**How to install kubernetes and run one pod using kubernetes such that the master node and worker node are on different virtual machines without using minikube?**

VM with IP Address 10.3.10.89 will have master node and VM with IP Address 10.3.10.90 will have worker node.

All the below commands can be implemented without using “sudo”, if in the beginning we use the command “sudo -s”.

**Edit On both VM :**

You need to enter you hostname and ip of both vm in /etc/hosts file as below

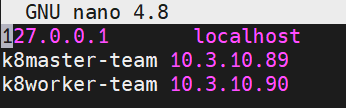
You can open this file by cmd-

nano /etc/hosts

If u want u can edit your hostname,it can be done by cmd-

sudo hostnamectl set-hostname <new-hostname>

When ever you want to change your hostname you need to change in /etc/hosts file



**On worker’s node:**

ssh -p 9094 guest@180.151.3.44

ssh jyoti@10.3.10.90

**Update the system:**

sudo apt update

sudo apt upgrade

**Install Docker:**

sudo apt install docker.io

**Start and enable the Docker service:**

sudo systemctl start docker

sudo systemctl enable docker

**Install Kubernetes components:**

sudo apt install kubeadm kubelet kubectl

**On master node’s VM :**

ssh -p 9094 guest@180.151.3.44

ssh jyoti@10.3.10.89

All the below commands can be implemented without using “sudo”, if in the beginning we use the command “sudo -s”.

**Update the system:**

sudo apt update

sudo apt upgrade

**Download the Google Cloud public signing key:**

curl -fsSL https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-archive-keyring.gpg

**Add the Kubernetes apt repository:**

echo "deb [signed-by=/etc/apt/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list

**Install Docker:**

sudo apt install docker.io

**Start and enable the Docker service:**

sudo systemctl start docker

sudo systemctl enable docker

**Install Kubernetes components:**

sudo apt install kubeadm kubelet kubectl

**Initialize the Kubernetes cluster:**

sudo kubeadm init --pod-network-cidr=192.168.0.0/16

This command will output a kubeadm join command, which you'll need later.

If any error comes during this command, check the 5th page **.**

**Set up the Kubernetes configuration for the current user:**

mkdir -p $HOME/.kube

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

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

**Install a network plugin (for networking between pods):**

kubectl apply -f <https://docs.projectcalico.org/v3.21/manifests/calico.yaml>

**Verify the status of the cluster:**

kubectl get nodes

You should see the master node in the "Ready" state.

**Join the worker node to the cluster (using the kubeadm join command from step 7):**

**(This step should be done on worker node’s VM)**

sudo kubeadm join <master-node-ip>:<master-node-port> --token <token> --discovery-token-ca-cert-hash <hash>

Replace <master-node-ip>, <master-node-port>, <token>, and <hash> with the values provided by the kubeadm init command on master node(step 7).you can also get this cmd by entering the cmd “sudo kubeadm token create --print-join-command" on master node.

**Verify that both the master and worker nodes are in the "Ready" state:**

kubectl get nodes

**Create a sample pod:**

Create a file named pod.yaml with the following contents:

apiVersion: v1

kind: Pod

metadata:

name: my-pod

spec:

containers:

- name: my-container

image: nginx:latest

resources:

limits:

cpu: "1"

memory: "512Mi"

requests:

cpu: "1"

memory: "256Mi"

**Deploy the pod:**

kubectl apply -f pod.yaml

**Verify the pod is running:**

kubectl get pods

**If this command shows pod with Status as ContainerCreating, then use command:**

kubectl get pods -o wide

This will show the pod with Status as Running.

Now, you can again check the Running status of nodes by using **kubectl get nodes** command.

**To enter inside the pod :**

kubectl exec -it my-pod -- /bin/bash

**REGARDING ERRORS AND PROCESS OF DELETING THE CLUSTER IF REQUIRED (MASTER AND WORKER NODE):**

**Errors:**

If you are facing error like connection refused when you reboot your system while trying to get nodes or pods run this cmd

export KUBECONFIG=/etc/kubernetes/admin.conf

You can keep this cmd in bash file so you no need to run this cmd every time you login to your vm

You can describe the pods with which you are facing

Kubectl describe pod <pod-name> -n <namespace> (for pod)

Kubectl describe node <node-name> (for node)

You can check the logs of a pod or node

Kubectl logs <pod-name> -n <namespace>

**PROCESS DELETING THE CLUSTER:**

[Restart the cluster(master node)]  
 kubectl delete pods --all  
 kubectl drain <node name>  
 kubectl delete node <node name>  
 iptables -F  
 kubeadm reset  
 sudo kubeadm init --pod-network-cidr=192.168.0.0/16 --apiserver-advertise-address=10.3.10.89 --v=5  
 sudo rm -rf $HOME/.kube  
 mkdir -p $HOME/.kube  
 sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  
 sudo chown $(id -u):$(id -g) $HOME/.kube/config

**To create a pod using helm-charts:**

**INSTALLING HELM CHARTS:**

1. **Install Helm:** The installation instructions according to your specific operating system can be found from the official Helm documentation <https://helm.sh/docs/helm/helm_install/>.
2. **Initialize Helm:** Once Helm is installed, initialize Helm on your cluster by running the following command:

helm init

1. **Create a Helm chart by using cmd :**

helm create my-chart

**To create a pod using helm charts:**

Run cmd : helm install <pod-name> .<directory in which charts is present>

Eg. helm install my-pod ./my-chart/

(if you are outside my-chart directory)

helm install my-pod .

(if you are inside my-pod directory)

Use cmd : helm list

to see the deployed pods.

Cmd : kubectl get pods

It will give the list of running pods.