**Networking in K8s**

**Process:**

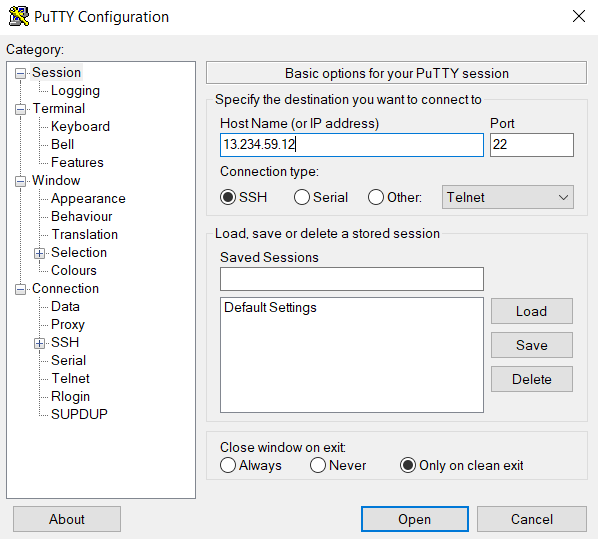
**Set Up AWS Resources**

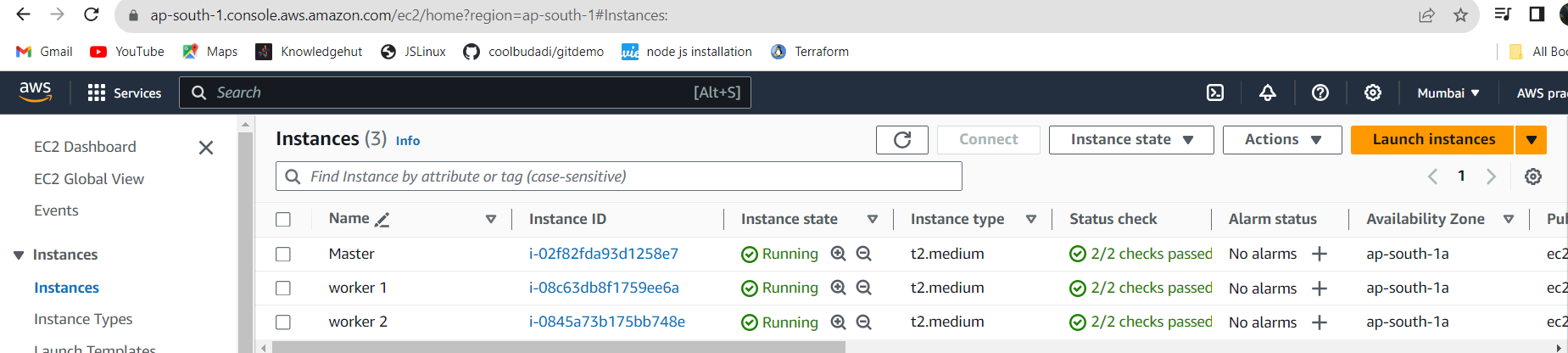
Logged in to AWS Management Console.  
• Launched EC2 Instance: (virtual machine - Ubuntu).

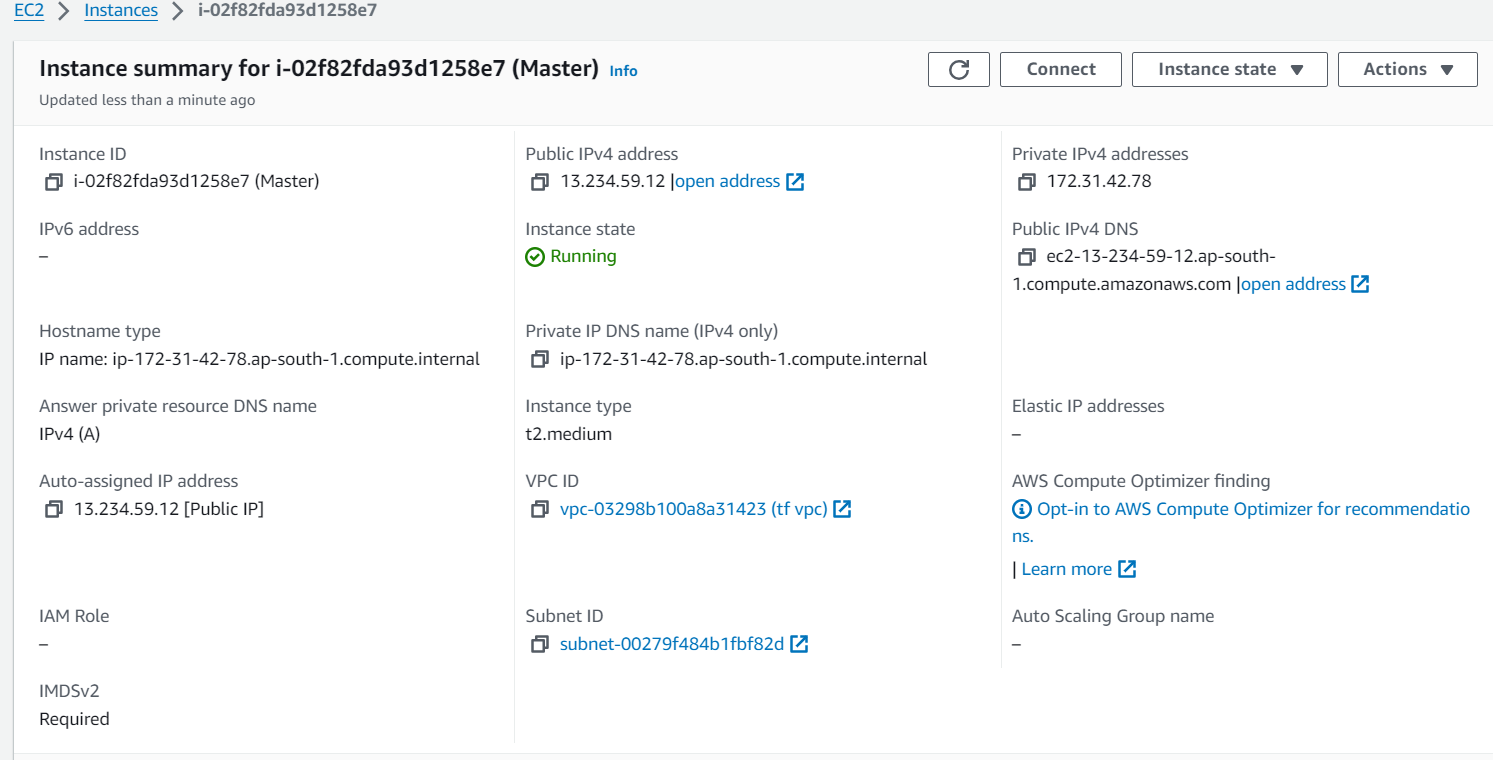
And security groups to allowed SSH (port 22) traffic with instance type - T2.medium and 15Gb disk size and created PPK file.

**Connect to Your EC2 Instance**

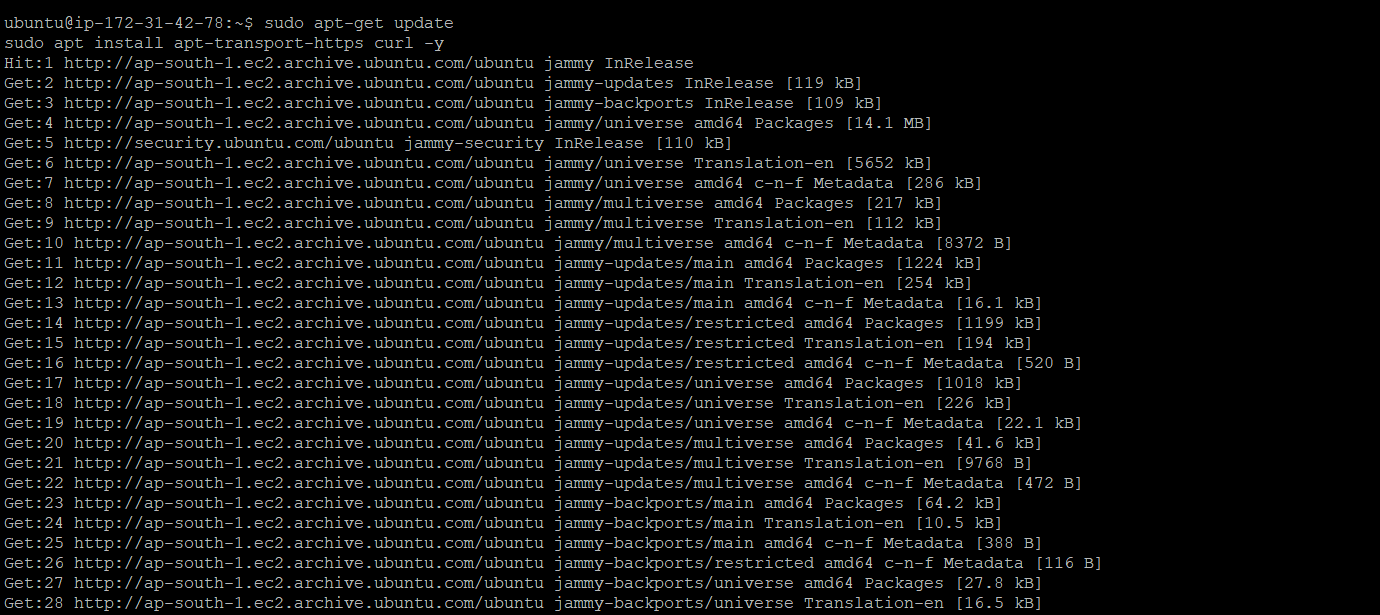
•    Used SSH to connect to EC2 instance using the PPK file via Putty as below.



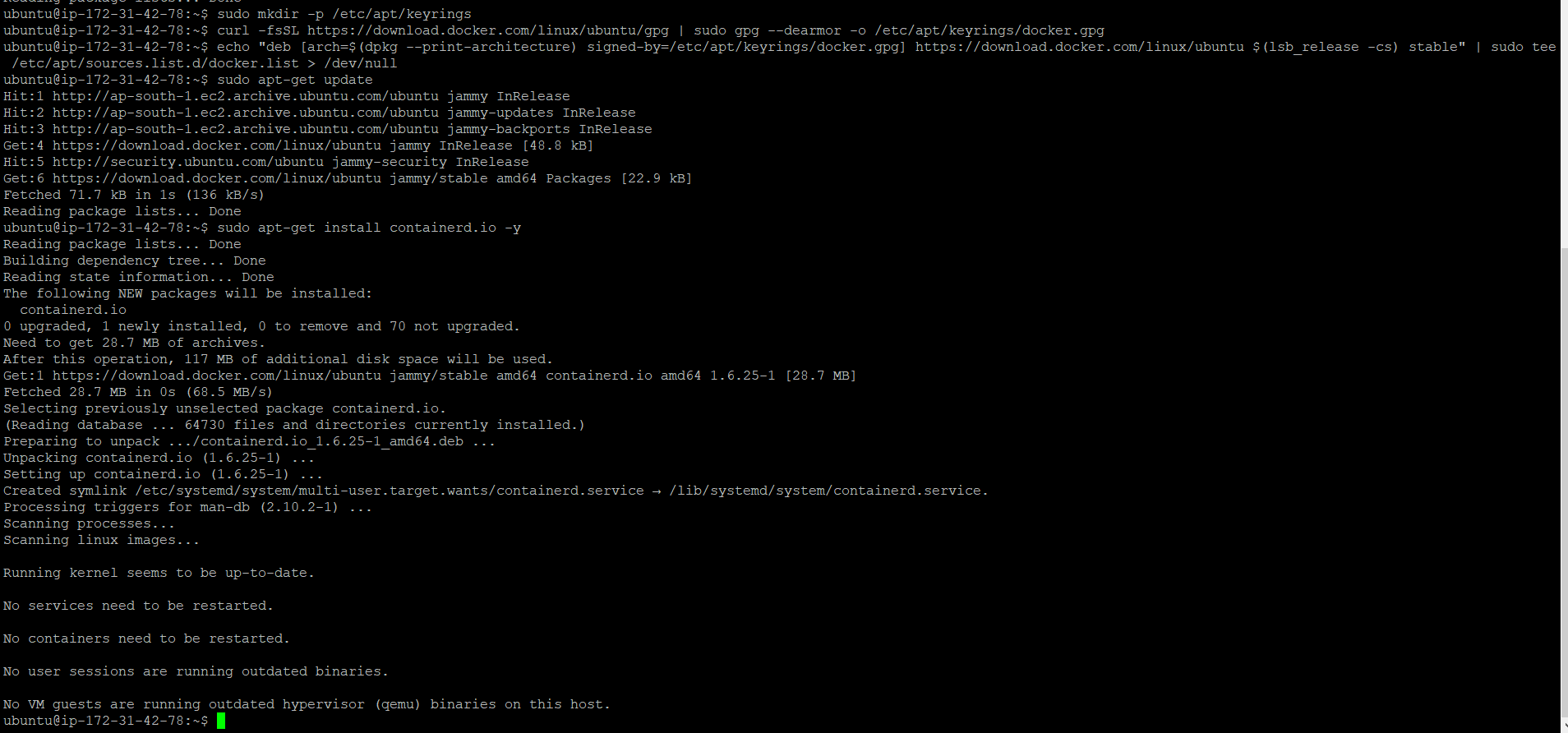




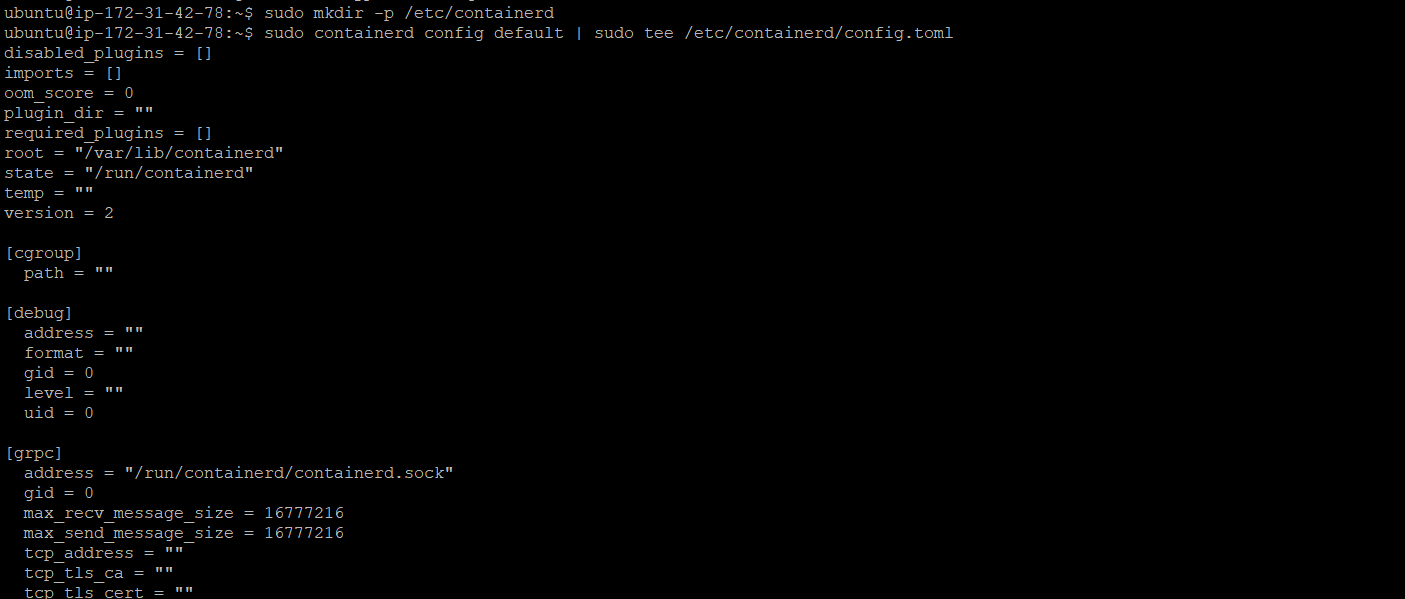
Updated the system's package list and install necessary dependencies using the following commands on three (Master and worker machines)



Installed Containerd, used the following commands(Applied to all machine)



created the containerd configuration file using the following commands:(On three machine)



Edited the containerd configuration file to set SystemdCgroup to true. Used the following command to open the file:(Apply to all machine)

Setted SystemdCgroup to true:

SystemdCgroup = true





Restart containerd(Over all machines)

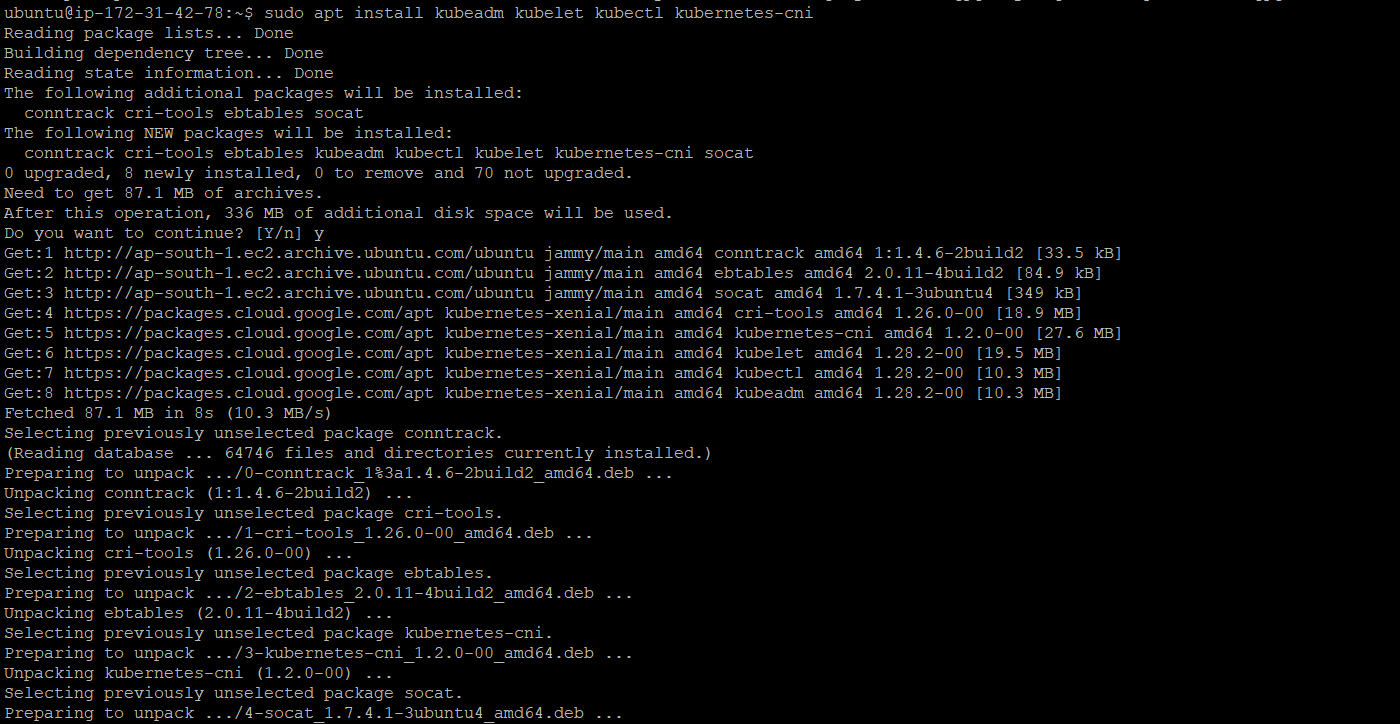


Installed Kubernetes on three machines(Master and worker nodes), use the following commands:

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add

sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"

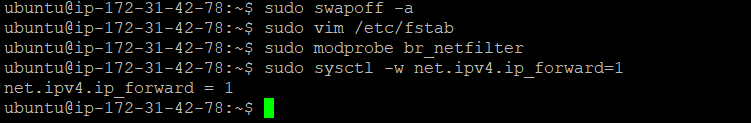
sudo apt install kubeadm kubelet kubectl kubernetes-cni



Disabled swap on master machine: :(Apply on three machines)

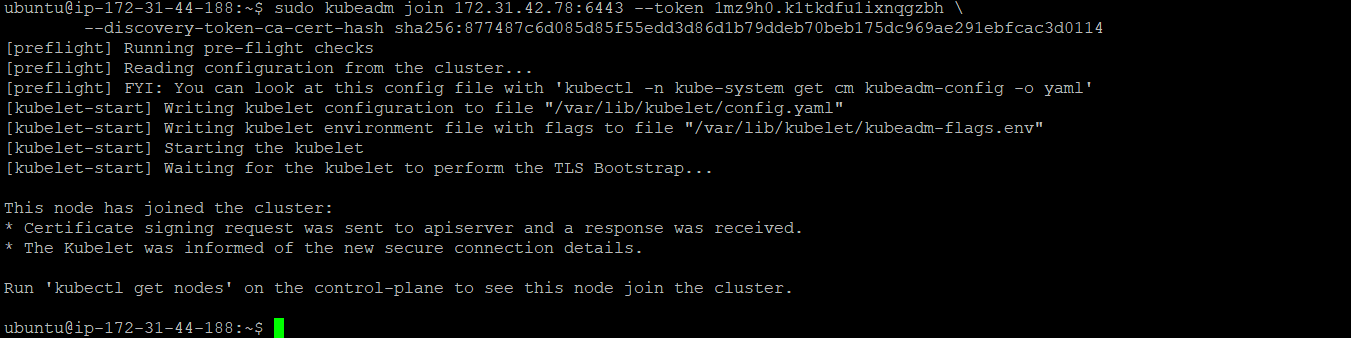
Enabled kernel modules using following command:(Apply on three machines)

* sudo modprobe br\_netfilter
* Added some settings to sysctl using below command: (Apply on three machines)



## Initialized the Cluster (only on master) using following command:

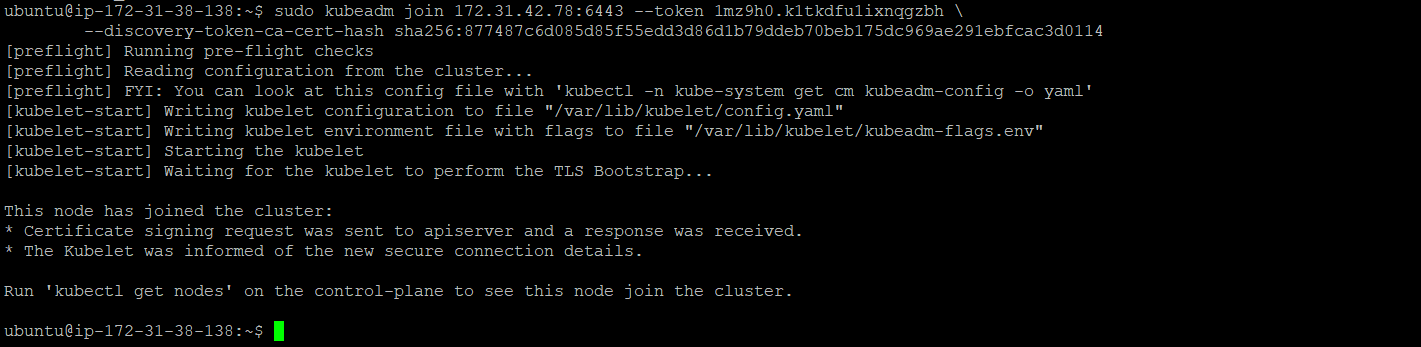
## sudo kubeadm init --pod-network-cidr=10.244.0.0/16



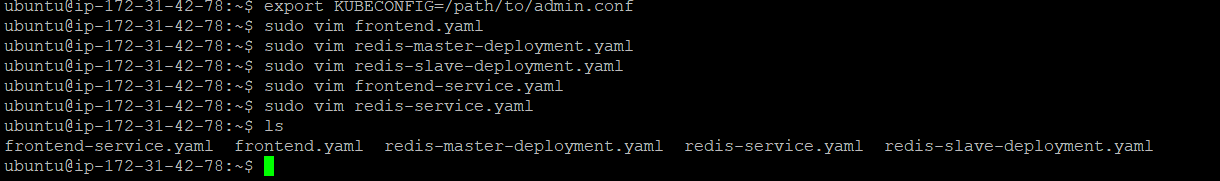
**Token :**

kubeadm join 172.31.42.78:6443 --token 1mz9h0.k1tkdfu1ixnqgzbh \

--discovery-token-ca-cert-hash sha256:877487c6d085d85f55edd3d86d1b79ddeb70beb175dc969ae291ebfcac3d0114



Created yaml files as below:



Frontend-deployment.yaml:

apiVersion: apps/v1

kind: Deployment

metadata:

name: php-frontend

spec:

replicas: 3

selector:

matchLabels:

app: php-frontend

template:

metadata:

labels:

app: php-frontend

spec:

containers:

- name: php-frontend

image: gcr.io/google\_samples/gb-frontend:v5

redis-master-deployment.yaml:

apiVersion: apps/v1

kind: Deployment

metadata:

name: redis-master

spec:

replicas: 1

selector:

matchLabels:

app: redis-master

template:

metadata:

labels:

app: redis-master

spec:

containers:

- name: redis-master

image: gcr.io/google\_samples/gb-redis-follower:v2

command: ["redis-server", "--slaveof", "NO ONE"]

redis-slave-deployment.yaml:

apiVersion: apps/v1

kind: Deployment

metadata:

name: redis-slave

spec:

replicas: 2

selector:

matchLabels:

app: redis-slave

template:

metadata:

labels:

app: redis-slave

spec:

containers:

- name: redis-slave

image: gcr.io/google\_samples/gb-redis-follower:v2

command: ["redis-server", "--slaveof", "redis-master", "6379"]

Frontend-service.yaml :

apiVersion: v1

kind: Service

metadata:

name: php-frontend-service

spec:

selector:

app: php-frontend

ports:

- protocol: TCP

port: 80

targetPort: 80

type: LoadBalancer

redis-service.yaml:

apiVersion: v1

kind: Service

metadata:

name: redis-service

spec:

selector:

app: redis-master

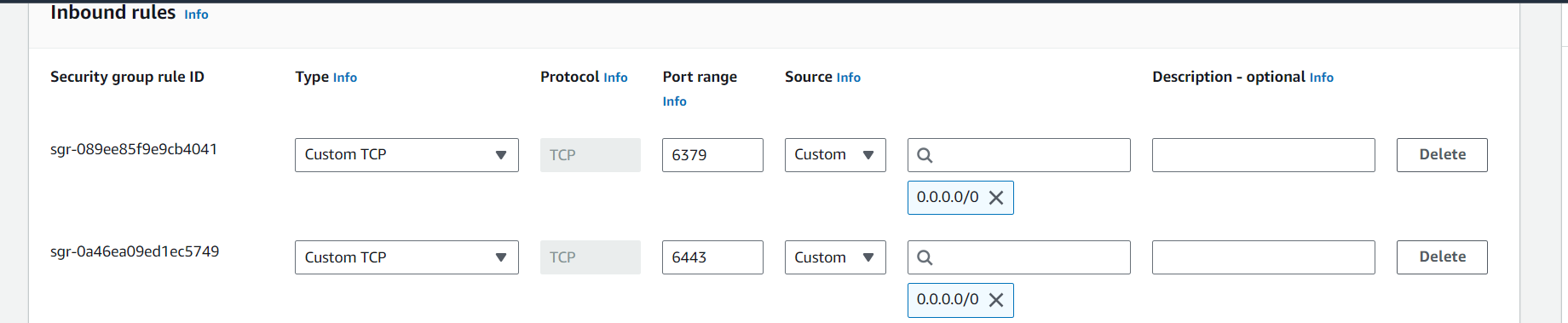
ports:

- protocol: TCP

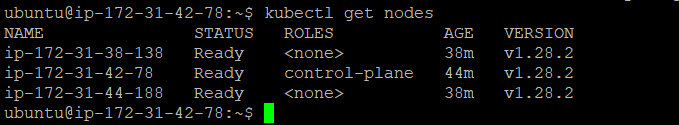
port: 6379

targetPort: 6379

Added port 6379 entry to the master machine (Redis)



After successful run we got the nodes on master machine and ensured that the 3 cluster node cluster created properly:



Applied all yaml files for deployment creation we can see the nodes ,deployments running successfully as below screenshot:

