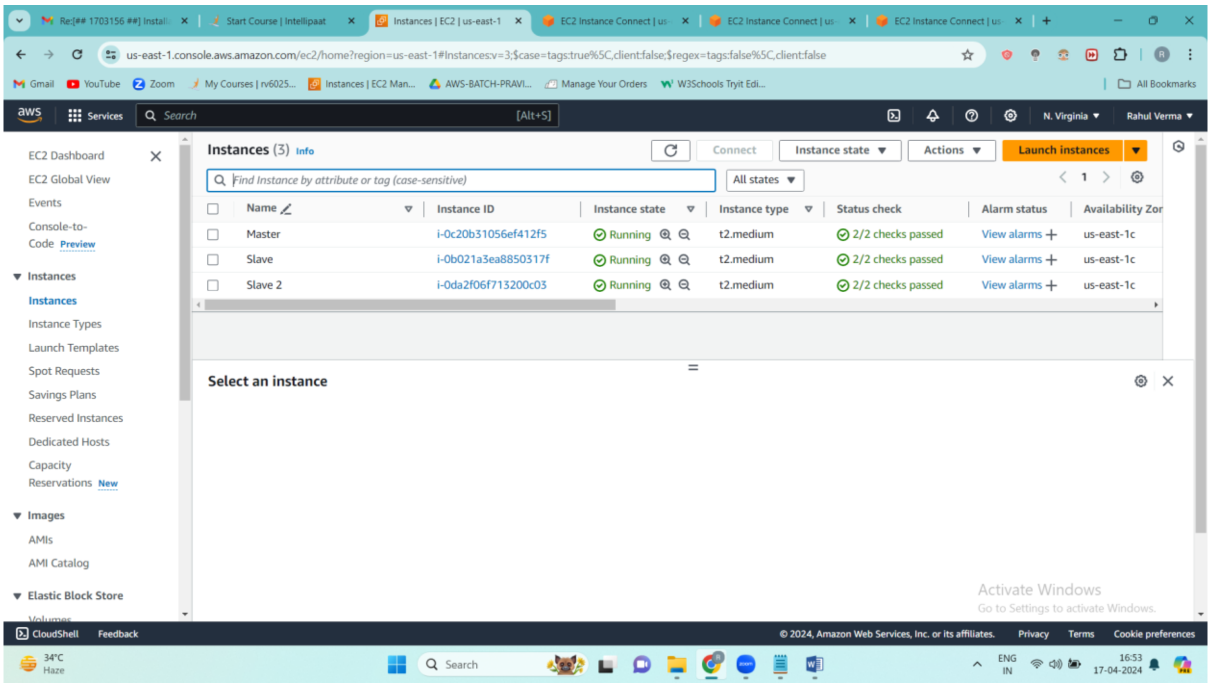
Hands-on

Installation of Kubernetes

Prerequisite:

Create 3 ec2 instances on any region with t2.medium instance type with Ubuntu OS.

1. Master
2. Slave
3. Slave 2



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zor
Master	i-0c20b31056ef412f5	Running	t2.medium	2/2 checks passed	View alarms	us-east-1c
Slave	i-0b021a3ea8850317f	Running	t2.medium	2/2 checks passed	View alarms	us-east-1c
Slave 2	i-0da2f06f713200c03	Running	t2.medium	2/2 checks passed	View alarms	us-east-1c

- Make sure all traffic is allowed on both the instance

Step 1: connect to your instances, **Execute on all the instances i.e "Master" , "Slave" and "Slave 2"**

Write-

```
sudo apt update
```

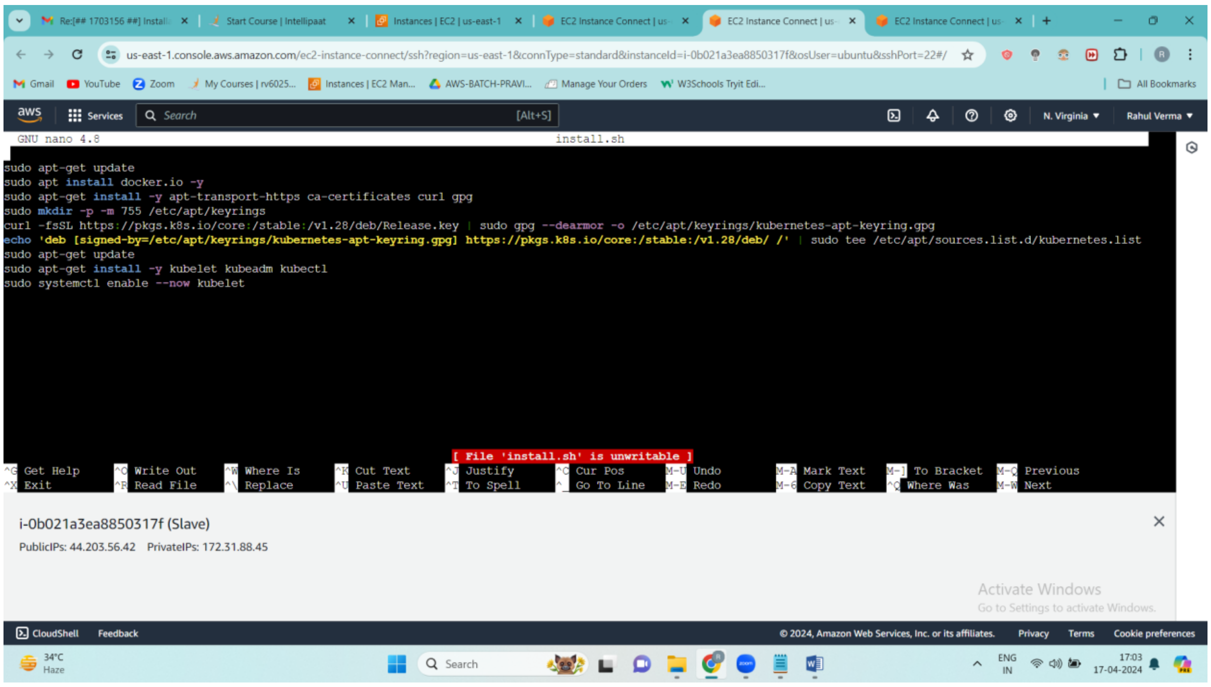
`sudo nano install.sh` (this will open nano editor)

now copy the below commands-

```
sudo apt-get update

sudo apt install docker.io -y

sudo apt-get install -y apt-transport-https ca-certificates curl gpg

sudo mkdir -p -m 755 /etc/apt/keyrings

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/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:/v1.28/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

sudo systemctl enable --now kubelet
```

Paste it in nano editor. Save and exit from it

Press ctrl+s and ctrl+x



```
GNU nano 4.8                                        install.sh

sudo apt-get update
sudo apt install docker.io -y
sudo apt-get install -y apt-transport-https ca-certificates curl gpg
sudo mkdir -p -m 755 /etc/apt/keyrings
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/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:/v1.28/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo systemctl enable --now kubelet

                                        [ File 'install.sh' is unwritable ]
^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo       M-A Mark Text   M-] To Bracket  M-Q Previous
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo       M-6 Copy Text   ^Q Where Was    M-W Next
```

i-0b021a3ea8850317f (Slave)

PublicIPs: 44.203.56.42   PrivateIPs: 172.31.88.45

Write the below command, it execute all the commands from our install.sh file

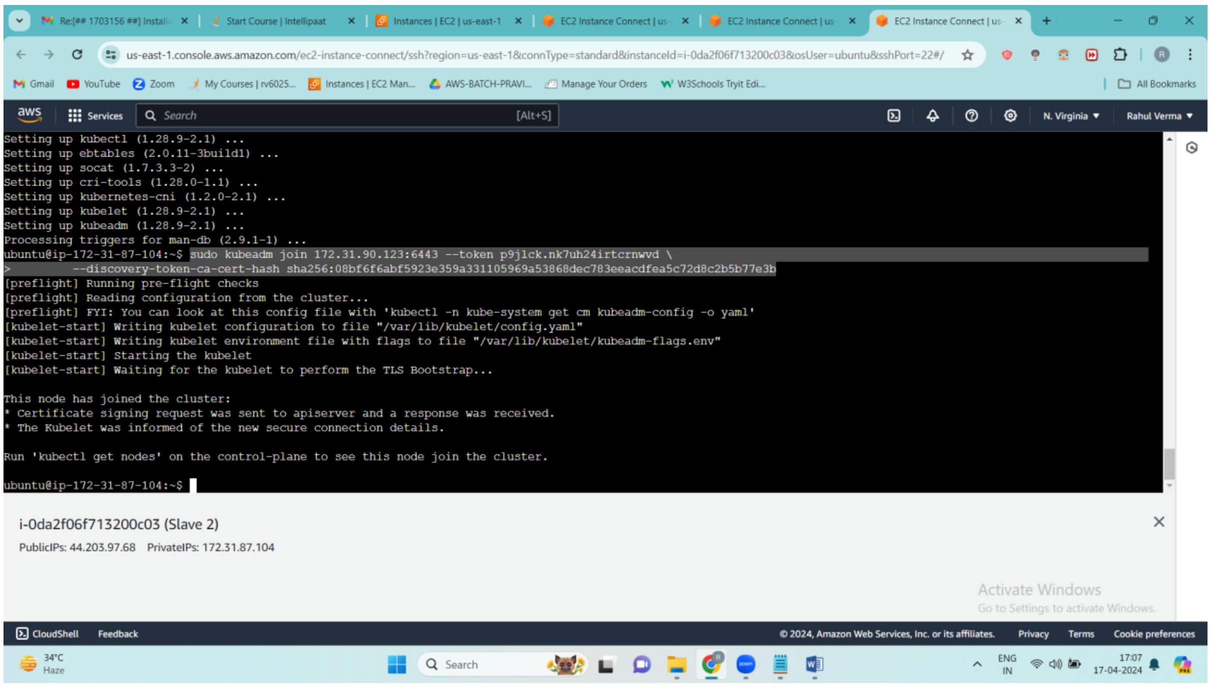
bash install.sh

Step 2: Execute this command on master NODE only

sudo kubeadm init --apiserver-advertise-address=privateipofmaster

Note: You need to replace “private_ip_of_master” with the actual private ip of your kubernetes master.

Paste the Token on Slave and Slave2



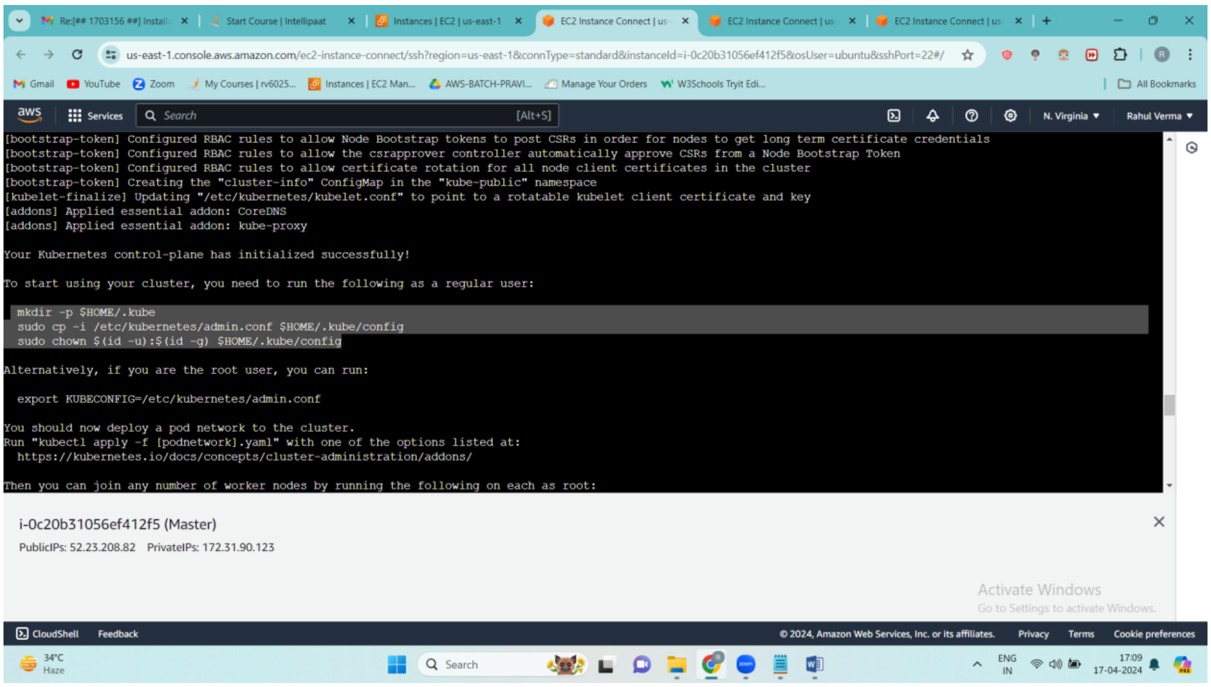
```
Setting up kubectl (1.28.9-2.1) ...
Setting up ebtables (2.0.11-3build1) ...
Setting up socat (1.7.3.3-2) ...
Setting up cri-tools (1.28.0-1.1) ...
Setting up kubernetes-cni (1.2.0-2.1) ...
Setting up kubelet (1.28.9-2.1) ...
Setting up kubeadm (1.28.9-2.1) ...
Processing triggers for man-db (2.9.1-1) ...
ubuntu@ip-172-31-87-104:~$ sudo kubeadm join 172.31.90.123:6443 --token p9jlck.nk7uh24irtcrnwvd \
>         --discovery-token-ca-cert-hash sha256:08bf6f6abf5923e359a331105969a53868dec783eeacdfea5c72d8c2b5b77e3b
[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-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.

ubuntu@ip-172-31-87-104:~$
```

i-0da2f06f713200c03 (Slave 2)

PublicIPs: 44.203.97.68   PrivateIPs: 172.31.87.104

Step3: Execute on Master node only

Copy the highlighted commands and paste it in master node one by one



```
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials
[bootstrap-token] Configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token
[bootstrap-token] Configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

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:
```

i-0c20b31056ef412f5 (Master)

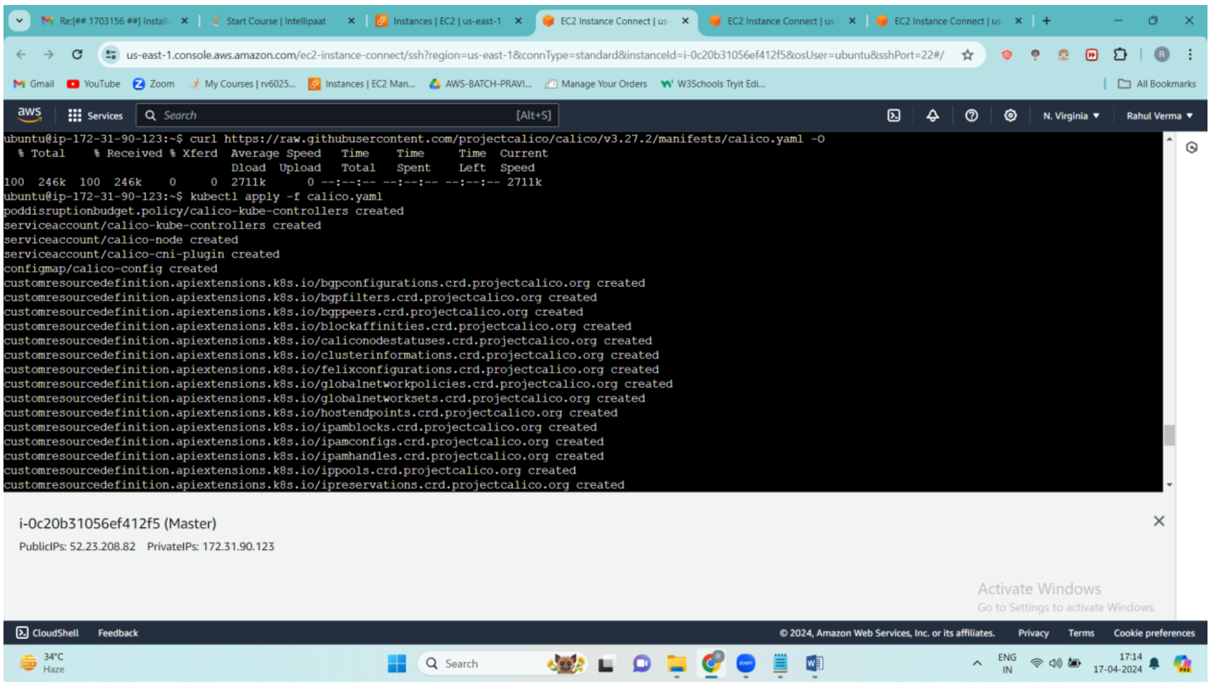
PublicIPs: 52.23.208.82   PrivateIPs: 172.31.90.123

```
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
```

Step 4: Execute on master node only - **Installing Calico**

curl https://raw.githubusercontent.com/projectcalico/calico/v3.27.2/manifests/calico.yaml -O

kubectl apply -f calico.yaml



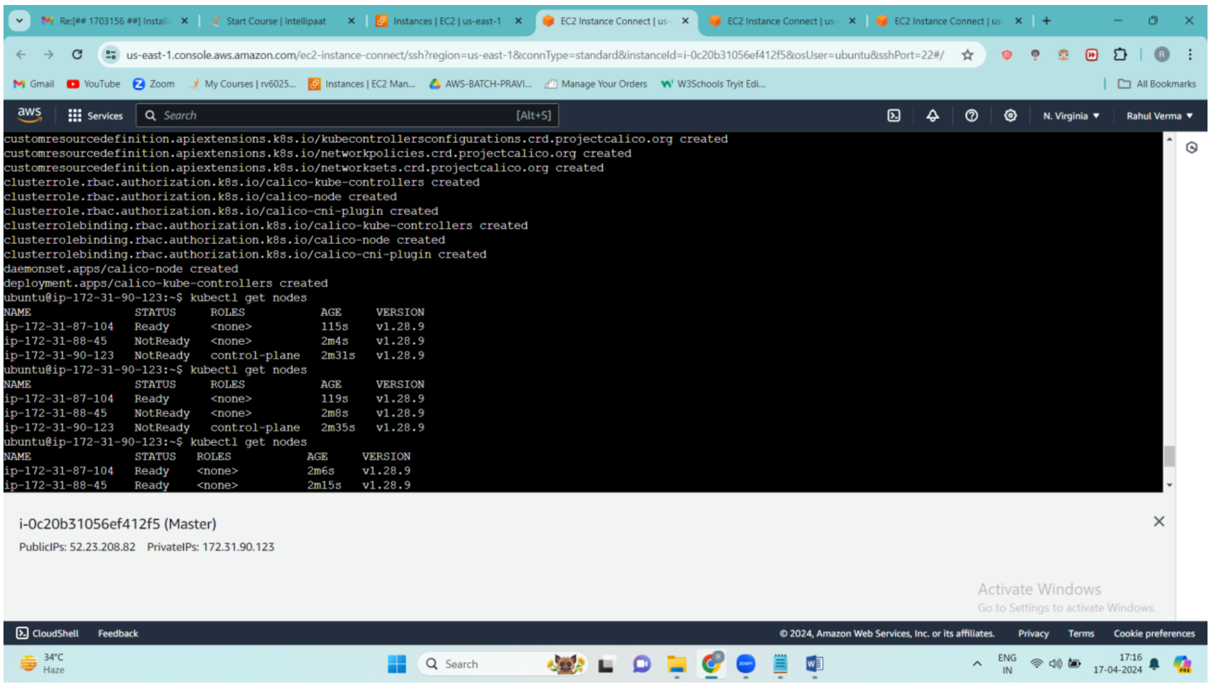
```
ubuntu@ip-172-31-90-123:~$ curl https://raw.githubusercontent.com/projectcalico/calico/v3.27.2/manifests/calico.yaml -O
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  246k  100  246k    0     0  2711k      0 --:--:-- --:--:-- --:--:-- 2711k
ubuntu@ip-172-31-90-123:~$ kubectl apply -f calico.yaml
poddisruptionbudget.policy/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
serviceaccount/calico-node created
serviceaccount/calico-cni-plugin created
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgpfilters.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
```

i-0c20b31056ef412f5 (Master)

PublicIPs: 52.23.208.82 PrivateIPs: 172.31.90.123

Now to verify write the below command-

kubectl get nodes



```
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
clusterrole.rbac.authorization.k8s.io/calico-cni-plugin created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-cni-plugin created
daemonset.apps/calico-node created
deployment.apps/calico-kube-controllers created
ubuntu@ip-172-31-90-123:~$ kubectl get nodes
NAME               STATUS     ROLES           AGE     VERSION
ip-172-31-87-104   Ready      <none>          115s    v1.28.9
ip-172-31-88-45    NotReady   <none>          2m4s    v1.28.9
ip-172-31-90-123   NotReady   control-plane   2m31s   v1.28.9
ubuntu@ip-172-31-90-123:~$ kubectl get nodes
NAME               STATUS     ROLES           AGE     VERSION
ip-172-31-87-104   Ready      <none>          119s    v1.28.9
ip-172-31-88-45    NotReady   <none>          2m8s    v1.28.9
ip-172-31-90-123   NotReady   control-plane   2m35s   v1.28.9
ubuntu@ip-172-31-90-123:~$ kubectl get nodes
NAME               STATUS   ROLES           AGE     VERSION
ip-172-31-87-104   Ready    <none>          2m6s    v1.28.9
ip-172-31-88-45    Ready    <none>          2m15s   v1.28.9
```

i-0c20b31056ef412f5 (Master)

PublicIPs: 52.23.208.82   PrivateIPs: 172.31.90.123