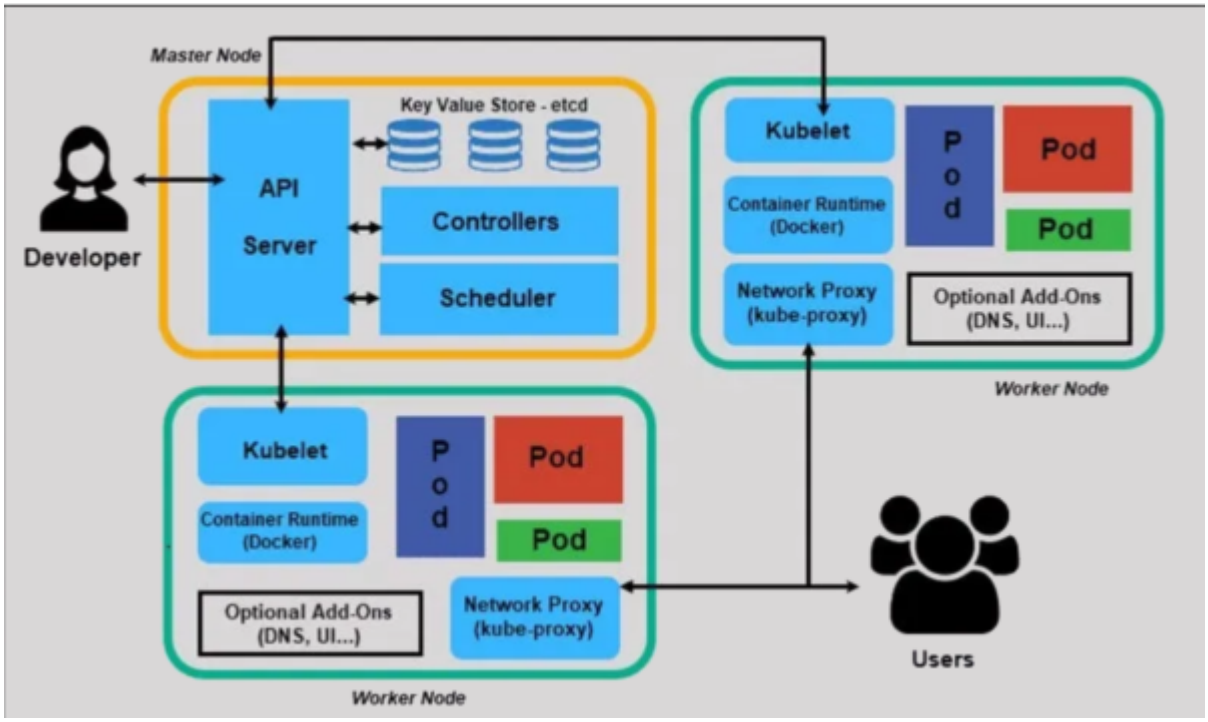
Basics of Kubernetes (K8s)

What is Kubernetes (K8s)?

Kubernetes is a platform to orchestrate the deployment, scaling, and management of container-based applications. The core functionality is scheduling workloads in containers across your infrastructure. Some other capabilities of Kubernetes are:

> Providing authentication and authorization, Debugging applications, Monitoring and Logging, Rolling updates, scaling Cluster, Balancing loads, Naming and service discovery, Distributing secrets, Mounting storage systems

Kubernetes is a big open source project and ecosystem which came out of Google, but joined the Cloud Native Computing Foundation (CNCF) and is now the de facto standard in the space of container-based applications. According to the 2021 CNCF survey, 96% of organizations use or evaluate Kubernetes.



What means (container) orchestration?

The primary responsibility of Kubernetes is container orchestration. That means:

1. Making sure that all the containers that execute various workloads are scheduled to run on (physical or virtual) machines.
2. Containers must be packed efficiently following the constraints of the deployment environment and the cluster configuration.

3. Monitoring all running containers and replace dead, unresponsive, or otherwise unhealthy containers.

Components of K8s

Nodes

At the core of a Kubernetes cluster are the nodes, which can be physical or virtual machines. Each node runs a container runtime, such as Docker, and a kubelet agent responsible for managing containers.

Control Plane Components

The control plane manages and orchestrates the entire Kubernetes cluster. It consists of several components working together to maintain the desired state of the system. The primary control plane components are as follows:

- `etcd`: A distributed key-value store that stores the cluster's configuration data and state.
- `API Server`: Acts as the central management point and exposes the Kubernetes API to clients for cluster communication.
- `Scheduler`: Determines which pods are assigned to nodes based on resource availability and other scheduling policies.
- `Controller Manager`: Includes various controllers responsible for maintaining desired cluster state, handling tasks such as scaling, replication, and node management.

Pod

A pod is the fundamental unit of deployment in Kubernetes. It represents a group of one or more containers, with shared storage and network resources, and a specification for how to run the containers. A Pod has a few specific characteristics:

- All containers for a pod run on the same node.
- Any container running within a pod will share the node's network.
- Containers within a pod can share files through volumes, attached to the containers.
- A pod has an explicit life cycle (Pending, Running, Succeeded, Failed, or Unknown), and will always remain on the node in which it was started.

When you want to know what's running on a Kubernetes cluster, you are generally analyzing the Pods running within Kubernetes and their state.

K8s service and load balancing

Kubernetes services provide a stable network endpoint for accessing a set of pods. They act as load balancers, distributing incoming traffic across the pods for improved availability and scalability.

Namespaces

Namespaces provide a logical partitioning mechanism within a cluster, allowing teams to isolate and manage their resources independently. They enable better organization, security, and resource allocation.

Namespaces can be used to provide quotas and limits around resource usage, have an impact on DNS names that Kubernetes creates internal to the cluster. If no namespace is specified when interacting with Kubernetes through `kubectl`, the command assumes you are working with the default namespace (named `default`).

Deployment and ReplicaSets

Deployments are used to manage the lifecycle of pods and ensure the desired state of an application. They create and update ReplicaSets, which are responsible for maintaining a specified number of pod replicas.

How to use K8s successfully?

One of the keys to successfully using Kubernetes is to consider how you want to operate and structure your code so that it fits cleanly into Pods and Containers. Structuring your software solutions to break problems down into components that operate with constraints by Kubernetes, you can easily take advantage of parallelism and container orchestration to use many machines as seamlessly as you would use a single machine.