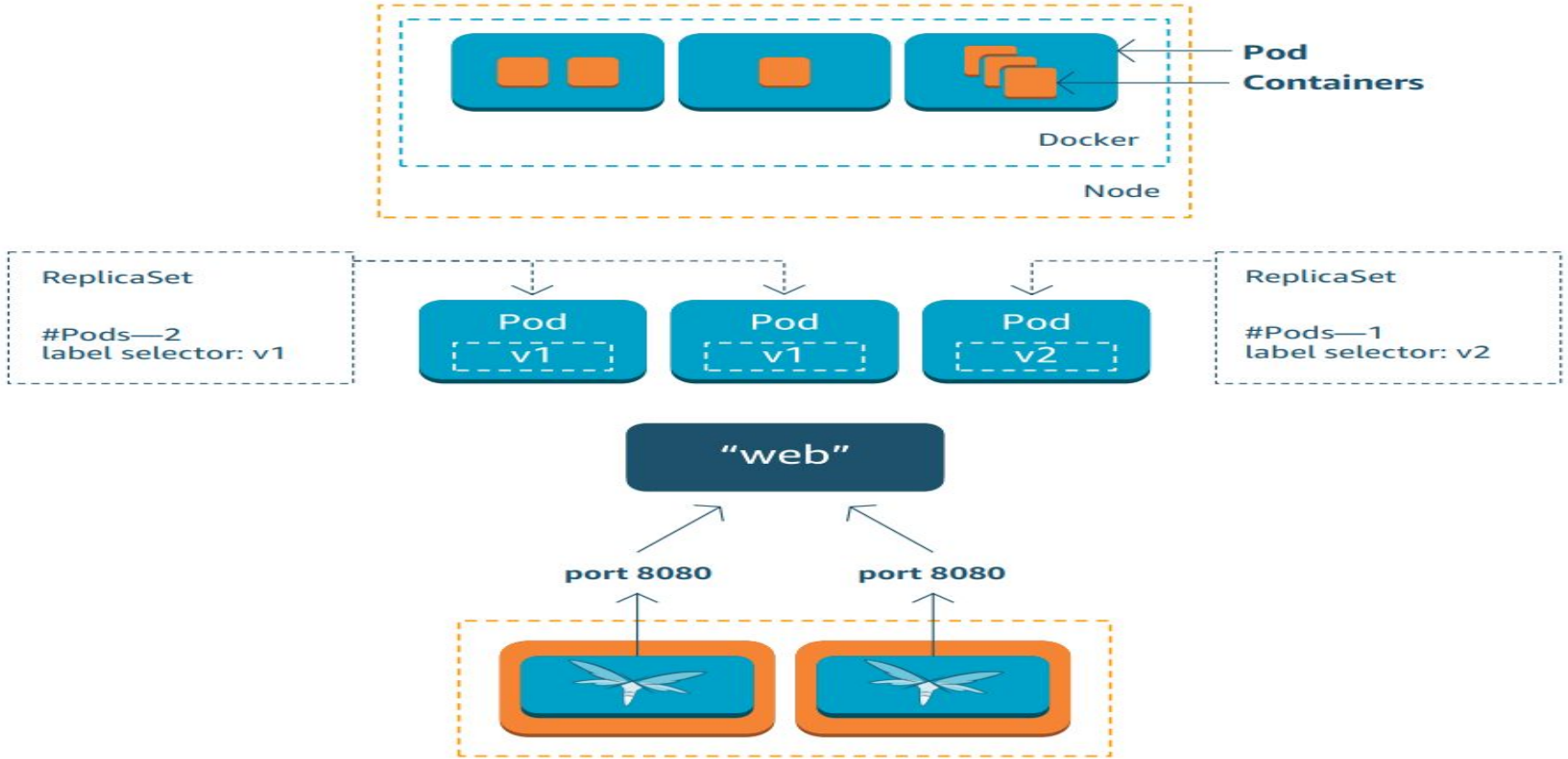
Kubernetes Cluster

- **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

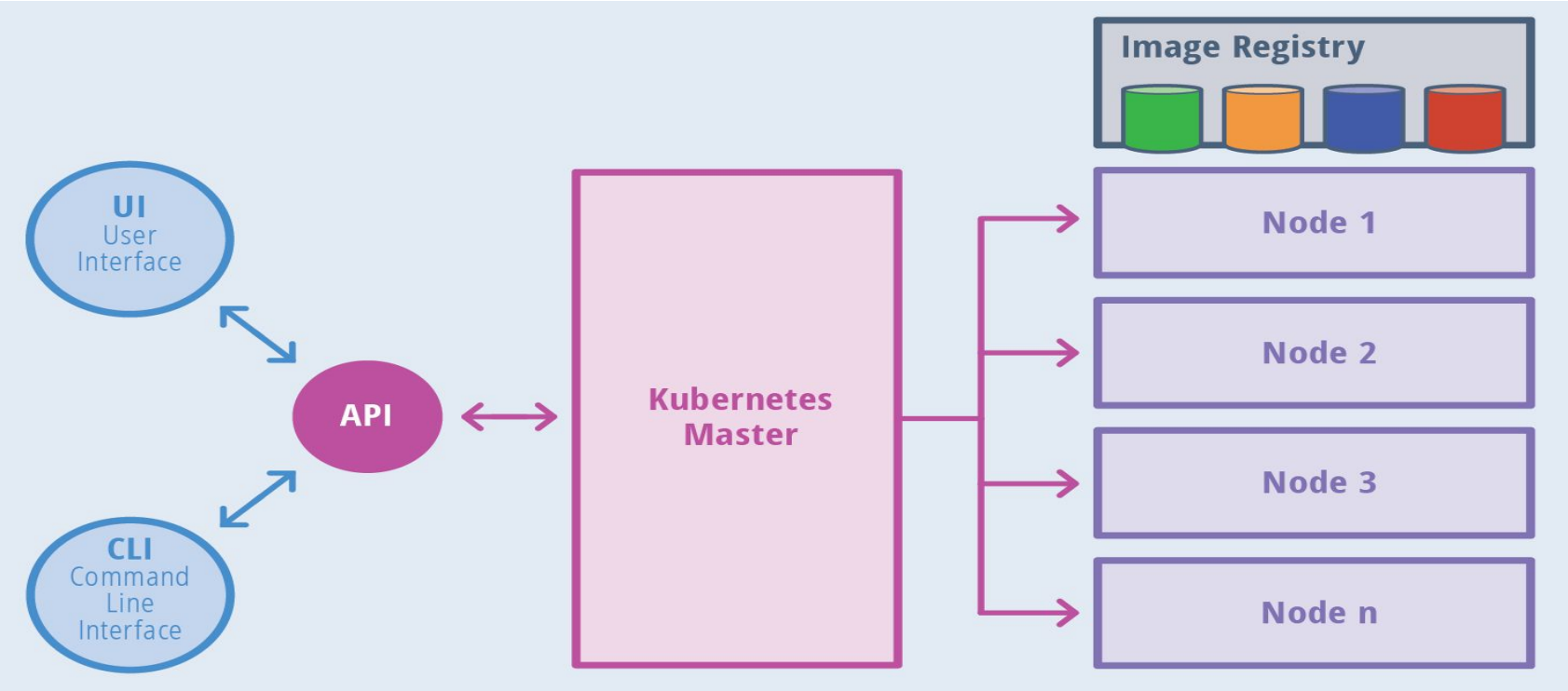
Kubernetes Architecture:



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:

- 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.**
- **Minikube is a single node kubernetes cluster**
- https://kubernetes.io/docs/tutorials/hello-minikube/
- https://kubernetes.io/docs/tutorials/kubernetes-basics/create-cluster/cluster-intro/

- Example Commands here:
- https://kubernetes.io/docs/tasks/run-application/run-stateless-application-deployment/
- 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 (**kube-system**)
  - All master components are pods under this namespace.
- **Application containers**, which run single applications. (**default**)
- 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 Service

- You can expose an application running on a set of PODs using different types of Services available in k8s.
- **ClusterIP**
  - Used for communication between applications inside k8s cluster
- **NodePort**
  - To access our application **outside of k8s cluster**, you can use **NodePort** service.
  - Exposes the Service on each **Worker Node's IP** at a static port (nothing but **NodePort**).
  - Port Range **30000-32767**
- **LoadBalancer**
  - Primarily for Cloud Providers to integrate with their Load Balancer services (Example: AWS Elastic Load Balancer)

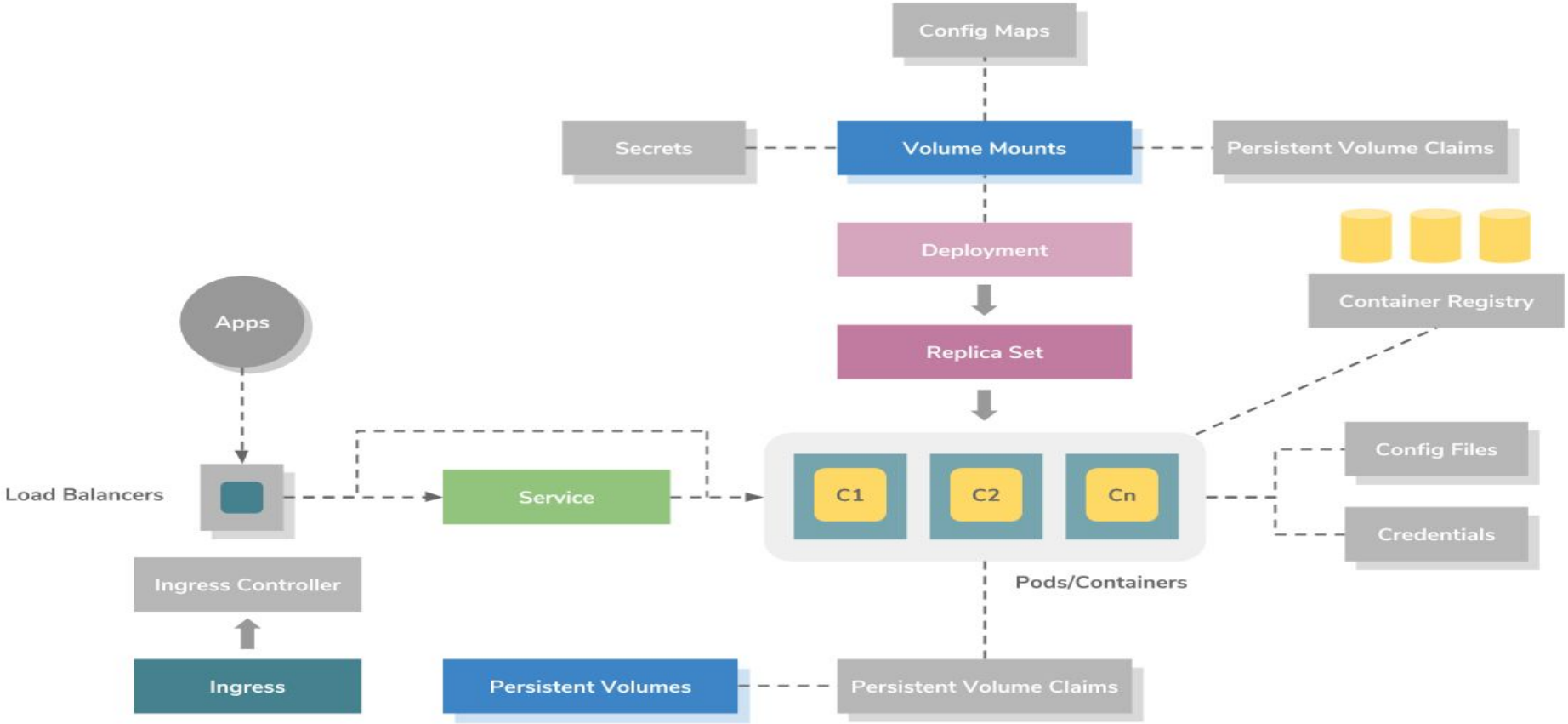
Kubernetes Architecture:

- 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.
- 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.

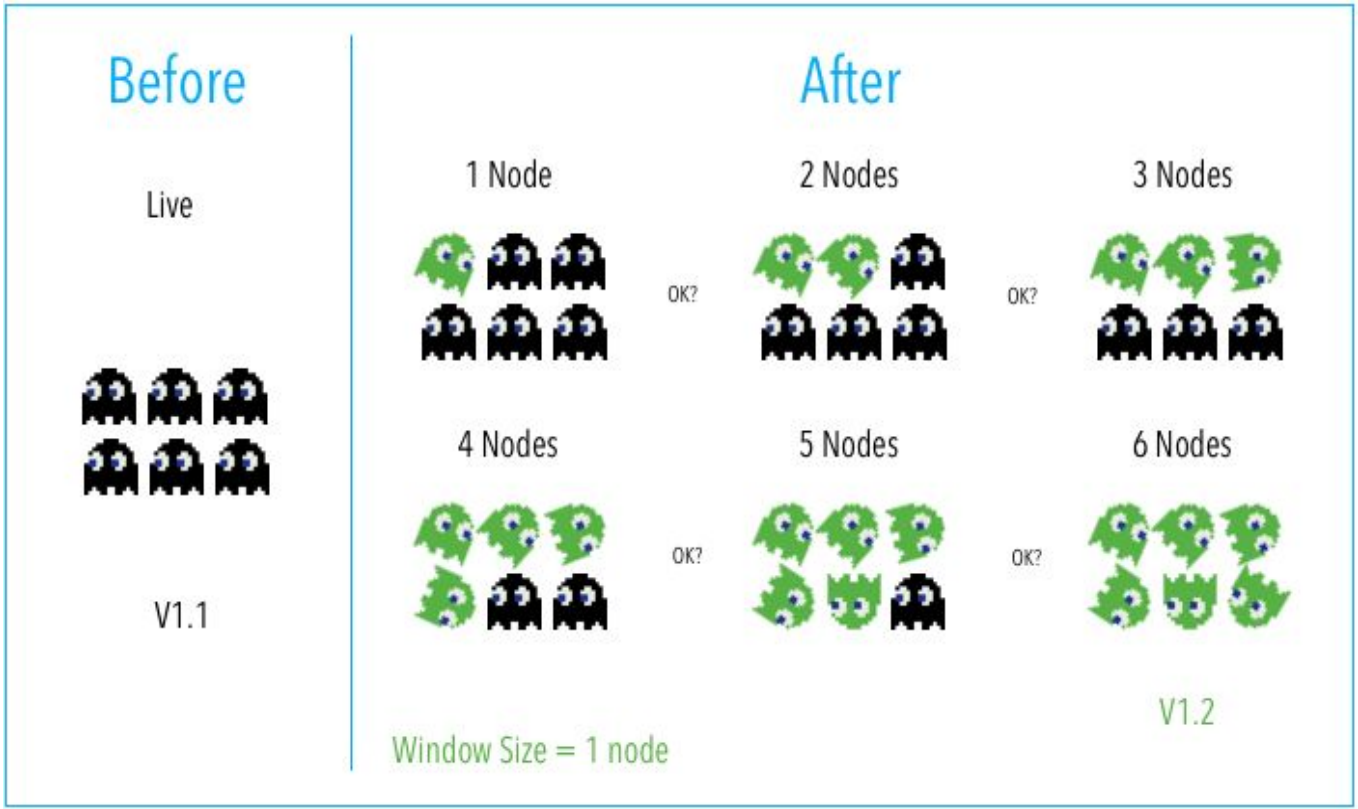
Kubernetes Architecture:

- 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



Rolling Updates in K8s



A rolling deployment is a deployment strategy that updates running instances of an application with the new release. All nodes in a target environment are incrementally updated with the service or artifact version in integer N batches.

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/

  - **https://aws.amazon.com/eks/pricing/**
    - **Pay $0.10 for your master node**
    - **Pay underlying EC2 instance cost on the basis of type**

Companies adopting Amazon EKS

Companies adopting Amazon EKS

- https://www.infoworld.com/article/3664052/why-mercedes-benz-runs-on-900-kubernetes-clusters.html
- https://aws.amazon.com/solutions/case-studies/
  - Filter by Domain to view all AWS Case Studies
- https://aws.amazon.com/solutions/case-studies/itv-case-study/
- https://aws.amazon.com/solutions/case-studies/netflix/

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/

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

Managing containers using Kubernetes

- **Memory resources management and limitation**
- **CPU resources management and limitation**

- Running a Stateless application
  - https://kubernetes.io/docs/tasks/run-application/run-stateless-application-deployment/
- Running a Stateful application
  - https://kubernetes.io/docs/tasks/run-application/run-single-instance-stateful-application/