KUBERNETES (K8s)

- Kubernetes is an open-source container management tool which automates container deployment, container scaling and load balancing.
- It schedules, runs and manages isolated containers which are running on virtual/ physical/ cloud machines.
- All top cloud providers support Kubernetes.

History:

- Google developed an internal system called 'borg' (later named as omega) to deploy and manage thousands google application and services on their cluster.
- In 2014, google introduced Kubernetes as an open-source platform written in Golang, and later donated to CNCF (cloud Native Computing Foundation).

Online platform for K8s:

i. Kubernetes playground
ii. Play with K8s
iii. Play with Kubernetes classroom

Cloud based K8s services:

i. GKE: Google Kubernetes Services
ii. AKS: Azure Kubernetes services
iii. Amazon EKS: Amazon Elastic Kubernetes Services

Kubernetes installation tool:

i. Minicube
ii. Kubeadm

Problems with scaling up the containers:

- Containers cannot communicate with each other.
- Autoscaling and load balancing was not possible.
- Containers had to be managed carefully.

Features of Kubernetes:

- Orchestration (clustering of any number of containers running on different networks)
- Autoscaling (supports both horizontal and vertical scaling)
- Auto-healing
- Load balancing
- Platform independent (cloud/ virtual/ physical)
- Fault tolerance (node/ POD failure)

- Rollback (going back to previous version)
- Health monitoring of containers
- Batch execution (one time, sequential, parellel)

FEATURES	KUBERNETES	DOCKER SWARM
Installation and cluster configuration	Complicated and time consuming	Fast and easy
Supports	K8s can work with almost all container types like rocket, docker, containerD	Work with docker only
GUI	GUI available	GUI not available
Data volumes	Only shared with containers in same POD	Can be shared with any other containers
Update and rollback	Process scheduling to maintain services while updating	Progressive updates and service health monitoring throughout the update
Autoscaling	Support vertical and horizontal autoscaling	Not support autoscaling
Logging and monitoring	Inbuilt tool present for monitoring	Used 3rd party tools like splunk

Working with Kubernetes:

- We create manifest (.yml/json).
- Apply this to cluster (to master) to bring desired state.
- POD runs on node which is controlled by 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 master and all others as workers.
- The master is now going to run set of k8s processes. These processes will resume smooth functioning of cluster. These processes are called “control plane”.
- Can be multi-master for high availability.
- Master runs control plane to run cluster smoothly.

Components of Control Plane (master):

1. Kube-API server
2. Etcd
3. Kube-scheduler
4. Controller manager

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

2. **Etcd:**
   - It stores metadata and status of cluster.
   - Etcd is consistent and high available store (key-value store).
   - Source of touch for cluster state (info about state of cluster).

   Etcd has following features: -
   a. Fully replicated: the entire state is available on every node in the cluster.
   b. Secure: implements automatic TLS with optional client-certificate authentication.
   c. Fast: benchmarked at 10,000 writes per second.

3. **Kube scheduler:**
   - When users make request for the creation and 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 best node for that POD to run on.
   - Scheduler gets the information for hardware configuration from configuration file and schedules the PODs on nodes accordingly.

4. **Controller Manager:**
   - Make sure that actual state of cluster matches the desired state.
   - Two possible choices for controller manager:
     a. If k8s on cloud, then it will be cloud-controller-manager.
     b. If k8s on non-cloud then it will be kube-controller-manager

   Components on master that runs controller:

   a. **Node controller:** for checking the cloud providers to determine of a node has been detected in the cloud after id stops responding.
   b. **Route controller:** responsible for setting up network, routes on your cloud.
   c. **Service controller:** responsible for load balancers on your cloud against services of type load balancers.
   d. **Volume controller**: for creating, attaching and mounting volumes and interacting with the cloud provider to orchestrate volume.

Nodes (kublet and container engine.)

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

1. Kublet
2. Container engine
3. Kubeproxy

1. **Kublet:**
   - Agent running on the node.
   - Listens to Kubernetes master. (e.g-POD creation request)
   - Use port 10255.
   - Send success/fail reports to master.

2. **Container Engine:**
   - Works with kublet.
   - Pilling images.
   - Start/ stop containers.
   - Exposing containers on ports specified in manifest.

3. **Kubeproxy:**
   - Assign IP to each POD.
   - It is required to assign IP address to PODs (dymanic).
   - Kube-proxy runs on each node and this make sure that each POD will get its own unique IP address. These 3 components collectively called NODE.

POD:

- 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 master node.
- In Kubernetes the control unit is the POD, not containers.
- It consists of one or more tightly coupled containers.
- POD runs on node which is control by master.
- Kubernetes only knows about PODs (does not know about individual container).
- Cannot start containers without a POD.
- One POD usually contains one container.

Multi container PODs:

- Share access to memory space.
- Connect to each other using local host <container port>.
- Share access to the same volume.
- Containers within POD are deployed in an all-or-nothing manner.
- Entire POD is hosted on the same node (scheduler will decide about which node).

POD limitations:

- No auto healing or auto scaling.
- POD crashes.

Higher level Kubernetes objects:

a. Replication set: auto scaling and auto healing
b. Deployment: versioning and rollback
c. Service:
d. Static (non-ephemeral) IP and networking.
e. Volume: non-ephemeral storage.

Important:

Kubectl- single cloud

Kubeadm- on premise

Kubefed- federated

DEMO: Setup Kubernetes master and node on AWS:

Login into AWS account – launch 3 instances – ubuntu (t2.medium)

Master must have 2vcpu and 4gb RAM

Now using puttygen, create private key

Access all the 3 instances (1 master, 2 node) using putty.

Commands common for master and node:

sudo su

apt-get update

Now install https package

apt-get install apt-transport-https

This https is needed for intra cluster communication (particularly from control plane to individual pods)

Now install docker on all 3 instances

sudo apt install docker.io -y

To check whatever docker is installed or not

docker --version

systemctl start docker

systemctl enable docker

Setup open GPG key. This is required for intra cluster communication. It will be added to source key on this node i.e when k8s sends signed information to our host, it is going to accept those information because this open GPG key is present in the source key.

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | # sudo apt-key add

Edit source list file (apt-get install nano)

nano /etc/apt/source.list.d/Kubernetes.list

deb http://apt.kubernetes.io/kubernetes-xenial.main

Exit from nano: ctrl+x, caps+y – enter

apt-get update – install all packages

apt-get install -y kublet kubeadm kubectl Kubernetes-cni

Bootstrapping the master node (in master)

To initialize k8s cluster

kubeadm init

You will get one long command started from "kubeadm join 172.31.6.165:6443"

Copy this command and save on notepad

Create both .kube and its parent directories (-p)

mkdir -p $HOME/.kube

Copy configuration to kube directory (in configuration file)

sudo cp -i /etc/Kubernetes/admin.config $HOME /.kube/config

Provide user permissions to config file

chown $(id-u): $(id-g) $ HOME/.kube/config

Deploy flannel node network for its repository path. Flannel is going to place a binary in each node.

sudo kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/documentation/kube-flannel.yml

sudo kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/k8s-manifest/kube-flannel-rbac.yml

Configure worker node: paste the long commands above on both the nodes

Go to master: # kubectl get nodes