# Kubernetes

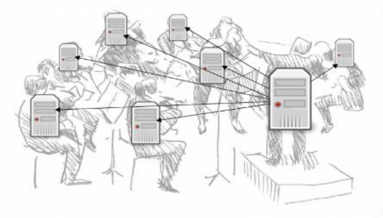
## What is Kubernetes?

- Open source container orchestration tool.
- Developed by Google.
- Helps manage containerized applications in Different deployment environments.

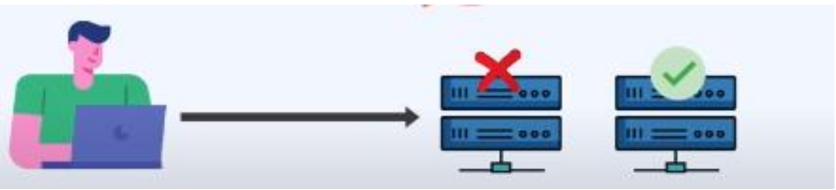
## What problem did Kubernetes solve?

Kubernetes was introduced to solve many major issues like:
**Better infrastructure**.
**Better management through modularity**.
**Deploying and updating software at scale**.

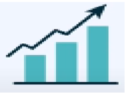
## What features do orchestration tool offer?

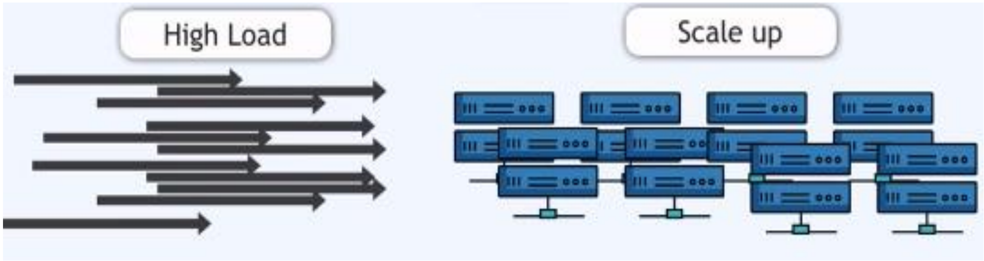


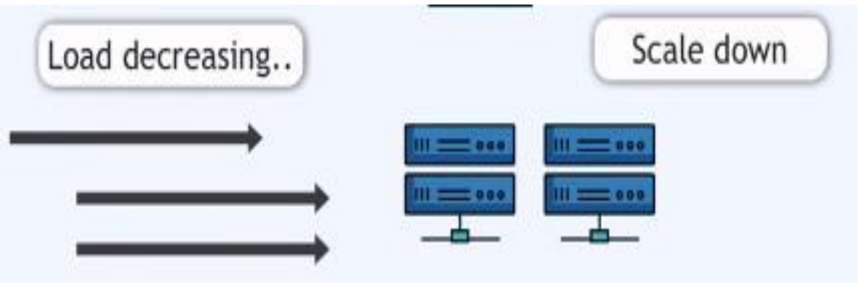
- High Availability or no downtime.



- Scalability or high performance



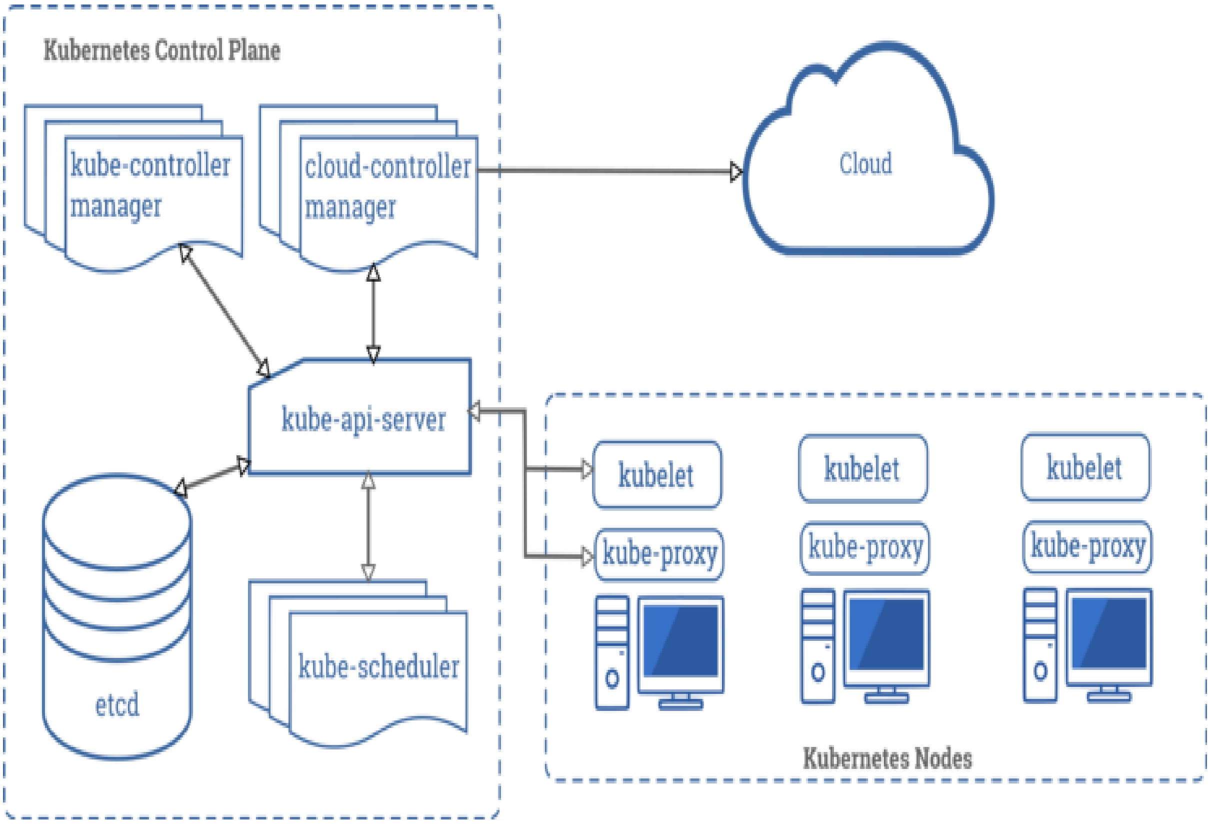




- Disaster Recovery- backup and restore.

## Kubernetes architecture

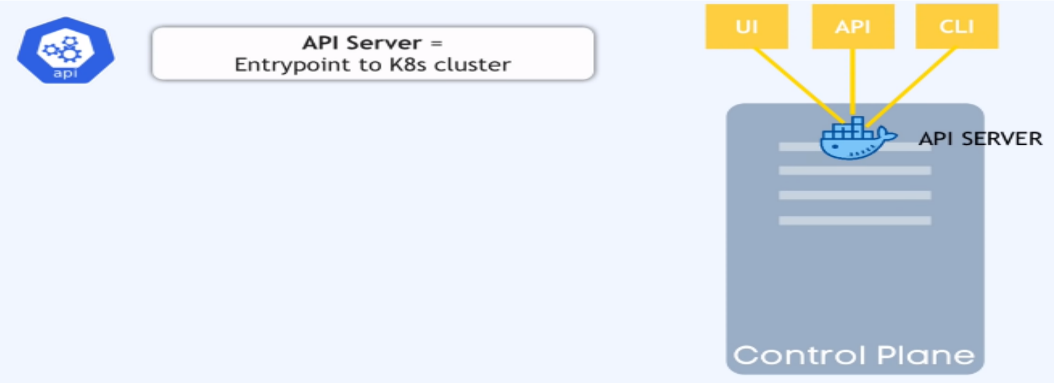
Looking inside *Kubernetes* from the architecture point of view we can see two main parts: ***control plane*** and ***worker nodes***.



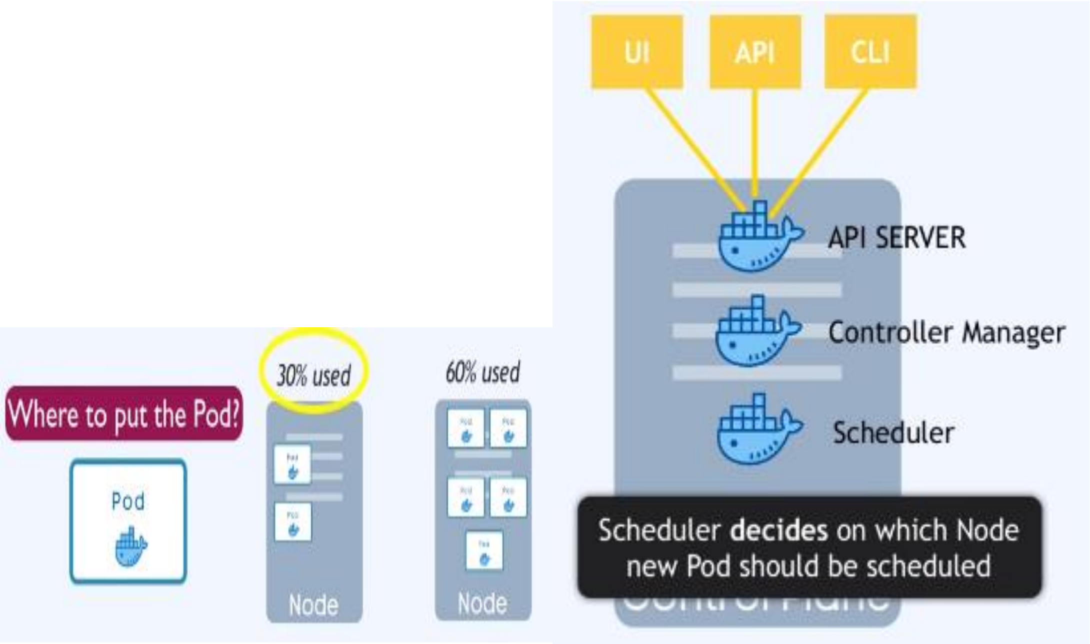
***Control plane*** decides on which worker node run each container, checks health state of a cluster, provides an API to communicate with cluster and many more. If one of the nodes will go down and if some containers were running on that broken machine, it will take care of rerunning those applications on other nodes.

**Inside control plane we can find several, smaller components:**

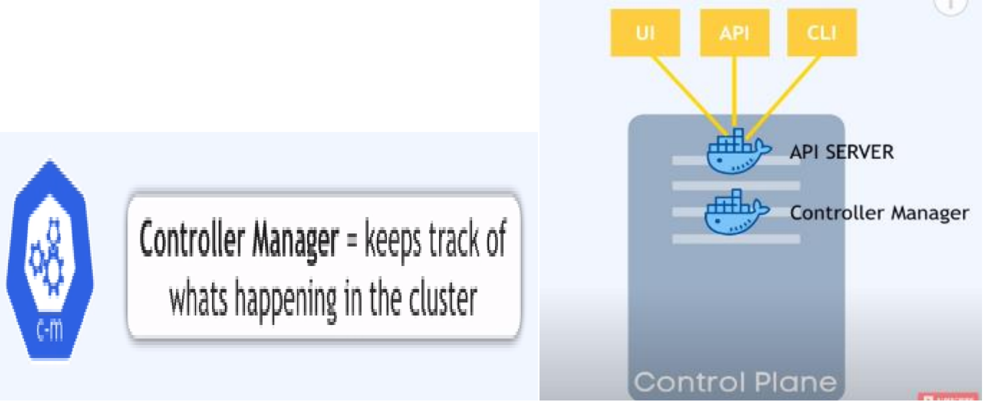
- ***kube-api-server*** — it's responsible for providing an API to a cluster, it provides endpoints, validates requests and delegates them to other components,



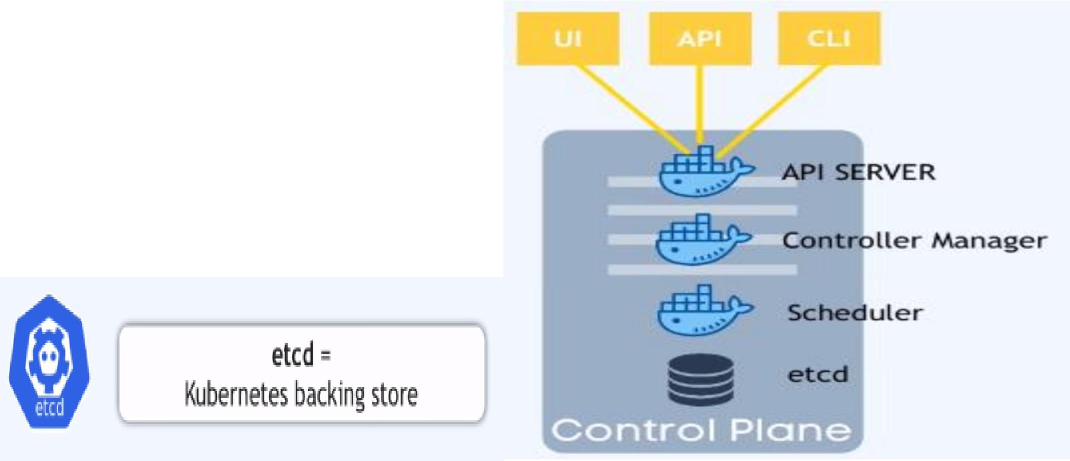
- ***kube-scheduler*** — constantly checks if there are new applications (*Pods*, to be specific, the smallest objects in K8s, representing applications) and assign them to nodes,



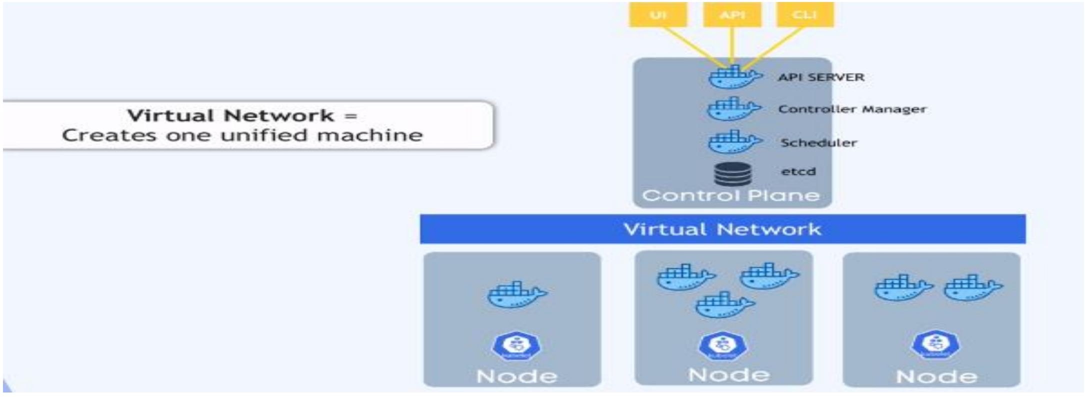
- ***kube-controller-manager*** — contains a bunch of controllers, which are watching a state of a cluster, checking if a desire state is the same as current state and if not they communicate with *kube-api-server* to change it; this process is called control loop and it concerns several *Kubernetes* objects (like nodes, Pod replicas and many more); for each *K8s* object there is one controller which manages its lifecycle,


c-m
Controller Manager = keeps track of whats happening in the cluster
UI
API
CLI
API SERVER
Controller Manager
Control Plane

- ***etcd*** — it's a reliable key-value store database, which stores configuration data for the entire cluster,



***cloud controller manager*** — holds controllers that are specific for a cloud providers, it's available only when you use at least one cloud service in a cluster.



***DNS***- It enables applications inside the cluster to be able to communicate with each by specific (human-readable) names, and not IP addresses.

***work order nodes*** Apart from the *control plane* each *Kubernetes cluster* can have one or more *work order nodes* on which application are running. To integrate them with K8s each one of them has:

- ***kubelet*** - is responsible of managing *Pods* inside the node and communicating with control plane (both components talk with each other when a state of a cluster needs to be changed),
- ***Kube-proxy*** -take care of networking inside a cluster, make specific rules etc.

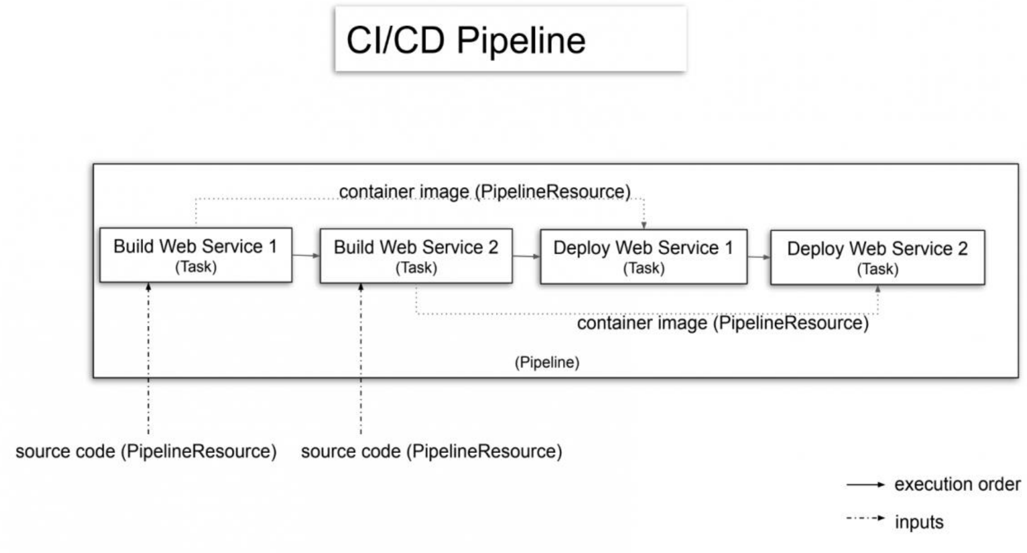
## Kubernetes Objects

- K8s provides an abstraction of an infrastructure. And to interact with a cluster we need to use some kind of the interface that will represent a state of it. And these are the Kubernetes objects, all of them represent a state of entire system.
- **Pods** —Pods are the smallest Kubernetes objects that represents an application. Pods are not containers. They're wrapper for one or more containers.
- **Deployments** — are responsible for a life cycle of Pods. They take care of creating Pods, upgrading and scaling them.

- **Services**— take care of networking tasks, communication between Pods inside a cluster. And every time the IP address can change so other Pods inside cluster would need to constantly update addresses of all depended applications (service discovery). Services take care of load balancing a traffic between those Pods.
- **Persistent Volumes** — provide an abstract way for data storage, which could be required by Pods (e.g. to save some data permanently or in cache).
- **ConfigMaps**— they holds key-value data that can be injected to Pods, for example as an environment variable, which allows to decouple an application from its configuration.

## Why we use k8s in Tekton?

- It is CI/CD service tool
- Tekton is a Knative-based framework for CI/CD pipelines,
- Tekton stores everything related to a pipeline as custom resources (CRs) within the cluster


CI/CD Pipeline
container image (PipelineResource)
Build Web Service 1 (Task)
Build Web Service 2 (Task)
Deploy Web Service 1 (Task)
Deploy Web Service 2 (Task)
container image (PipelineResource)
(Pipeline)
source code (PipelineResource)
source code (PipelineResource)
execution order
inputs