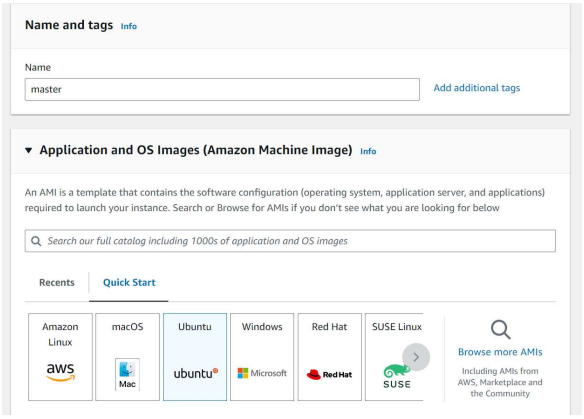
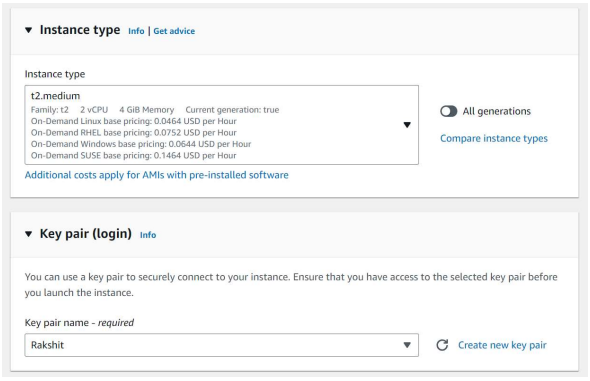
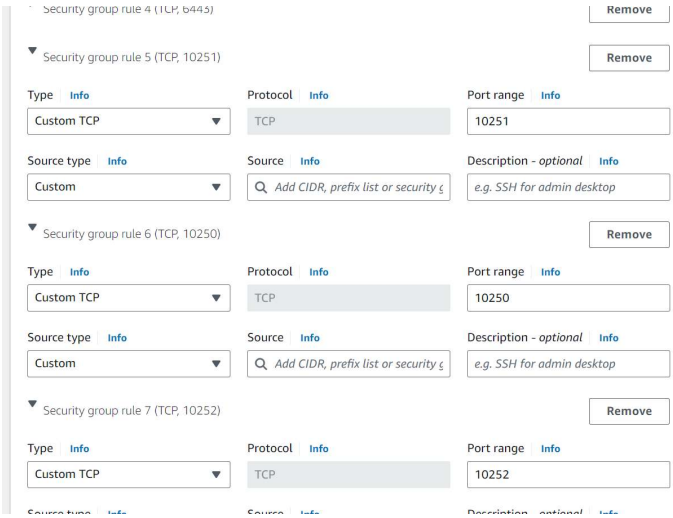
Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.

Step 1: Log in to your AWS Academy/personal account and launch 3 new Ec2 Instances. Select Ubuntu as AMI and t2.medium as Instance Type and create a key of type RSA with .pem extension and move the downloaded key to the new folder.We can use 3 Different keys or 1 common key also. Note: A minimum of 2 CPUs are required so Please select t2.medium and do not forget to stop the instance after the experiment because it is not available in the free tier.

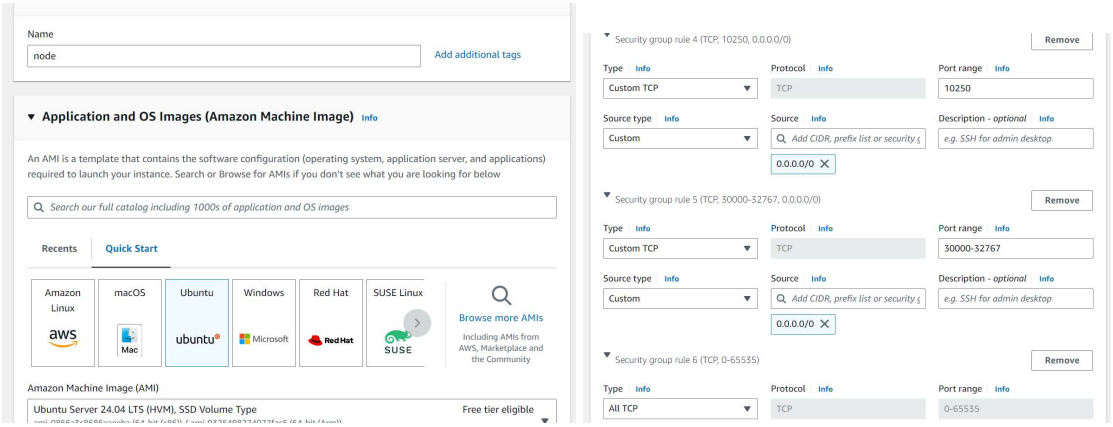
Master:



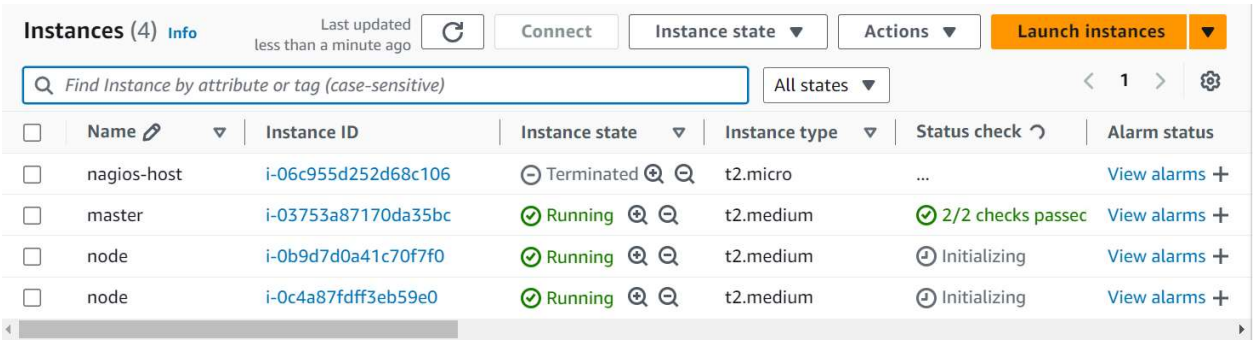


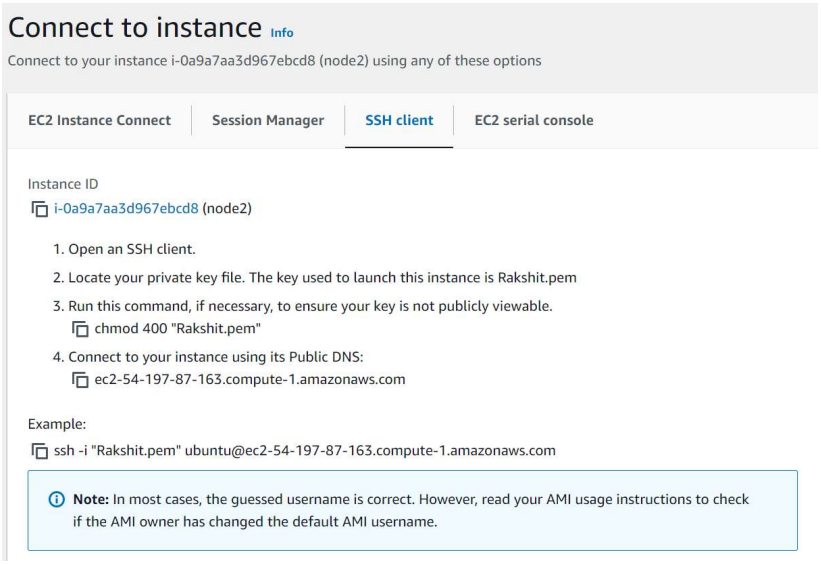


Worker:



Step 2: After creating the instances click on Connect & connect all 3 instances and navigate to SSH Client.





Step 3: Now open the folder in the terminal 3 times for Master, Node1& Node 2 where our .pem key is stored and paste the Example command (starting with ssh -i …..) in the terminal.(ssh -i "Master_Ec2_Key.pem" ubuntu@ec2-54-196-129-215.compute-1.amazonaws.com) Master:

```
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



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-81-239:~$
```

Step 4: Run on Master,Node 1,and Node 2 the below commands to install and setup Docker in Master, Node1, and Node2.

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add - curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

```
ubuntu@ip-172-31-81-239:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee
/etc/apt/trusted.gpg.d/docker.gpg > /dev/null
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable"
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBFit2ioBEADhWpZ8/wvZ6hUTiXOwQHXMAlaFHcPH9hAtr4F1y2+OYdbtMuth
lqqwp028AqyY+PRfVMtSYMbjuQuu5byyKR01BbqYhuS3jtqQmljZ/bJvXqnmiVXh
38UuLa+z077PxyxQhu5BbqntTPQMfiyqEiU+BKbq2WmANUKQf+1AmZY/IruOXbnq
L4C1+gJ8vfmXQt99npCaxEjaNRVYfOS8QcixNzHUYnb6emjlANyEVlZzeqo7XKl7
UrwV5inawTSzWNvtjEjj4nJL8NsLwscpLPQUhTQ+7BbQXAwAmeHCUTQIvvWXqw0N
cmhh4HgeQscQHYgOJjjDVfoY5MucvglbIgCqfzAHW9jxmRL4qbMZj+b1XoePEtht
ku4bIQN1X5P07fNWzlgaRL5Z4POXDDZTlIQ/El58j9kp4bnWRCJW0lya+f8ocodo
vZZ+Doi+fy4D5ZGrL4XEcIQP/Lv5uFyf+kQtl/94VFYVJOleAv8W92KdgDkhTcTD
G7c0tIkVEKNUq48b3aQ64NOZQW7fVjfoKwEZdOqPE72Pa45jrZzvUFxSpdiNk2tZ
XYukHjlxxEgBdC/J3cMMNRE1F4NCA3ApfV1Y7/hTeOnmDuDYwr9/obA8t016Yljj
q5rdkywPf4JF8mXUW5eCN1vAFHxeg9ZWemhBtQmGxXnw9M+z6hWwc6ahmwARAQAB
tCtFb2NrZXIgUmVsZWFzZSAoQ0UgZGViKSA8ZG9ja2VyQGRvY2tlci5jb20+iQI3
```

sudo apt-get update

sudo apt-get install -y docker-ce

```
ubuntu@ip-172-31-81-239:~$ sudo apt-get update
sudo apt-get install -y docker-ce
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt
/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  containerd.io docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0
  pigz slirp4netns
Suggested packages:
```

sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF

```
ubuntu@ip-172-31-81-239:~$ sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
```

sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker

```
ubuntu@ip-172-31-81-239:~$ sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
```

Step 5: Run the below command to install Kubernets.

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]

https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

```
ubuntu@ip-172-31-81-239:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o
/etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
gpg: missing argument for option "-o"
-bash: /etc/apt/keyrings/kubernetes-apt-keyring.gpg: No such file or directory
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
```

sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl

sudo systemctl enable --now kubelet
sudo apt-get install -y containerd

```
ubuntu@ip-172-31-81-239:~$ sudo apt-get install -y apt-transport-https ca-certificates curl
curl -fsSL https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list
sudo apt-get update
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203).
ca-certificates set to manually installed.
curl is already the newest version (8.5.0-2ubuntu10.4).
curl set to manually installed.
The following NEW packages will be installed:
  apt-transport-https
0 upgraded, 1 newly installed, 0 to remove and 6 not upgraded.
Need to get 3974 B of archives.
After this operation, 35.8 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 apt-transport-https all 2.7.14build2 [3974 B]
```

sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml

```
ubuntu@ip-172-31-81-239:~$ sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
temp = ""
version = 2

[cgroup]
  path = ""

[debug]
  address = ""
  format = ""
  gid = 0
  level = ""
```

sudo systemctl restart containerd
sudo systemctl enable containerd

sudo systemctl status containerd

```
ubuntu@ip-172-31-81-239:~$ sudo systemctl restart containerd
sudo systemctl enable containerd
sudo systemctl status containerd
● containerd.service - containerd container runtime
     Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: enabled)
     Active: active (running) since Thu 2024-10-03 12:03:56 UTC; 187ms ago
       Docs: https://containerd.io
   Main PID: 4388 (containerd)
      Tasks: 8
     Memory: 13.8M (peak: 14.1M)
        CPU: 47ms
     CGroup: /system.slice/containerd.service
             └─4388 /usr/bin/containerd

Oct 03 12:03:56 ip-172-31-81-239 containerd[4388]: time="2024-10-03T12:03:56.810259144Z" level=info msg="Start subscrib>
Oct 03 12:03:56 ip-172-31-81-239 containerd[4388]: time="2024-10-03T12:03:56.810436084Z" level=info msg="Start recoveri>
Oct 03 12:03:56 ip-172-31-81-239 containerd[4388]: time="2024-10-03T12:03:56.810483329Z" level=info msg=serving... addr>
Oct 03 12:03:56 ip-172-31-81-239 containerd[4388]: time="2024-10-03T12:03:56.810489545Z" level=info msg="Start event mo>
Oct 03 12:03:56 ip-172-31-81-239 containerd[4388]: time="2024-10-03T12:03:56.810583963Z" level=info msg="Start snapshot>
Oct 03 12:03:56 ip-172-31-81-239 containerd[4388]: time="2024-10-03T12:03:56.810599063Z" level=info msg="Start cni netw>
Oct 03 12:03:56 ip-172-31-81-239 containerd[4388]: time="2024-10-03T12:03:56.810605397Z" level=info msg="Start streamin>
Oct 03 12:03:56 ip-172-31-81-239 containerd[4388]: time="2024-10-03T12:03:56.810635592Z" level=info msg=serving... addr>
Oct 03 12:03:56 ip-172-31-81-239 systemd[1]: Started containerd.service - containerd container runtime.
Oct 03 12:03:56 ip-172-31-81-239 containerd[4388]: time="2024-10-03T12:03:56.813064087Z" level=info msg="containerd suc>
```

sudo apt-get install -y socat

```
ubuntu@ip-172-31-81-239:~$ sudo apt-get install -y socat
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz
  slirp4netns
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  socat
0 upgraded, 1 newly installed, 0 to remove and 6 not upgraded.
Need to get 374 kB of archives.
After this operation, 1649 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 socat amd64 1.8.0.0-4build3 [374 kB]
Fetched 374 kB in 0s (12.2 MB/s)
Selecting previously unselected package socat.
(Reading database ... 68148 files and directories currently installed.)
Preparing to unpack .../socat_1.8.0.0-4build3_amd64.deb ...
Unpacking socat (1.8.0.0-4build3) ...
```

Step 6: Initialize the Kubecluster .Now Perform this Command only for Master.
sudo kubeadm init --pod-network-cidr=10.244.0.0/16

```
ubuntu@ip-172-31-82-119:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action beforehand using 'kubeadm config images pull'
W1003 12:37:31.290835   10158 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the containe
r runtime is inconsistent with that used by kubeadm.It is recommended to use "registry.k8s.io/pause:3.10" as the CRI san
dbox image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [ip-172-31-82-119 kubernetes kubernetes.default kubernetes.defaul
t.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 172.31.82.119]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [ip-172-31-82-119 localhost] and IPs [172.31.82.119 127.0.0.1 :
:1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-82-119 localhost] and IPs [172.31.82.119 127.0.0.1 ::1
```

Run this command on master and also copy and save the Join command from above.
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

```
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

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

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.31.82.119:6443 --token h4iisg.g9tdfmc88m9toefp \
        --discovery-token-ca-cert-hash sha256:5b3c7cfdf8115a26f1e73b752820d2b27b84ed476b344aaa3bd1a90e4b1f2105
ubuntu@ip-172-31-82-119:~$  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Step 7: Now Run the command kubectl get nodes to see the nodes before executing Join command on nodes.

```
ubuntu@ip-172-31-82-119:~$ kubectl get nodes
NAME               STATUS     ROLES           AGE   VERSION
ip-172-31-82-119   NotReady   control-plane   82s   v1.31.1
```

Step 8: Now Run the following command on Node 1 and Node 2 to Join to master.
sudo kubeadm join 172.31.82.119:6443 --token h4iisg.g9tdfmc88m9toefp \
--discovery-token-ca-cert-hash
sha256:5b3c7cfdf8115a26f1e73b752820d2b27b84ed476b344aaa3bd1a90e4b1f2105

```
ubuntu@ip-172-31-84-169:~$ sudo kubeadm join 172.31.82.119:6443 --token h4iisg.g9tdfmc88m9toefp \
        --discovery-token-ca-cert-hash sha256:5b3c7cfdf8115a26f1e73b752820d2b27b84ed476b344aaa3bd1a90e4b1f2105
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 501.315929ms
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

Step 9: Now Run the command kubectl get nodes to see the nodes after executing Join command on nodes.

```
ubuntu@ip-172-31-82-119:~$ kubectl get nodes
NAME                STATUS     ROLES           AGE     VERSION
ip-172-31-82-119    NotReady   control-plane   2m56s   v1.31.1
ip-172-31-84-169    NotReady   <none>          43s     v1.31.1
ip-172-31-87-189    NotReady   <none>          39s     v1.31.1
```

Step 10: Since Status is NotReady we have to add a network plugin. And also we have to give the name to the nodes.

kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml

```
ubuntu@ip-172-31-82-119:~$ kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
poddisruptionbudget.policy/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
serviceaccount/calico-node created
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/caliconodestatuses.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipreservations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
```

sudo systemctl status kubelet

```
ubuntu@ip-172-31-82-119:~$ sudo systemctl status kubelet
● kubelet.service - kubelet: The Kubernetes Node Agent
     Loaded: loaded (/usr/lib/systemd/system/kubelet.service; enabled; preset: enabled)
    Drop-In: /usr/lib/systemd/system/kubelet.service.d
             └─10-kubeadm.conf
     Active: active (running) since Thu 2024-10-03 12:37:52 UTC; 4min 4s ago
       Docs: https://kubernetes.io/docs/
   Main PID: 10849 (kubelet)
      Tasks: 10 (limit: 4676)
     Memory: 32.3M (peak: 33.0M)
        CPU: 5.030s
     CGroup: /system.slice/kubelet.service
             └─10849 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/k>

Oct 03 12:41:52 ip-172-31-82-119 kubelet[10849]: E1003 12:41:52.678397   10849 kuberuntime_container.go:851] "Kill cont>
Oct 03 12:41:52 ip-172-31-82-119 kubelet[10849]:         rpc error: code = Unknown desc = failed to kill container "233>
Oct 03 12:41:52 ip-172-31-82-119 kubelet[10849]:         : unknown
Oct 03 12:41:52 ip-172-31-82-119 kubelet[10849]:  > pod="kube-system/etcd-ip-172-31-82-119" podUID="201366195a044a56173>
Oct 03 12:41:52 ip-172-31-82-119 kubelet[10849]: E1003 12:41:52.688038   10849 log.go:32] "StopPodSandbox from runtime >
Oct 03 12:41:52 ip-172-31-82-119 kubelet[10849]:         rpc error: code = Unknown desc = failed to stop container "233>
Oct 03 12:41:52 ip-172-31-82-119 kubelet[10849]:         : unknown
```

Now Run command kubectl get nodes -o wide we can see Status is ready.

```
ubuntu@ip-172-31-82-119:~$ kubectl get nodes -o wide
NAME               STATUS   ROLES           AGE     VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-
VERSION   CONTAINER-RUNTIME
ip-172-31-82-119   Ready    control-plane   4m32s   v1.31.1   172.31.82.119   <none>        Ubuntu 24.04.1 LTS   6.8.0-1
016-aws   containerd://1.7.12
ip-172-31-84-169   Ready    <none>          2m19s   v1.31.1   172.31.84.169   <none>        Ubuntu 24.04.1 LTS   6.8.0-1
016-aws   containerd://1.7.12
ip-172-31-87-189   Ready    <none>          2m15s   v1.31.1   172.31.87.189   <none>        Ubuntu 24.04.1 LTS   6.8.0-1
016-aws   containerd://1.7.12
```

Now to Rename run this command kubectl label node ip-172-31-18-135 kubernetes.io/role=worker Rename to Node 1:kubectl label node ip-172-31-28-117 kubernetes.io/role=Node1 Rename to Node 2:kubectl label node ip-172-31-18-135 kubernetes.io/role=Node2

Step 11: Run command kubectl get nodes -o wide . And Hence we can see we have Successfully connected Node 1 and Node 2 to the Master.

```
ubuntu@ip-172-31-82-119:~$ kubectl get nodes
NAME               STATUS   ROLES           AGE     VERSION
ip-172-31-82-119   Ready    control-plane   6m52s   v1.31.1
ip-172-31-84-169   Ready    Node1           4m39s   v1.31.1
ip-172-31-87-189   Ready    Node2           4m35s   v1.31.1
```

Conclusion:

In this experiment, we successfully set up a Kubernetes cluster on AWS EC2 instances. We began by provisioning three EC2 instances—Master, Node1, and Node2—with appropriate security rules to facilitate communication within the cluster. After securely SSHing into each instance, we installed Docker as the container runtime on all machines. We proceeded by installing Kubernetes on each instance and initializing the cluster from the master node. Following cluster initialization, the worker nodes were connected to the master using the kubeadm join command, effectively forming a cohesive Kubernetes cluster. Additionally, we implemented a network plugin to ensure proper communication between nodes, allowing the nodes' status to transition to "Ready."