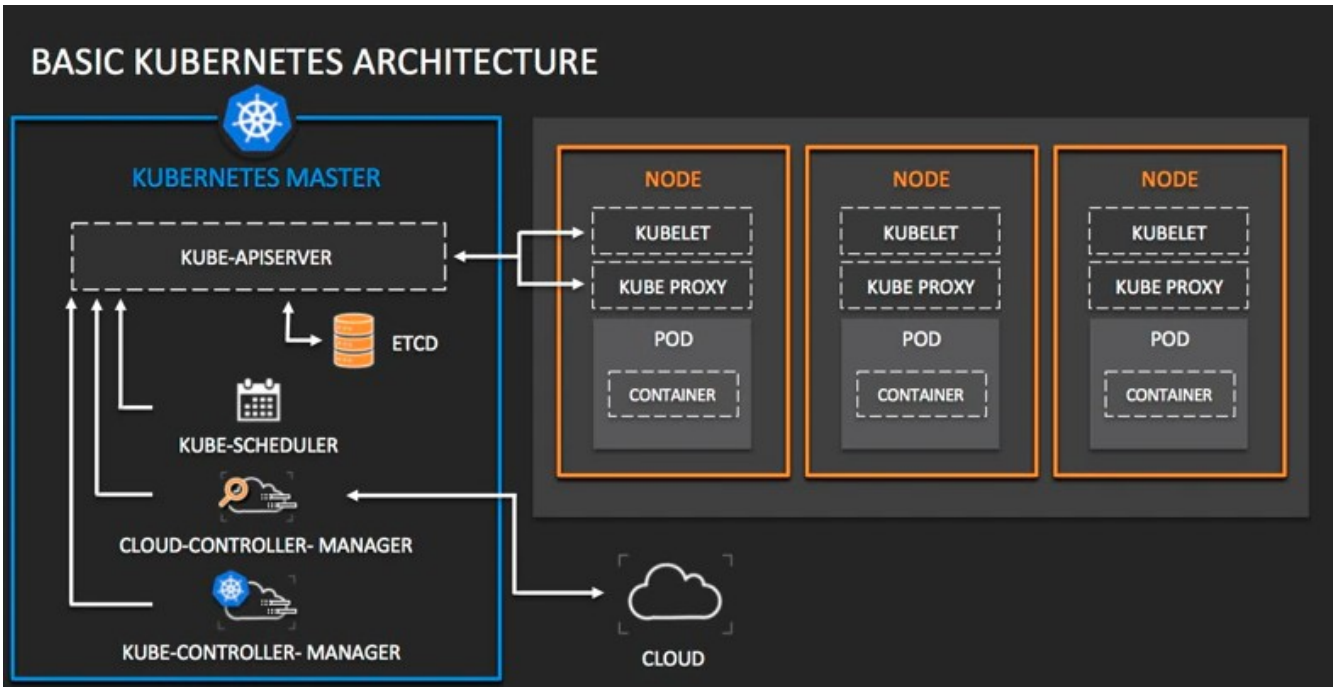
PART 1: Introduction

► What is Kubernetes?



Kubernetes is at its heart, a container orchestration tool.

It's responsible for allocating and scheduling containers, and then providing abstracted functionality like internal networking and file storage. It also includes monitoring the health of all of these elements and stepping in to repair or adjust them as necessary.

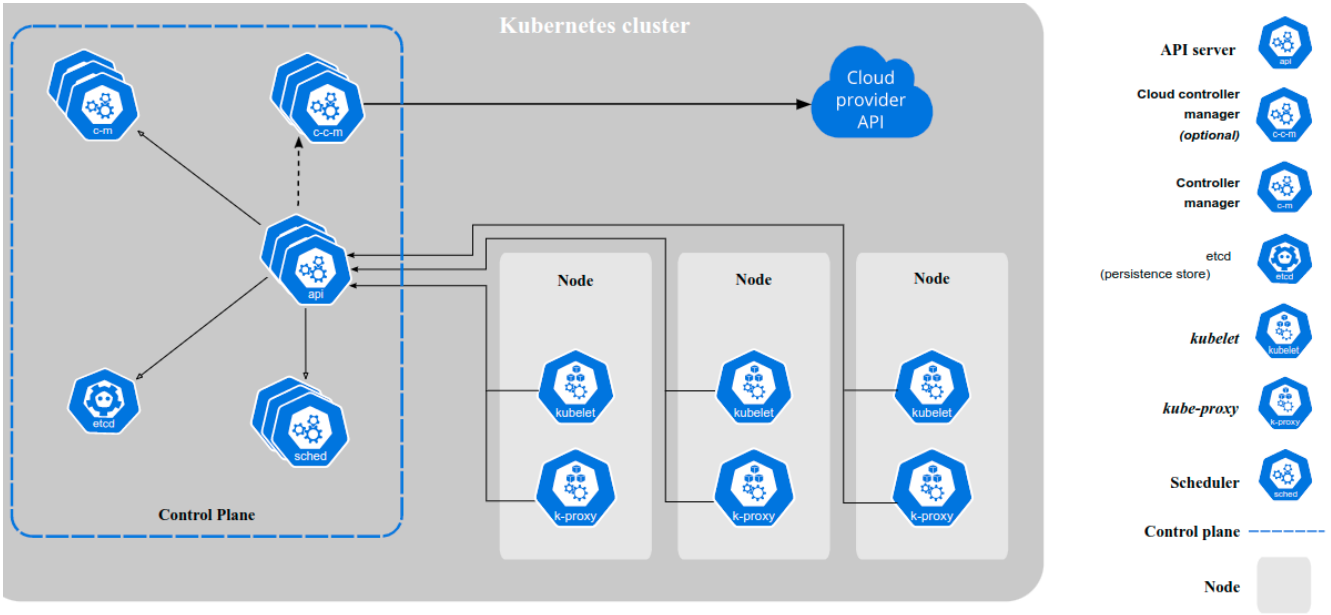
In short, it's all about abstracting the how, when, and where containers are run.

► **What problems does Kubernetes solve?**

Kubernetes provides an orchestration tool that maintains consistency, speed, and elastic horizontal scaling of containers.

There's also a ton of standardization, tooling, and vendor support from cloud providers for kubernetes.

► **Basic Components**



Containers are often represented as container images in Docker to establish the software or application requirements for a pod.
Pods are the smallest, most basic unit of kubernetes. It will encapsulate the running application and is ephemeral (automatically replaces failed pods). It is comprised of one or more containers and shares network/storage resources.

Nodes are essentially a pod "fleet" or pool of worker machines.
Controllers define the desired state of the cluster. They're exposed as a workspace API object, which creates and configures pods to maintain a healthy state. If the health or state of a pod changes, the controller responds.

Clusters consist of the components that represent the control plane and multiple nodes. When you deploy Kubernetes, you get a cluster, which has at least one worker node and IP.

► Services and Configmaps

Services are resources (IP, DNS, Routing, LoadBalancing) you create to make a single, constant point of entry and this point of entry is always available to a group of pods. Each service has an IP address and port that never changes while the service exists. Clients can open connections to that IP and port, and those connections are then routed to one of the pods backing that service. This way, clients of a service don’t need to know the location of individual pods providing the service, allowing those pods to be moved around the cluster at any time.

Configmaps are for separating configuration from deployment. Without configmaps you'd have to download and redeploy your containers every time you wanted to change the configuration. This K8s primitive is intended for defining the configuration of the apps deployed to Kubernetes. Briefly, your config is a dictionary of settings represented by key-value pairs. They are stored in YAML, and a K8s resource called ConfigMap is responsible for handling them.

► Volumes

Kubernetes doesn't provide data persistence out of the box, which means when a pod is re-created, the data is gone. You need to create and configure the actual physical storage and manage it by yourself.

The 1st component "Persistent Volume" is a cluster resource, like CPU or RAM, which is created and provisioned by administrators.

The 2nd component "Persistent Volume Claim" on the other hand is a user's or pod's request for a persistent volume.

With the 3rd component "Storage Class" you can dynamically provision Persistent Volume component and so automate the storage provisioning process.

► ReplicaSet and Deployments

ReplicaSets allow us to define the number of replicas for a pod. Replicas maintain healthy pods by replacing those that fail automatically.

Deployments manage the roll out of ReplicaSets and control the transition of new pods running different versions.

► Worker Nodes and Control Plane Node (Heart of the Cluster)

The worker node(s) host the Pods that are the components of the application workload. The control plane manages the worker nodes and the Pods in the cluster. This is the part of the cluster that actually executes the containers and applications on them.

The control plane or master node, is responsible for providing the API that is used to configure and manage resources within the Kubernetes cluster. It implements functions for operations, monitoring, and is the access point for cluster administration. This is what CLI communicates with for operating the cluster.

► Scheduler

Responsible for managing the workloads throughout the cluster and assigning pods to nodes. Its main function is to assign newly created pods in the cluster to a feasible worker node as per the optimal resource requirements of the containers inside the pod.

► Controller Manager

The Cloud Controller Manager can be described as three different things looking at it from a high level view:
A binary
A number of control loops
Part of the glue between k8s and your cloud

► etcd (Brain of the Cluster)

Kubernetes uses etcd to store all of its data – its configuration data, its state, and its metadata. Kubernetes is a distributed system, so it needs a distributed data store like etcd. Etcd lets any of the nodes in the Kubernetes cluster read and write data. It's a consistent and highly-available key value store used as Kubernetes' backing store for all cluster data.