**Kubernetes in Detail**

* Kubernetes is an open-source container management tool that automates container deployment, scaling & load balancing.
* It schedules, runs, and manages isolated containers that are running on virtual/physical/cloud machines.
* All top cloud providers support Kubernetes.
* One popular name for Kubernetes is K8s.

**⬤ History**

* Google developed an internal system called ‘borg’ (later named Omega) to deploy and manage thousands of google applications and services on their cluster.
* In 2014, google introduced Kubernetes an open-source platform written in ‘Golang’ and later donated to CNCF (Cloud Native Computing Foundation).

**⬤ Online platform for K8s**

* Kubernetes playground
* play with K8s
* Play with Kubernetes classroom

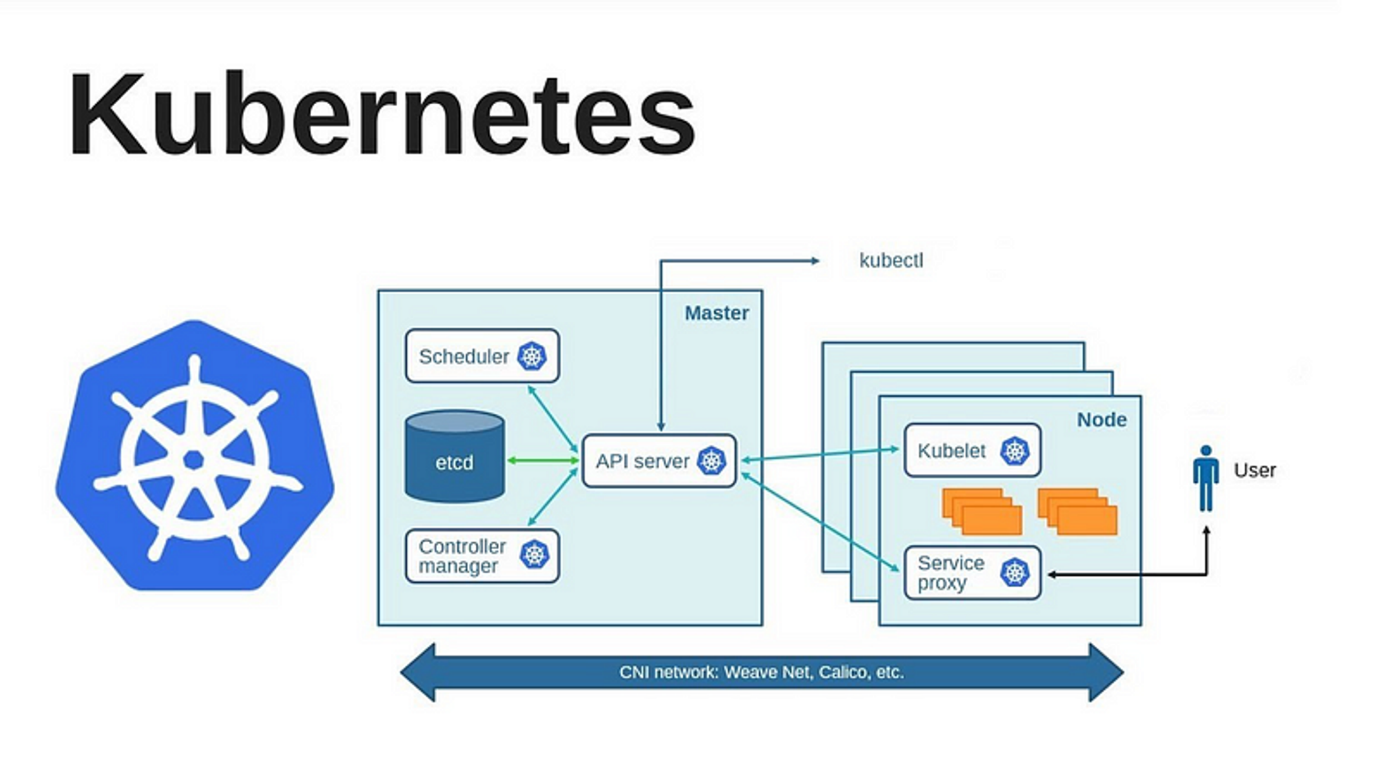
**⬤ Cloud-based K8s services**

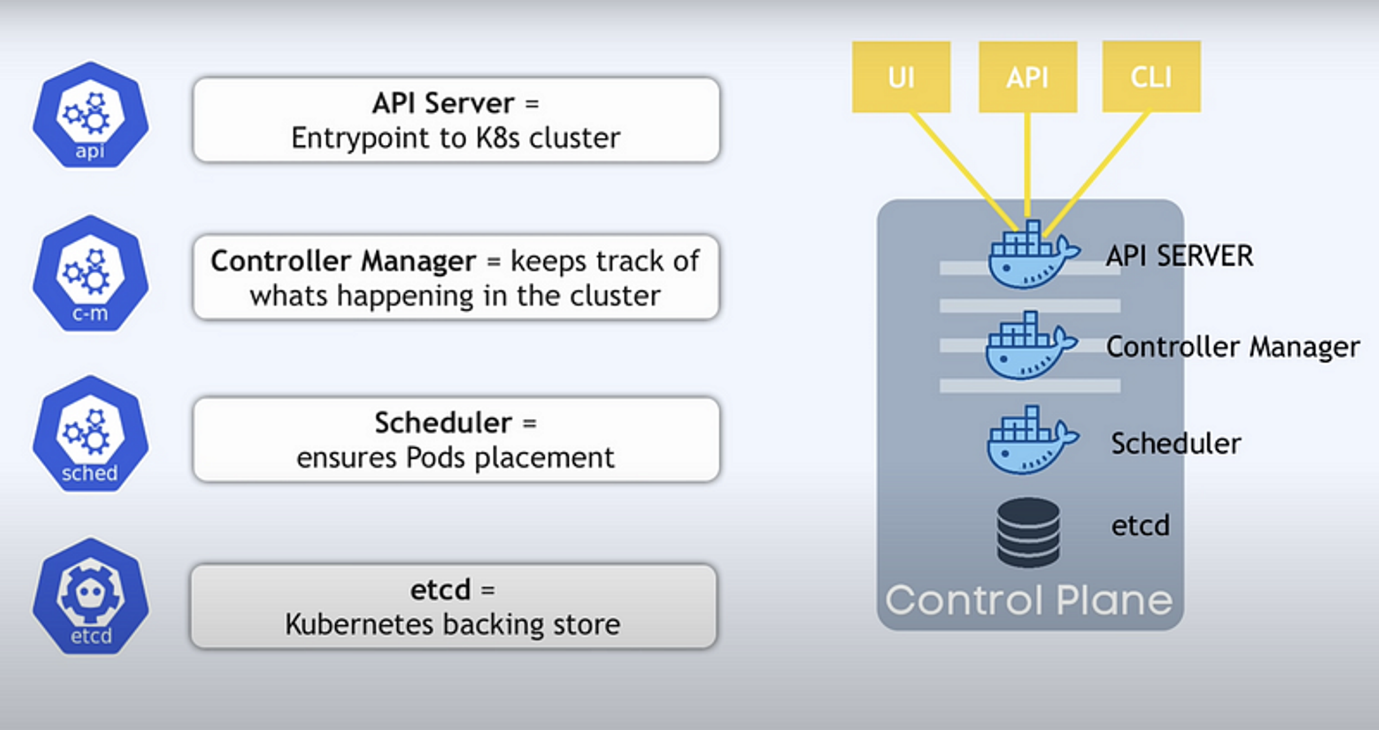
* GKE [Google Kubernetes Engine]
* AKS [Azure Kubernetes Service]
* Amazon EKS [Amazon Elastic Kubernetes Service]

**⬤ Features of Kubernetes**

* Orchestration (clustering of any number of containers running on a different network).
* Autoscaling (Vertical & Horizontal)
* Auto Healing
* Load Balancing
* Platform Independent (Cloud/Virtual/Physical)
* Fault Tolerance (Node/POD/Failure)
* Rollback (Going back to the previous version)
* Health monitoring of containers
* Batch Execution (One time, Sequential, Parallel)

**⬤ Architecture of Kubernetes**





**⬤ Working with Kubernetes**

* We create a Manifest (.yml) file
* Apply those to cluster (to master) to bring it into the desired state.
* POD runs on a node, which is controlled by the master.

**⬤ Role of Master Node**

* Kubernetes cluster contains containers running or Bare Metal / VM instances/cloud instances/ all mix.
* Kubernetes designates one or more of these as masters and all others as workers.
* The master is now going to run a set of K8s processes. These processes will ensure the smooth functioning of the cluster. These processes are called the ‘Control Plane.
* Can be Multi-Master for high availability.
* Master runs control plane to run cluster smoothly.

**⬤ Components of Control Plane**

**◼ Kube-api-server → (For all communications)**

* This api-server interacts directly with the user (i.e we apply .yml or .json manifest to kube-api-server)
* This kube-api-server is meant to scale automatically as per load.
* Kube-api-server is the front end of the control plane.

**◼ etcd**

* Stores metadata and status of the cluster.
* etcd is a consistent and high-available store (key-value-store)
* Source of touch for cluster state (info about the state of the cluster)

***→ etcd has the following features***

1. Fully Replicated → The entire state is available on every node in the cluster.
2. Secure → Implements automatic TLS with optional client-certificate authentication.
3. Fast → Benchmarked at 10,000 writes per second.

**◼ Kube-schedular (action)**

* When users request the creation & management of Pods, Kube-scheduler is going to take action on these requests.
* Handles POD creation and Management.
* Kube-scheduler match/assign any node to create and run pods.
* A scheduler watches for newly created pods that have no node assigned. For every pod that the scheduler discovers, the scheduler becomes responsible for finding the best node for that pod to run.
* The scheduler gets the information for hardware configuration from configuration files and schedules the Pods on nodes accordingly.

**◼ Controller-Manager**

* Make sure the actual state of the cluster matches the desired state.

→ Two possible choices for controller manager —

1. If K8s is on the cloud, then it will be a cloud controller manager.
2. If K8s is on non-cloud, then it will be kube-controller-manager.

**◼ Components on the master that runs the controller**

**Node Controller →** For checking the cloud provider to determine if a node has been detected in the cloud after it stops responding.

**Route-Controller →** Responsible for setting up a network, and routes on your cloud.

**Service-Controller →** Responsible for load Balancers on your cloud against services of type Load Balancer.

**Volume-Controller →** For creating, attaching, and mounting volumes and interacting with the cloud provider to orchestrate volume.

**◼ Nodes (Kubelet and Container Engine)**

* Node is going to run 3 important pieces of software/process.

**Kubelet**

* The agent running on the node.
* Listens to Kubernetes master (eg- Pod creation request).
* Use port 10255.
* Send success/Fail reports to master.

**Container Engine**

* Works with kubelet
* Pulling images
* Start/Stop Containers
* Exposing containers on ports specified in the manifest.

**Kube-Proxy**

* Assign IP to each pod.
* It is required to assign IP addresses to Pods (dynamic)
* Kube-proxy runs on each node & this makes sure that each pod will get its unique IP Address.
* These 3 components collectively consist of ‘node’.

**⬤ POD**

* The smallest unit in Kubernetes.
* POD is a group of one or more containers that are deployed together on the same host.
* A cluster is a group of nodes.
* A cluster has at least one worker node and a master node.
* In Kubernetes, the control unit is the POD, not the containers.
* Consist of one or more tightly coupled containers.
* POD runs on a node, which is controlled by the master.
* Kubernetes only knows about PODs (Does not know about individual containers).
* Cannot start containers without a POD.
* One POD usually contains One Container.

◼ Multi Container PODs →

* Share access to memory space.
* Connect using Localhost <Container-Port>
* Share access to the Same Volume.
* Containers within POD are deployed in an all-or-nothing manner.
* The entire POD is hosted on the same node (Scheduler will decide which node).
* There is no auto-healing or scaling by default.

**⬤ Higher-level Kubernetes Objects**

* **Replication Set →** Auto scaling and auto-healing.
* **Deployment →** Versioning and Rollback.
* **Service →** Static (Non-ephemeral) IP and Networking.
* **Volume →** Non-ephemeral storage [Ephemeral → Storage outside the node]