ADVANCE DEVOPS EXP 3

Name:Manav Punjabi
Class:D15A
Roll No:45

Aim:To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.

Step 1: Pre-requisites

1.1 Create 3 EC2 instances,one for the master node and two for the worker nodes.

1.2 Proceed with the following settings and create a new key pair as follows(use the same key pair for all the three nodes)

▼ **Instance type** Info | Get advice

Instance type

t2.medium
Family: t2   2 vCPU   4 GiB Memory   Current generation: true
On-Demand Linux base pricing: 0.0496 USD per Hour
On-Demand Windows base pricing: 0.0676 USD per Hour
On-Demand RHEL base pricing: 0.0784 USD per Hour
On-Demand SUSE base pricing: 0.1496 USD per Hour

⬤ All generations

Compare instance types

Additional costs apply for AMIs with pre-installed software

▼ **Key pair (login)** Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

two-tier-app-k8s

↻ Create new key pair

▼ **Network settings** Info   Edit

Network | Info

vpc-04007898e59a6979f

Subnet | Info

Create key pair

×

Key pair name

Key pairs allow you to connect to your instance securely.

```
two-tier-app-k8s
```

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

- ● **RSA**
  RSA encrypted private and public key pair

- ○ **ED25519**
  ED25519 encrypted private and public key pair

Private key file format

- ● **.pem**
  For use with OpenSSH

- ○ **.ppk**
  For use with PuTTY

⚠ When prompted, store the private key in a secure and accessible location on

Cancel   **Create key pair**

Instances (1/3) **Info**

Last updated less than a minute ago ⟳ | Connect | Instance state ▼ | Actions ▼ | **Launch instances** | ▼

Q *Find Instance by attribute or tag (case-sensitive)*   All states ▼

Instance state = running ✕   Clear filters   ‹ 1 › ⚙

	Name ✎ ▽	Instance ID	Instance state ▽	Instance type ▽	Status check	Alarm status	Availability Zone ▽	Public IPv4 DNS ▽	Public IP
☐	Worker-2	i-0e3930ceb2d892d01	⊘ Running ⊕ ⊖	t2.medium	⊘ 2/2 checks passec	View alarms ＋	ap-south-1a	ec2-13-234-226-219.ap…	13.234.22
☐	Worker-1	i-0d16e01d1824e0e3a	⊘ Running ⊕ ⊖	t2.medium	⊘ 2/2 checks passec	View alarms ＋	ap-south-1a	ec2-65-0-104-95.ap-so…	65.0.104.
☑	Master	i-01ae3d388db90ad73	⊘ Running ⊕ ⊖	t2.medium	⊘ 2/2 checks passec	View alarms ＋	ap-south-1a	ec2-13-232-36-34.ap-s…	13.232.36

1.3 After the instances have been created,copy the text given in the example part of each of the three instances into git bash.

EC2 Instance Connect | Session Manager | **SSH client** | EC2 serial console

Instance ID

🗗 i-0e3930ceb2d892d01 (Worker-2)

1. Open an SSH client.

2. Locate your private key file. The key used to launch this instance is two-tier-app-k8s.pem

3. Run this command, if necessary, to ensure your key is not publicly viewable.
   🗗 chmod 400 "two-tier-app-k8s.pem"

4. Connect to your instance using its Public DNS:
   🗗 ec2-13-234-226-219.ap-south-1.compute.amazonaws.com

Example:
🗗 ssh -i "two-tier-app-k8s.pem" ubuntu@ec2-13-234-226-219.ap-south-1.compute.amazonaws.com

```
acer@TMP214-53 MINGW64 ~/Downloads
$ ssh -i "two-tier-app-k8s.pem" ubuntu@ec2-13-232-36-34.ap-south-1.compute.amazo
naws.com
The authenticity of host 'ec2-13-232-36-34.ap-south-1.compute.amazonaws.com (13.
232.36.34)' can't be established.
ED25519 key fingerprint is SHA256:uVGEO+FWYefj60jOft7OSralv8NrzEi/IwxAtBY+EPE.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-13-232-36-34.ap-south-1.compute.amazonaws.com' (
ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-1022-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Wed Sep 11 14:07:10 UTC 2024

  System load:  0.0                Processes:              106
  Usage of /:   20.7% of 7.57GB    Users logged in:        0
  Memory usage: 5%                 IPv4 address for eth0: 172.31.45.227
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
```
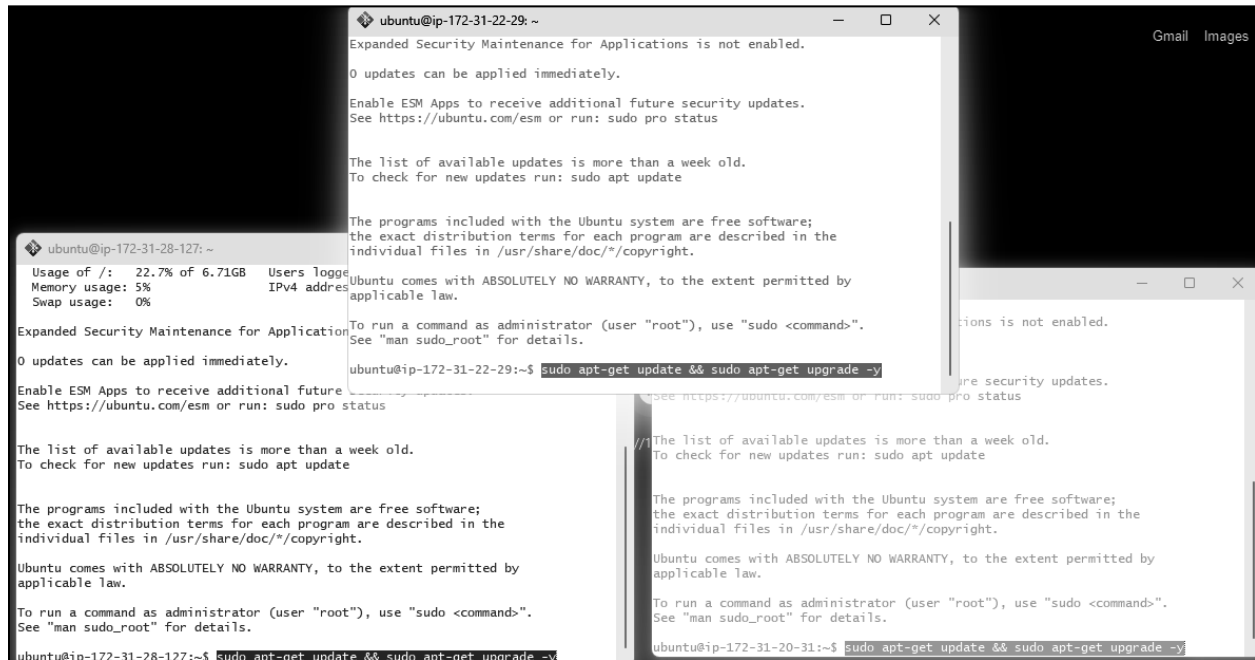
Step 2:Prepare Nodes

2.1. Update the package manager on all nodes:

sudo apt-get update && sudo apt-get upgrade -y

2.2. Disable Swap (Kubernetes requires swap to be off):

sudo swapoff -a

sudo sed -i '/ swap / s/^/#/' /etc/fstab

```
ubuntu@ip-172-31-22-29:~$ sudo swapoff -a
sudo sed -i '/ swap / s/^/#/' /etc/fstab
```

2.3. Load necessary kernel modules for networking and iptables:

cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf

overlay

br_netfilter

EOF

sudo modprobe overlay

sudo modprobe br_netfilter

```
ubuntu@ip-172-31-22-29:~$ cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF
sudo modprobe overlay
sudo modprobe br_netfilter
overlay
br_netfilter
```

2.4. Configure sysctl settings for Kubernetes networking:

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf

net.bridge.bridge-nf-call-ip6tables = 1

net.bridge.bridge-nf-call-iptables = 1

EOF

sudo sysctl --system

```
ubuntu@ip-172-31-22-29:~$ cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF
sudo modprobe overlay
sudo modprobe br_netfilter
overlay
br_netfilter
ubuntu@ip-172-31-22-29:~$ cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
sudo sysctl --system
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
* Applying /etc/sysctl.d/10-console-messages.conf ...
kernel.printk = 4 4 1 7
* Applying /etc/sysctl.d/10-ipv6-privacy.conf ...
net.ipv6.conf.all.use_tempaddr = 2
net.ipv6.conf.default.use_tempaddr = 2
* Applying /etc/sysctl.d/10-kernel-hardening.conf ...
kernel.kptr_restrict = 1
```

Step 3: Install Docker

Kubernetes uses container runtimes like Docker. Install Docker on all nodes.

sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable"
sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.io

```
ubuntu@ip-172-31-22-29:~$ sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl software-proper
ties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubun
tu $(lsb_release -cs) stable"
sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.io
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Fetched 129 kB in 1s (241 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20230311ubuntu0.22.04.1).
ca-certificates set to manually installed.
curl is already the newest version (7.81.0-1ubuntu1.17).
curl set to manually installed.
software-properties-common is already the newest version (0.99.22.9).
software-properties-common set to manually installed.
```

Configure Docker for Kubernetes:

```
cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF
sudo systemctl restart docker
```

```
ubuntu@ip-172-31-22-29:~$ cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF
sudo systemctl restart docker
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
```

Step 4: Install kubeadm, kubelet, kubectl

Install Kubernetes tools on all nodes.

4.1. Add Kubernetes APT repository:

```
sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg
https://packages.cloud.google.com/apt/doc/apt-key.gpg
echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg]
https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee
/etc/apt/sources.list.d/kubernetes.list
```

```
ubuntu@ip-172-31-22-29:~$ sudo curl -fsSLo /usr/share/keyrings/kubernetes-archiv
e-keyring.gpg https://packages.cloud.google.com/apt/doc/apt-key.gpg
echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https:/
/apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/k
ubernetes.list
deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.k
ubernetes.io/ kubernetes-xenial main
```

4.2. Install kubeadm, kubelet, and kubectl:

sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl

```
ubuntu@ip-172-31-22-29:~$ sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu jammy InRelease
```

Step 5: Initialize the Kubernetes Cluster on Master Node

On the master node:

sudo kubeadm init --pod-network-cidr=10.244.0.0/16

```
ubuntu@ip-172-31-22-29:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --v
=5
Found multiple CRI endpoints on the host. Please define which one do you wish to
 use by setting the 'criSocket' field in the kubeadm configuration file: unix://
/var/run/containerd/containerd.sock, unix:///var/run/crio/crio.sock
k8s.io/kubernetes/cmd/kubeadm/app/util/runtime.detectCRISocketImpl
        cmd/kubeadm/app/util/runtime/runtime.go:167
k8s.io/kubernetes/cmd/kubeadm/app/util/runtime.DetectCRISocket
        cmd/kubeadm/app/util/runtime/runtime.go:175
k8s.io/kubernetes/cmd/kubeadm/app/util/config.SetNodeRegistrationDynamicDefaults
        cmd/kubeadm/app/util/config/initconfiguration.go:118
k8s.io/kubernetes/cmd/kubeadm/app/util/config.SetInitDynamicDefaults
        cmd/kubeadm/app/util/config/initconfiguration.go:64
k8s.io/kubernetes/cmd/kubeadm/app/util/config.DefaultedInitConfiguration
        cmd/kubeadm/app/util/config/initconfiguration.go:248
k8s.io/kubernetes/cmd/kubeadm/app/util/config.LoadOrDefaultInitConfiguration
        cmd/kubeadm/app/util/config/initconfiguration.go:282
k8s.io/kubernetes/cmd/kubeadm/app/cmd.newInitData
        cmd/kubeadm/app/cmd/init.go:319
k8s.io/kubernetes/cmd/kubeadm/app/cmd.newCmdInit.func3
        cmd/kubeadm/app/cmd/init.go:170
k8s.io/kubernetes/cmd/kubeadm/app/cmd/phases/workflow.(*Runner).InitData
        cmd/kubeadm/app/cmd/phases/workflow/runner.go:183
k8s.io/kubernetes/cmd/kubeadm/app/cmd.newCmdInit.func1
```

5.1. Set up kubectl on the master node:

mkdir -p $HOME/.kube

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

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

```
ubuntu@ip-172-31-22-29:~$ sudo kubeadm config images pull

sudo kubeadm init

mkdir -p "$HOME"/.kube
sudo cp -i /etc/kubernetes/admin.conf "$HOME"/.kube/config
sudo chown "$(id -u)":"$(id -g)" "$HOME"/.kube/config


# Network Plugin = calico
kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.0/manifests/calico.yaml

kubeadm token create --print-join-command --v=5
Found multiple CRI endpoints on the host. Please define which one do you wish to use by setting the 'criSocket' field in the kubeadm configuration file: unix:///var/run/containerd/containerd
ock, unix:///var/run/crio/crio.sock
To see the stack trace of this error execute with --v=5 or higher
Found multiple CRI endpoints on the host. Please define which one do you wish to use by setting the 'criSocket' field in the kubeadm configuration file: unix:///var/run/containerd/containerd
ock, unix:///var/run/crio/crio.sock
```

Step 6: Install a Pod Network Add-on

To enable communication between pods, install a pod network plugin like Flannel or Calico.

Install Flannel:

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

```
ubuntu@ip-172-31-22-29:~$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml --validate=False

E0913 15:35:04.261458   19259 memcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/api?timeout=32s": dial tcp 127.0.0.1:808
E0913 15:35:04.261902   19259 memcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/api?timeout=32s": dial tcp 127.0.0.1:808
E0913 15:35:04.263424   19259 memcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/api?timeout=32s": dial tcp 127.0.0.1:808
E0913 15:35:04.263795   19259 memcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/api?timeout=32s": dial tcp 127.0.0.1:808
E0913 15:35:04.265840   19259 memcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/api?timeout=32s": dial tcp 127.0.0.1:808
E0913 15:35:04.266524   19259 memcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/api?timeout=32s": dial tcp 127.0.0.1:808
unable to recognize "https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml": Get "http://localhost:8080/api?timeout=32s": dia
```

Step 7: Join Worker Nodes to the Cluster

On the **worker nodes**, run the command provided by the master node during initialization . It looks something like this:

sudo kubeadm join <master-ip>:6443 --token <token> --discovery-token-ca-cert-hash sha256:<hash>

```
clusterrolebinding.rbac.authorization.k8s.io/calico-cni-plugin created
daemonset.apps/calico-node created
deployment.apps/calico-kube-controllers created
kubeadm join 172.31.62.216:6443 --token br7fe5.hq28adbmn1mu17ky --discovery-token-ca-cert-hash sha256:2bc469a8d14fbeb0f879328d2b416fad
32b29a8505d3f448b98703fff3b014d9
```

Step 8: Verify the Cluster

Once the worker node joins, check the status on the **master node**

```
ubuntu@ip-172-31-45-227:~$ kubectl get nodes
NAME               STATUS   ROLES           AGE     VERSION
ip-172-31-43-211   Ready    <none>          50s     v1.29.0
ip-172-31-45-13    Ready    <none>          34s     v1.29.0
ip-172-31-45-227   Ready    control-plane   5m17s   v1.29.0
ubuntu@ip-172-31-45-227:~$
```