**KUBERNETES**

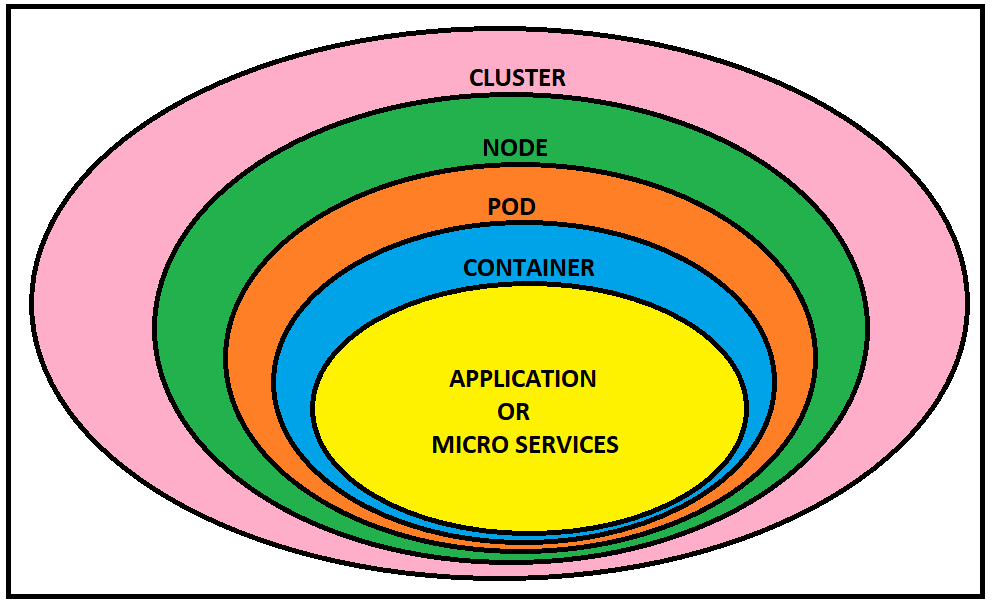
* Kubernetes is an Open-source container management tool which Automates Container Deployment, Container Scaling and Load Balancing.
* Kubernetes — also known as “k8s” or “kube”
* The project was created by Joe Beda, Brendan Burns, and Craig McLuckie
* The design and development of Kubernetes was influenced by Google's [Borg](https://en.wikipedia.org/wiki/Borg_(cluster_manager)) cluster manager. Many of its top contributors had previously worked on Borg;
* It schedules, runs and manages isolated containers which are running on Virtual/Physical/cloud machines.
* It Is supported by all the Cloud providers.

**HISTORY**

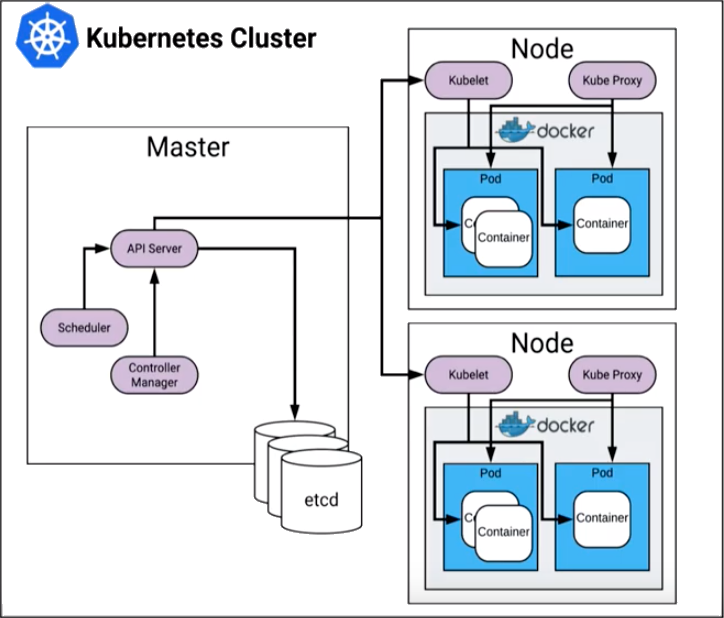
* Google developed an internal system called “BROG” (later names as OMEGA) to deploy and manages Thousands of Google Application and services and their cluster.
* In 2014, google introduced Kubernetes as an Open-source platform.
* Kubernetes written in Go-lang.
* Kubernetes was later donated to Cloud Native Computing Foundation (CNCF).
* Cloud native means it will develop existing features in cloud for better purpose usage.

**FEATURES**

* Orchestration (Clustering of any number of containers running on different networks).
* Auto-scaling (Vertical [Existing] - >most Preferable and Horizontal [New]) and Auto-healing.
* Load balancing.
* Platform Independent (Cloud/Virtual/Physical).
* Fault tolerance (Node/Pod failure).
* Roll back (Going back to previous version).
* Health Monitoring of Containers. If one Container fails it will create another container.
* Batch Execution (One time, Sequential, Parallel).
* Scripts in K8s is called Manifest, which is in form of Json or YAML.



**ARCHITECTURE**



**Master components**

* **API Server:** Exposes the API.
* **ETCD Cluster:** Key value stores all cluster data. (Can be run on the same server as a master node or on a dedicated cluster.)
* **Kube-scheduler:** Schedules new pods on worker nodes.
* **Kube-controller-manager:** Runs the controllers.
* **Cloud-controller-manager:** Talks to cloud providers.

**Node components**

* **Kube-proxy:** Keeps network rules like Addressing IP to Pods.
* **Kubelet:** Agent that ensures containers in a pod are running.
* **Container engine:** Maintains the containers like Docker, Rocket etc.

**CONTROLPANEL COMPONENTS:**

**KUBE-API SERVER** (For all Communications)

* It is front end of the Control Plane.
* The main implementation of a Kubernetes API server is Kube-Apiserver.
* kube-apiserver is designed to scale horizontally—that is, it scales by deploying more instances.
* It will directly interact with the user (I.e we apply .yml or json manifest to kube-Apiserver).
* This kube-Apiserver is directly meant to scale Automatically as per load.

**ETDC**

* It is nothing but the Data base which Stores Meta data and status of cluster.
* It is consistent and high available store (Key - Value store).
* Source of touch for Cluster state (information about the state of Cluster).

**KUBE-SCHEDULER**

* When user makes a request for creation and management of pods, Kube-scheduler is going to take the action on those requests, Handles pod creation and Management.
* Kube-scheduler match/assign any node to create and run pod.
* A scheduler watches for newly created pods that have no node assigned for every pod that the scheduler discovers, it becomes responsible for finding best node for that pod to run on.
* It gets the information for hardware configuration from configuration files and schedules the pods on the nodes Accordingly.

**CONTROL MANAGER**

* Makes sure actual state of cluster matches to the desired state.

Two possible choices for Control manager

1. If K8s on cloud, then it will be Cloud-Controller-Manager.
2. If K8s on non-cloud, then it will be Kube-Controller-Manager.

Components on master that runs 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**: For setting up network, routes on your cloud.
* **Service-controller**: For load balancers on your cloud against services of type Load Balancers.
* **Volume-controller**: For Creating, attaching and mounting volumes and interacting with the cloud provider to Orchestrate Volume.

**NODE COMPONENTS**

**KUBELET**

* It Is nothing but Agent running on the Node.
* Listens to the Kubernetes Master (Ex: Pod creation request).
* Uses the port :10255 (can be changeable).
* Sends Successful/Fail reports to the Master.

**CONTAINER ENGINE**

* It can be Docker, Rocket or any other.
* Works with Kubelet.
* Pulling images.
* Start/Stop the Containers.
* Exposing Containers on ports specified in the Manifest.

**KUBE-PROXY**

* Assigns IP to each pod.
* It is required to assign IP addresses to Pods (Dynamic IP).
* It runs on each node & this will make sure that each pod will gets its unique IP address.

All the three components Kubelet, Kube-Proxy and Container engine Is collectively called as Node.

**POD**

* Smallest unit of the Kubernetes.
* It 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 atleast one master node and one worker node.
* In Kubernetes, control unit is the pod, not containers.
* Consistent of one or more Tightly coupled containers.
* Pod runs on node, which is controlled by master.
* Kubernetes only knows about Pods (it does not know about individual container).
* Cannot start container without a Pod.
* One Pod usually contain only one container.

**MULTI CONTAINER PODS**

* Share access to memory space.
* Connect to each other using localhost. <container port>
* Share access to the same Volume.
* Containers with in the pods are deployed in an all-or-nothing manner.
* Entire pod is hosted on the same node (scheduler will decide about which node).

**LIMITATIONS**

* By default, no Auto-scaling or Auto-scaling, we need to do it Manually.
* POD Crashes, Solution for that is Higher level K8s Objects but, we need to add these.

**Higher level K8s Objects**

* **Replication-set**: Scaling and Healing.
* **Deployment**: Versioning and Roll back.
* **Service**: Static (Non-Ephemeral) IP and Networking.
* **Volume**: Non-Ephemeral Storage.

**IMPORTANT NOTATIONS**

* Kubectl : Single Cloud.
* Kubeadm : On-Premises.
* Kubefed : Federated (Hybrid cloud).

**Pod:** A group of one or more containers.

**Service:** An abstraction that defines a logical set of pods as well as the policy for accessing them.

**Volume:** An abstraction that lets us persist data. (This is necessary because containers are ephemeral—meaning data is deleted when the container is deleted.)

**Namespace:** A segment of the cluster dedicated to a certain purpose, for example a certain project or team of developers.

**Replica-Set (RS):** Ensures that desired amount of pod is what’s running.

**Deployment:** Offers declarative updates for pods an RS.

**Stateful-Set:** A workload API object that manages stateful applications, such as databases.

**Daemon-Set:** Ensures that all or some worker nodes run a copy of a pod. This is useful for daemon applications like fluentd.

**Job:** Creates one or more pods, runs a certain task(s) to completion, then deletes the pod(s).

**MINI KUBE:**

**Minikube**creates a single node cluster inside a VM or Cloud Instance. It is good for beginners to learn Kubernetes since you don’t have to create a master and worker node to create a cluster and we can practice basic Kubernetes functions and can also install the Kubernetes dashboard on it.

**PRE-REQUISTES**

* Minimum 2 CPU’s or more
* Minimum 2GB of free memory
* Minimum 20GB of free disk space
* Internet connection
* Container or virtual machine manager, such as: Docker, Hyperkit, Hyper-V, KVM, Parallels, Podman, VirtualBox, or VMware Fusion/Workstation

**MINI KUBE INSTALLATION:**

* apt update -y
* curl -LO https://storage.googleapis.com/kubernetes-release/release/`curl -s [https://storage.googleapis.com/kubernetes-release/release/stable.txt`/bin/linux/amd64/kubectl](https://storage.googleapis.com/kubernetes-release/release/stable.txt%60/bin/linux/amd64/kubectl)
* chmod +x ./kubectl
* sudo mv ./kubectl /usr/local/bin/kubectl
* kubectl version

**DOCKER INSTALLATION:**

* sudo apt-get install docker.io -y
* sudo systemctl status docker
* sudo usermod -aG docker $USER && newgrp docker

**MINIKUBE INSTALLATION:**

* curl -Lo minikube <https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64>
* chmod +x minikube
* sudo mv minikube /usr/local/bin/
* minikube version
* sudo minikube start --vm-driver=none
* X Exiting due to GUEST\_MISSING\_CONNTRACK: Sorry, Kubernetes 1.22.3 requires conntrack to be installed in root's path
* sudo apt-get install -y conntrack
* minikube start --vm-driver=none
* minikube status
* kubectl cluster-info
* kubectl get events
* kubectl config view
* kubectl run hello-minikube --image=gcr.io/google\_containers/echoserver:1.4 --port=8080
* kubectl get pods

**DEPLOYING AN APP**

* kubectl create deployment hello-node --image=k8s.gcr.io/echoserver:1.4
* kubectl get deployment
* kubectl expose deployment hello-node --type=NodePort --port=8080Copied!
* kubectl get svc
* curl -v public-ip:32548
* minikube ip
* curl -v private-ip:32548
* <http://private-ip:32548/>
* kubectl delete service hello-node
* kubectl delete deployment hello-node
* minikube stop
* minikube delete

