##
UBUNTU KUBERNETES INSTALLTION
##

########################===PRE-REQUISITES===#################################

#Ref https://kubernetes.io/docs/setup/independent/install-kubeadm

KMASTER
*Minimum 2 core CPU with 2 GB RAM
*Docker
*ssh
*kubeadm kubectl kubelet
----> Required ports

Master node(s)

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	6443*	Kubernetes API server	All
TCP	Inbound	2379-2380	etcd server client API	kube-apiserver, etcd
TCP	Inbound	10250	Kubelet API	Self, Control plane
TCP	Inbound	10251	kube-scheduler	Self
TCP	Inbound	10252	kube-controller-manager	Self

KNODE
*Minimum 1 core CPU with 1 GB RAM
*Docker
*ssh
*kubeadm kubectl kubelet
-----> Required ports

Worker node(s)

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	10250	Kubelet API	Self, Control plane
TCP	Inbound	30000-32767	NodePort Services**	All

```
##############################################################################
#                          Host Setup for kubernetes cluster                 #
##############################################################################

#updating packages

apt-get update

#changing the hostname of master

hostnamectl set-hostname kmaster

#to make hostname permenant
exec bash

#if swap memory is there, we need to turn it off , first we need to check the swap memory and
turn it off

df -h

#reason behind turn off swap memory :-

The idea of kubernetes is to tightly pack instances to as close to 100% utilized as possible.
All deployments should be pinned with CPU/memory limits. So if the scheduler sends a pod to a
machine it should never use swap at all.
You don't want to swap since it'll slow things down.Its mainly for performance.

#trun off the swap memeory

swapoff-a

#we need cluster hosts in etc/hosts

vi /etc/hosts

#Ex : 192.168.10.80 kmaster
        192.168.10.73 knode1
        192.168.10.32 knode2
```

```
##############################################################################
#                                DOCKER INSTALLTION                          #
##############################################################################

#Update the apt package index:

sudo apt-get update

#Install packages to allow apt to use a repository over HTTPS:

sudo apt-get install \
apt-transport-https \
ca-certificates \
curl \
gnupg-agent \
software-properties-common

#Add Docker’s official GPG key:

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

#adding figerprint key for repository

sudo apt-key fingerprint 0EBFCD88

#command to set up the stable repository

sudo add-apt-repository \
"deb [arch=amd64] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) \
stable"

#Update the apt package index.

sudo apt-get update

#Install the latest version of Docker CE and containerd, or go to the next step to install a specific version:

sudo apt-get install docker-ce docker-ce-cli containerd.io

#version checking
```

```
docker -v

# check the docker status

systemctl status docker

#Ref

#   https://docs.docker.com/install/linux/docker-ce/ubuntu/


################################################################################
#                                    kubernetes installation                                    #
################################################################################

apt-get update && apt-get install -y apt-transport-https curl

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

cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF

apt-get update

apt-get install -y kubelet kubeadm kubectl

apt-mark hold kubelet kubeadm kubectl

#Checking the versions
kubectl version
kubeadm version
kubelet --version


#Ref
# https://kubernetes.io/docs/setup/independent/install-kubeadm/

################################################################################
#          NOTE : above commands same for Kmaster and Knodes                                  #
################################################################################

################################################################################
```

Configuration setup in Kmaster server
##

Initilization of kmaster

kubeadm init --apiserver-advertise-address= <internal ip address of master VM>
--pod-network-cidr=192.168.0.0/16

#Ex :

kubeadm init --apiserver-advertise-address=10.1.3.71 --pod-network-cidr=192.168.0.0/16

#Execute the steps as normal user

mkdir -p $HOME/.kube

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

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

#To verify, if kubectl is working or not, run the following command

kubectl get pods -o wide --all-namespaces

#installation of pod newtork

kubectl apply -f
https://docs.projectcalico.org/v3.0/getting-started/kubernetes/installation/hosted/kubeadm/1.7/calico.yaml

kubectl get pods -o wide --all-namespaces

kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"

make sure all services including coredns and calico network services are running

kubectl get pods -o wide --all-namespaces

#Master setup has been done successfully

##
Nodes adding to Kmaster server
##

#For generating the join token for node

kubeadm token create --print-join-command

#EX :
kubeadm join 10.1.3.9:6443 --token w4vwwx.103s492wtfjfjed2
--discovery-token-ca-cert-hash
sha256:42b3d314a1d0d7ffbcbb7b6895f1c477033a627d2ec28bf41e5cf047925b66d9

#Check the host list in kmaster and knode

cat /etc/hosts

#NOTE :

#Above has to be execute in the kmaster server and then we will get join token

#Copy the join token and connect to knodes and execute the token

#Now check the all node status in kmaster server

kubectl get nodes

#After some time node status will show READY

##
CREATING THE DASHBORAD SERVICE
##

#Ref https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/

kubectl apply -f
https://raw.githubusercontent.com/kubernetes/dashboard/master/aio/deploy/recommended/kubernetes-dashboard.yaml

#Creating service account steps

```
kubectl create serviceaccount dashboard -n default

kubectl create clusterrolebinding dashboard-admin -n default \
--clusterrole=cluster-admin \
--serviceaccount=default:dashboard

kubectl get pods -o wide --all-namespaces

kubectl get svc --all-namespaces

#Curl <cluster_ip of dashboard> : <port>

#Edit the service file of kubernetes dashboard and change the type from ClusterIP to NodePort.

kubectl edit svc kubernetes-dashboard  --namespace=kube-system

kubectl get svc --all-namespaces

#Now open the firefox and type public IP with 443 and another port number

#Example :  https://12.111.33.111:3068

#For generating access token for Dashboard

kubectl get secret $(kubectl get serviceaccount dashboard -o jsonpath="{.secrets[0].name}") -o
jsonpath="{.data.token}" | base64 --decode

kubectl get secrets

kubectl describe secrets/dashboard-token-zgphz
```