VIT Vidyalankar Institute of Technology Accredited A+ by NAAC	*DEPARTMENT OF INFORMATION TECHNOLOGY*

Semester	T.E. Semester V – Information Technology
Subject	Advance DevOps Lab
Subject Professor In-charge	Prof. Indu Anoop
Laboratory	(Leave blank for now)

Student Name		
Roll Number		
Grade and Subject Teacher's Signature		

Experiment	2	
Problem Statement	To understand Kubernetes Cluster Architecture, install and Spin up a Kubernetes Cluster on Linux Machines/Cloud Platforms	
Resources / Apparatus Required	Hardware: Computer System (Internet Connectivity)	Software: Web Browser
Details	**Theory:** Kubernetes led by google is an open-source platform for managing container technologies such as Docker. Docker lets you create containers for a pre-configured image and application. *Kubernetes [Greek for "Pilot"] provides the next step, allowing you to balance loads between containers and run multiple containers across multiple systems.*	

custom tcp 6443 -->both on worker and node
ssh 22

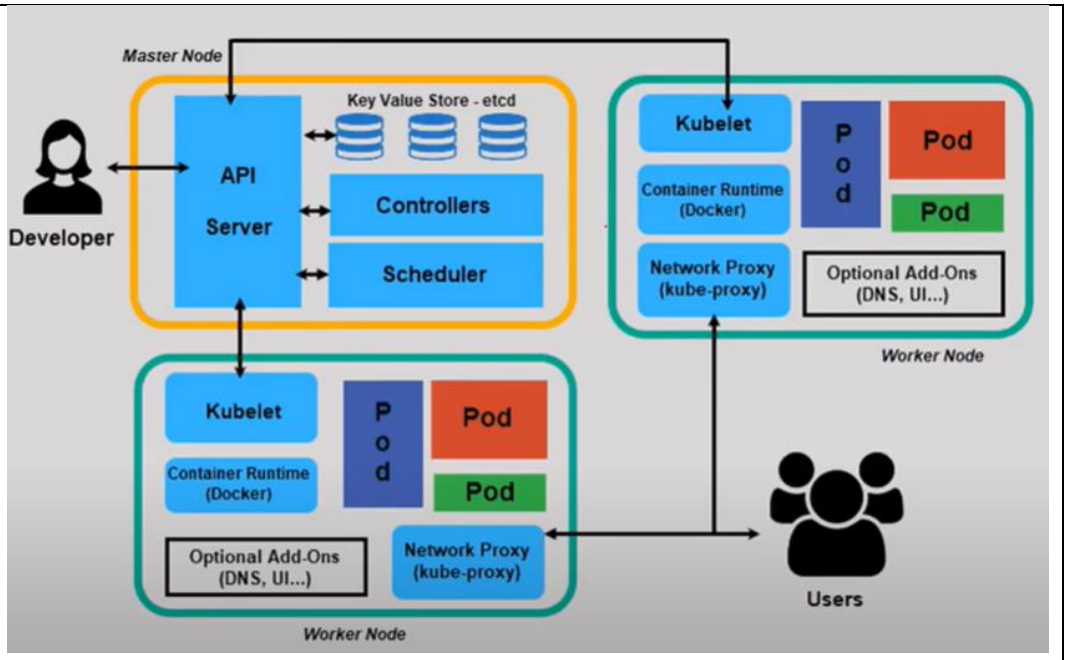
after step 8 copy the kubeadm msg

EC2 instance 2
settings default
security groups http 80
anywhere

step 4 kubernetes ke pehle space hai

no need of mobaxtrem we will do it on ec2 instance by clicking on connect

worker and master node

	 **Container:** Provides an isolated context in which an app together with it's environment (supporting structure eg: web server) can run. **Pods:** Represents a runnable unit usually consisting of a single container. [May contain more containers if containers are tightly coupled] Kubernetes connects the pod to the n/w and rest of the Kubernetes eco-system.
Code	Prerequisite: 2 AWS instance (virtual servers-ubuntu 20) one acting as Master Node and Other as Worker Node https://mobaxterm.mobatek.net/download.html **SECTION A: Docker Installation** **Step 1: Install Docker [on both master and worker]** Kubernetes requires an existing Docker installation. If you already have Docker installed, skip ahead to Step 2. If you do not have Docker , install it by following these steps on both master and worker nodes after having acquired root privilege using command [sudo su]: *apt-get update* *apt-get install docker.io -y* *docker --version*

Step 2: Start and Enable Docker [on both master and worker]

systemctl enable docker

systemctl status docker

systemctl start docker

SECTION B: Kubernetes Installation

Step 3: Add Kubernetes Signing Key [on both master and worker]

Since you are downloading Kubernetes from a non-standard repository, it is essential to ensure that the software is authentic. This is done by adding a signing key.

1. Enter the following to add a signing key:
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add

If you get an error that curl is not installed, install it with:
apt-get install curl

2. Then repeat the previous command to install the signing keys. Repeat for each server node.

Step 4: Add Software Repositories [on both master and worker]

Kubernetes is not included in the default repositories. To add them, enter the following:
apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"

Repeat on each server node.

Step 5: Kubernetes Installation Tools [on both master and worker]

Kubeadm (Kubernetes Admin) is a tool that helps initialize a cluster. It fast-tracks setup by using community-sourced best practices. Kubelet is the work package, which runs on every node and starts containers. The tool gives you command-line access to clusters.

1. Install Kubernetes tools with the command:
apt-get install kubeadm kubelet kubectl -y

apt-mark hold kubeadm kubelet kubectl
Allow the process to complete.

2. Verify the installation with:
kubeadm version

3. Repeat for each server node.

SECTION C: Kubernetes Deployment

Step 6: Begin Kubernetes Deployment

Start by disabling the swap memory on each server **[on both master and worker]** :
swapoff --a

Step 7: Assign Unique Hostname for Each Server Node [on master only]
Decide which server to set as the master node. Then enter the command:
hostnamectl set-hostname master-node

Next, set a worker node hostname by entering the following on the worker server **[on worker only]**:
hostnamectl set-hostname worker01

*If you have additional worker nodes, use this process to set a unique hostname on each.

Step 8: Initialize Kubernetes on Master Node [on master only]

Switch to the master server node, and enter the following:
kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=all

Once this command finishes, it will display a kubeadm join message at the end. Make a note of the whole entry. This will be used to join the worker nodes to the cluster.

Next, enter the following to create a directory for the cluster:
mkdir -p $HOME/.kube

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

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

Step 9: Deploy Pod Network to Cluster [on master only]

A Pod Network is a way to allow communication between different nodes in the cluster. We will use the flannel virtual network. Enter the following:
kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

Allow the process to complete.

Verify that everything is running and communicating:
kubectl get pods --all-namespaces

Step 10: Join Worker Node to Cluster [on worker only]

As indicated in Step 7, you can enter the kubeadm join command on each worker node to connect it to the cluster. Switch to the worker01 system and enter the command you noted from Step 7:

*kubeadm join 172.31.30.132:6443 --token td0tas.u34zdwnvwxgh0nke *
--discovery-token-ca-cert-hash sha256:47adb5a895cae9be531fa3219201bf2db921105dedd86f9a248f77c74edb3eac

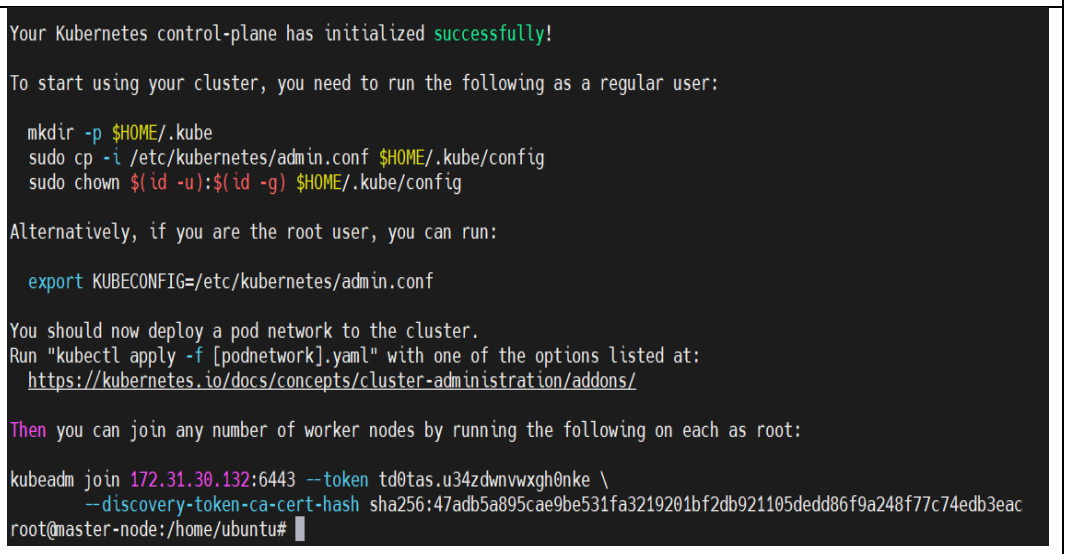
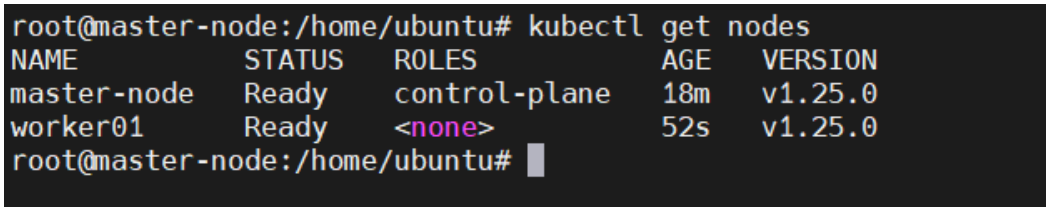
ON EC2 make sure you open the port in security group ADVERTISED HERE: Replace the alphanumeric codes with those from your master server. Repeat for each worker node on the cluster. Wait a few minutes; then you can check the status of the nodes.

Switch to the master server, and enter:
kubectl get nodes

The system should display the worker nodes that you joined to the cluster.

NAME	STATUS	ROLES	AGE	VERSION
master	Ready	master	1d	v1.14.0
worker1	Ready	<none>	1d	v1.14.0

If all of your nodes have the value Ready for STATUS, it means that they're part of the cluster and ready to run workloads. If, however, a few of the nodes have NotReady as the STATUS, it could mean that the worker nodes haven't finished their setup yet. Wait for around five to ten minutes before re-running kubectl get node and inspecting the new output. If a few nodes

	still have NotReady as the status, you might have to verify and re-run the commands in the previous steps. (*Kubernetes to be continued in Exp4: use of kubectl command and to manage cluster and deploy application) ***[Note: In case you want to stop and continue later, Remember to stop the running instances on AWS , also detach the volume to not incur costs just in case a free-tier resource is not chosen]***
Output	 
Conclusion	Kubernetes was installed and a Kubernetes Cluster was spun up on a ubuntu machine using AWS Cloud Platform.

do this in beginning

sudo su
hostnamectl set-hostname master-node
exit
sudo su
for master

sudo su
hostnamectl set-hostname worker
exit
sudo su
for worker

tput setaf 02 for green 01 for red