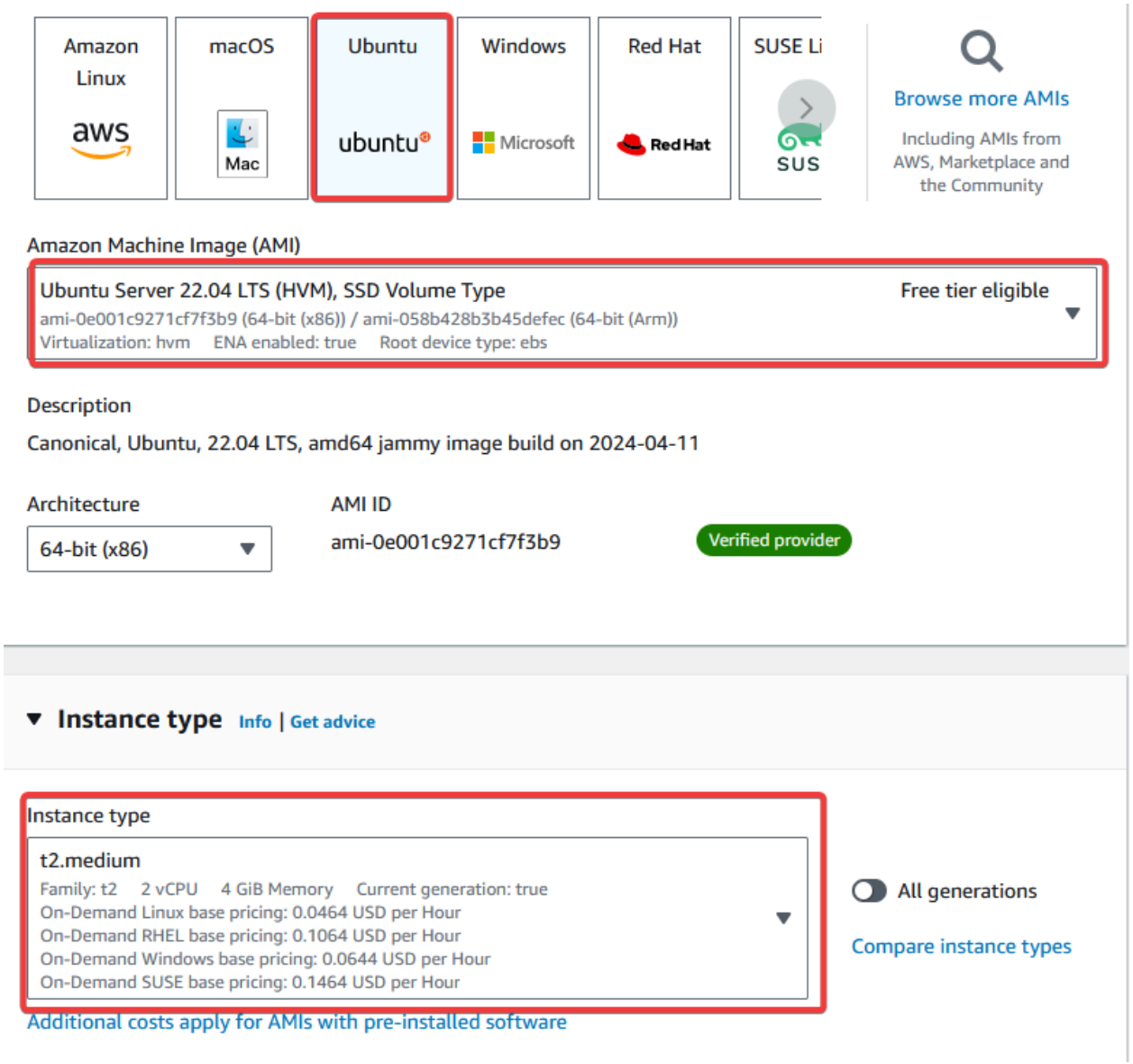
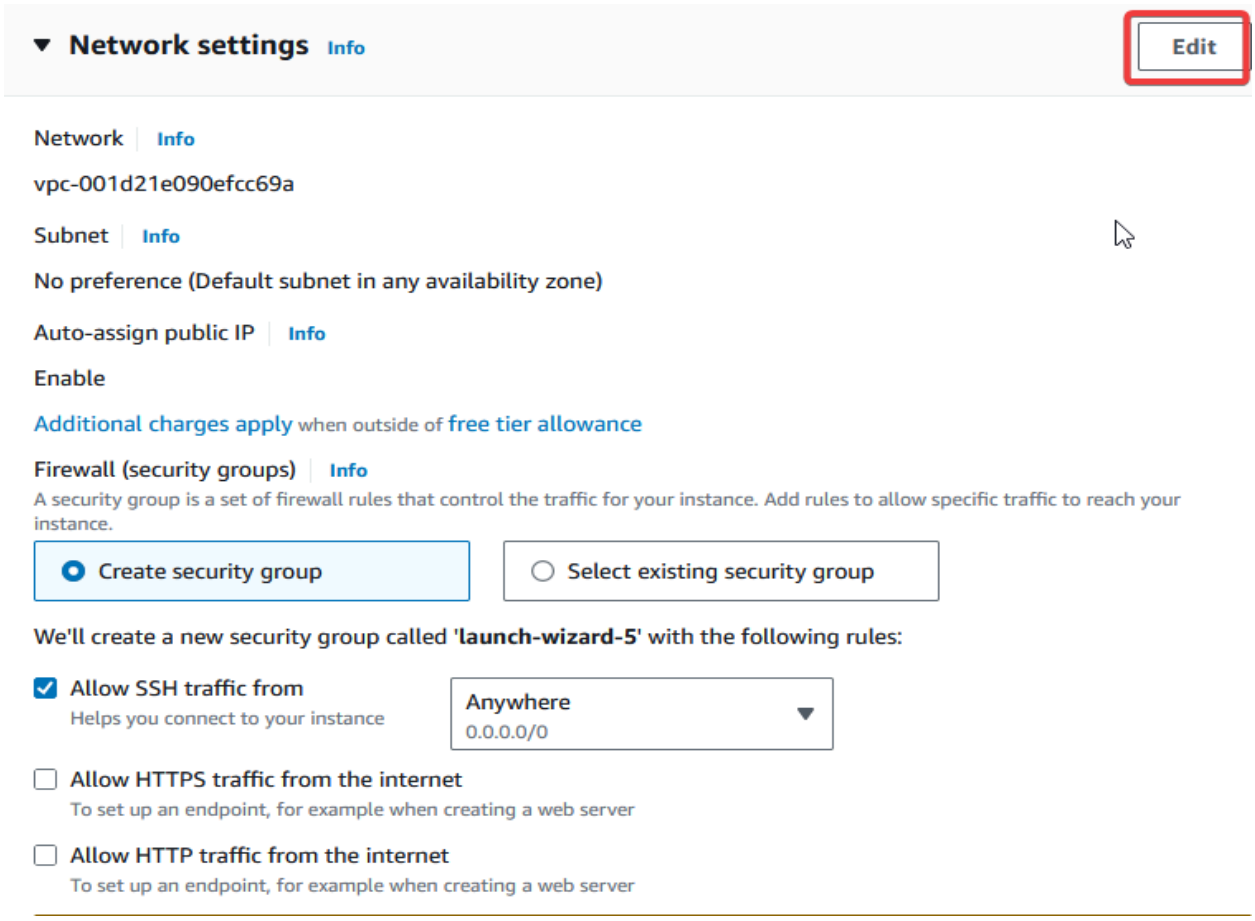
Setting up K8s cluster on ubuntu 22.04 using kubeadm tool

First you need to create ubuntu 22.04 ec2-instance on aws

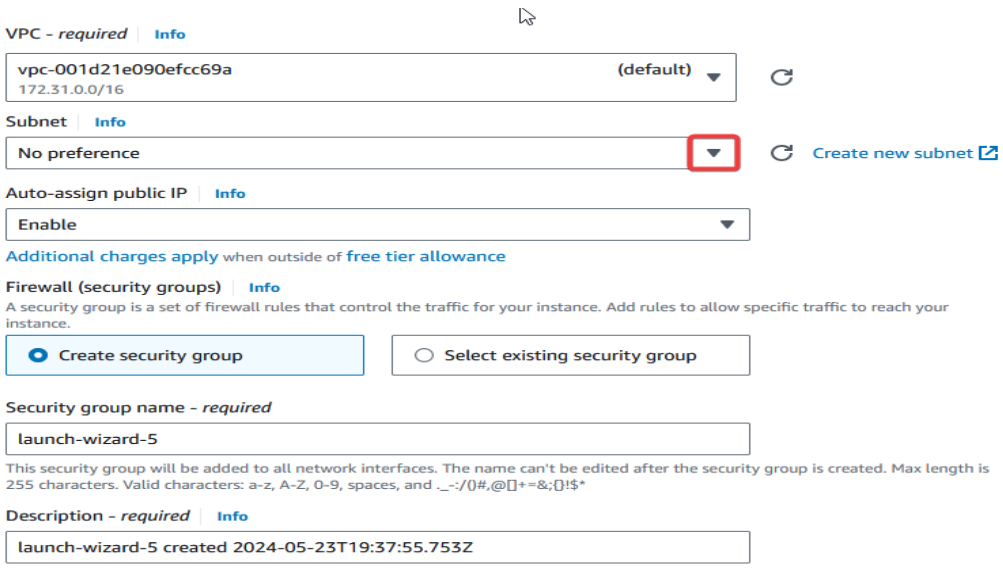
Ec2-instance configuration for master node

AWS AMI ==> Ubuntu 22.04 LTS
Instance type ==> t2.medium
Storage ==> 25 GB

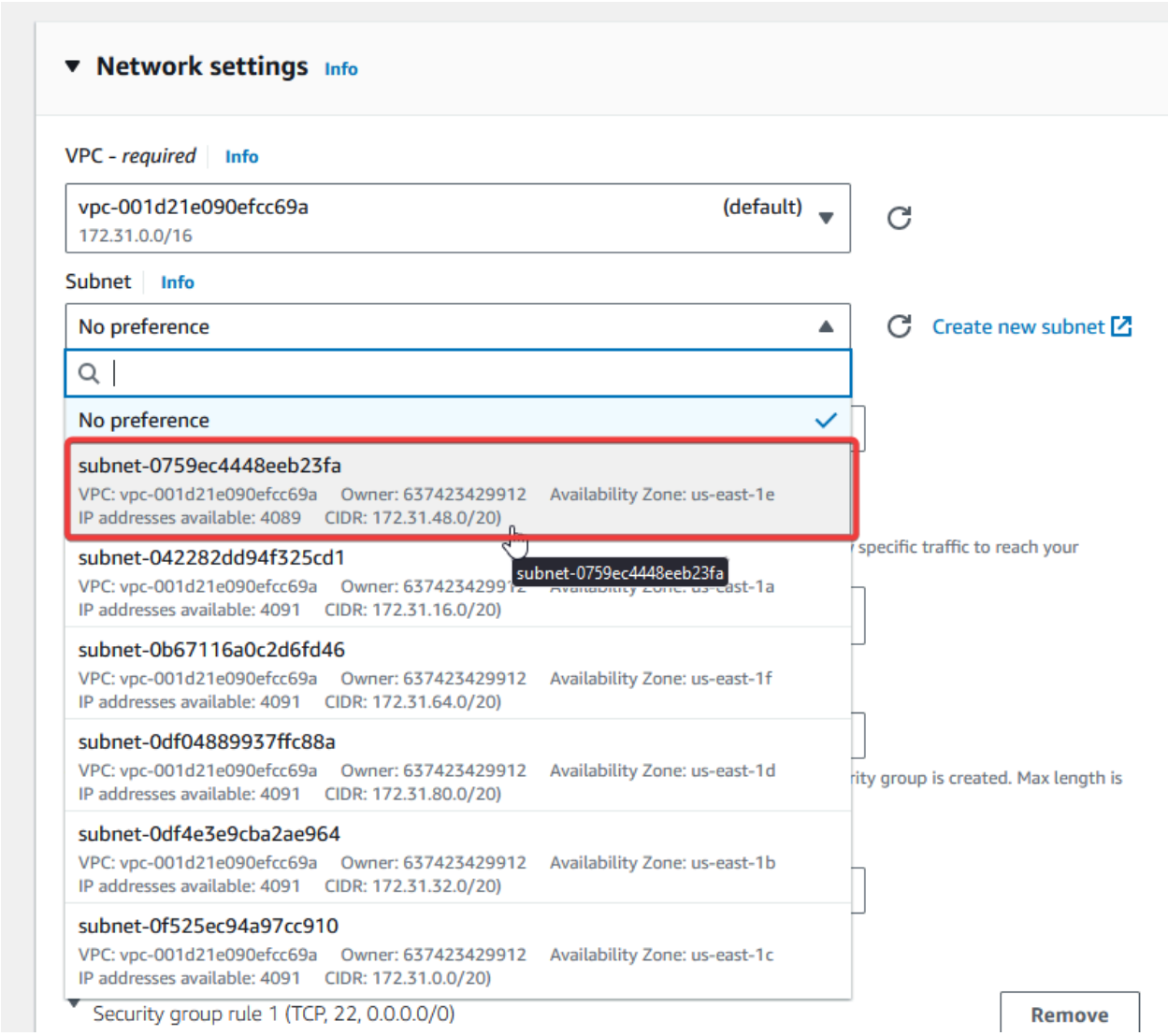




Click on edit and select on

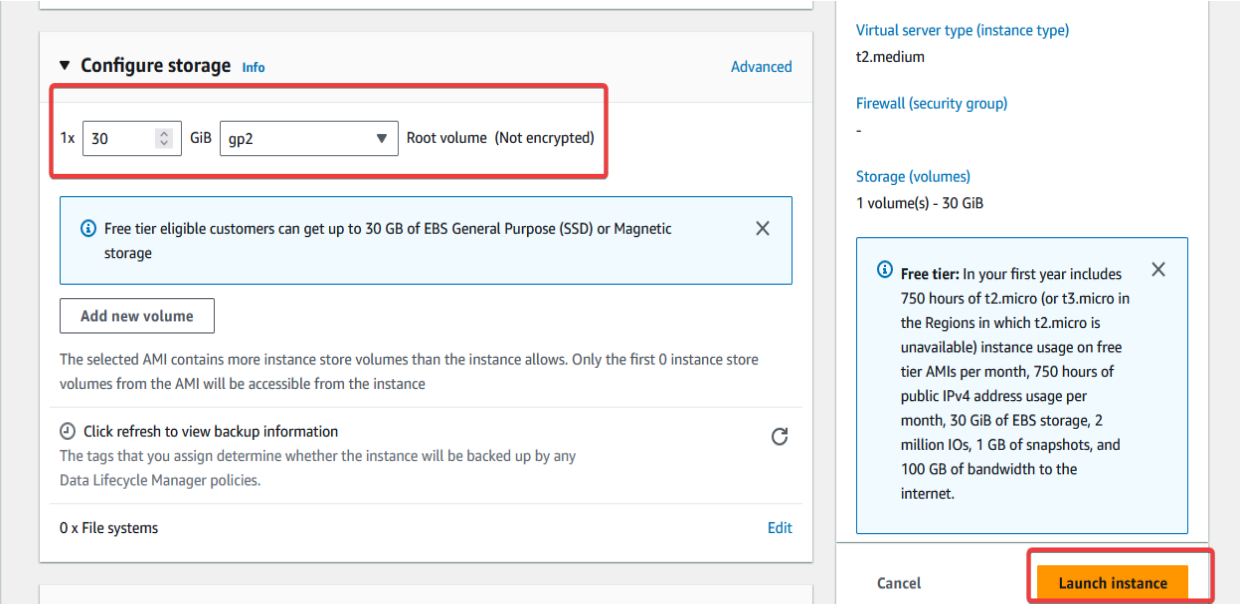


Click on drop down and select subnet



Note:- You need to remember the subnet you have selected because we will use the same subnet for the worker node as well, so that communication between the master and worker nodes can occur.

After this configure your storage and click on launch instance



After launching your master node instance, connect to it via SSH. You can also use software like Mobaxterm or PuTTY for this.

Now, we begin configuring our master node.

Now, we need to load the kernel modules for Kubernetes: overlay and br_netfilter. To load these modules temporarily, run the following commands:

```
# Enabling kernel modules (overlay and br_netfilter)
sudo modprobe overlay
sudo modprobe br_netfilter
```

To load the modules permanently, create a config file in /etc/modules-load.d. Run the following command:

```
[root@k8master ~]# vim /etc/modules-load.d/k8s.conf
```

In this file, enter the following two lines:

```
overlay
br_netfilter
```

To verify the config file, run:

```
[root@k8master ~]# cat /etc/modules-load.d/k8s.conf
overlay
br_netfilter
```

Run this command to verify whether modules are loaded or not

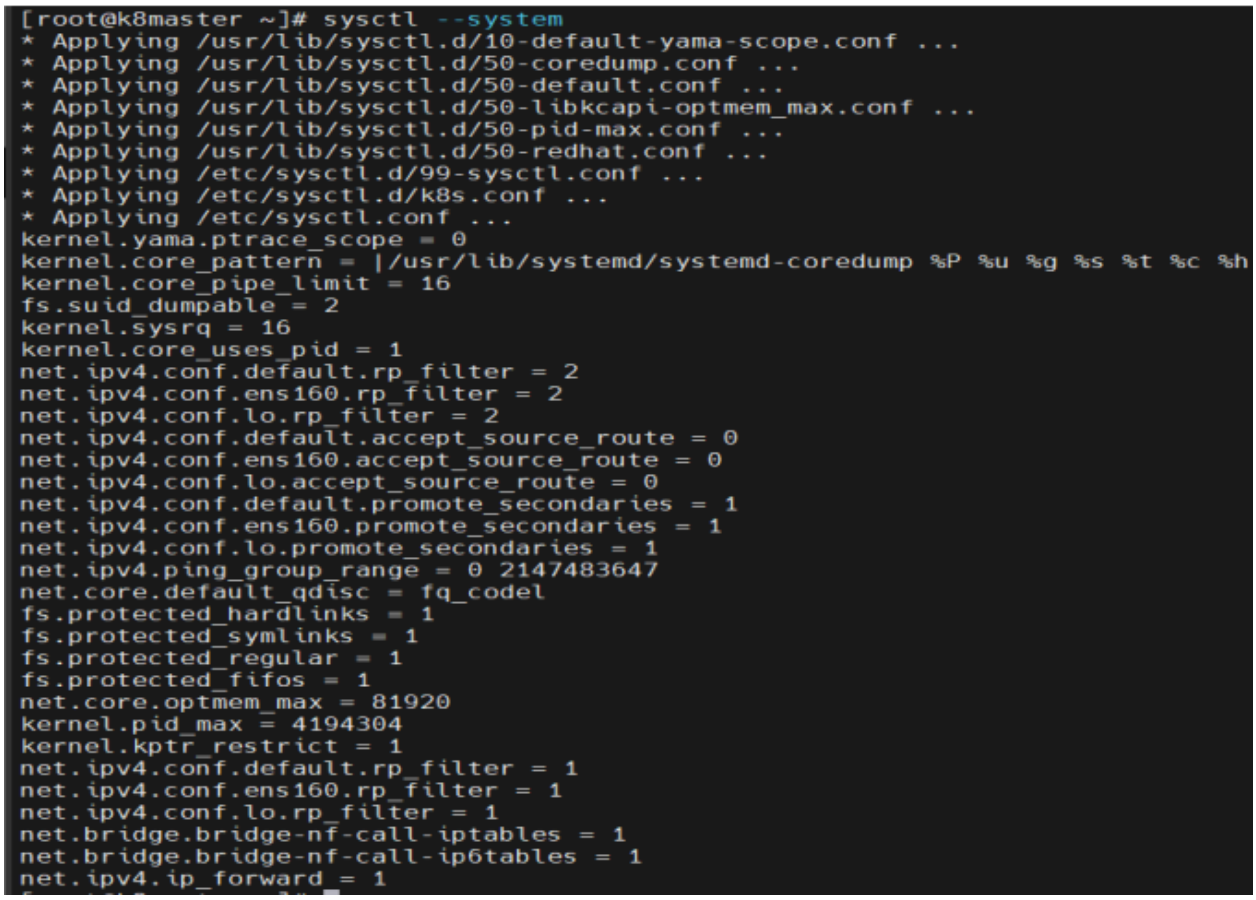
[root@k8master ~]# lsmod | grep br_
[root@k8master ~]# lsmod | grep over

To create a configuration file for iptables, run the following command in your terminal:

[root@k8master ~]# cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward = 1
EOF

To verify whether rules are added or not run this command

[root@k8master ~]# sysctl --system



Disabling Swap Memory

Run this command to disable swap memory
[root@k8master ~]# swapoff -a

To disable swap permanently, open the /etc/fstab file and comment out the swap entry.

```
#
/dev/mapper/rl-root     /                       xfs     defaults        0 0
UUID=63aa319a-771c-4d57-b812-d1c51d5c34a6 /boot                   xfs     defaults        0 0
#/dev/mapper/rl-swap     none                    swap    defaults        0 0
```

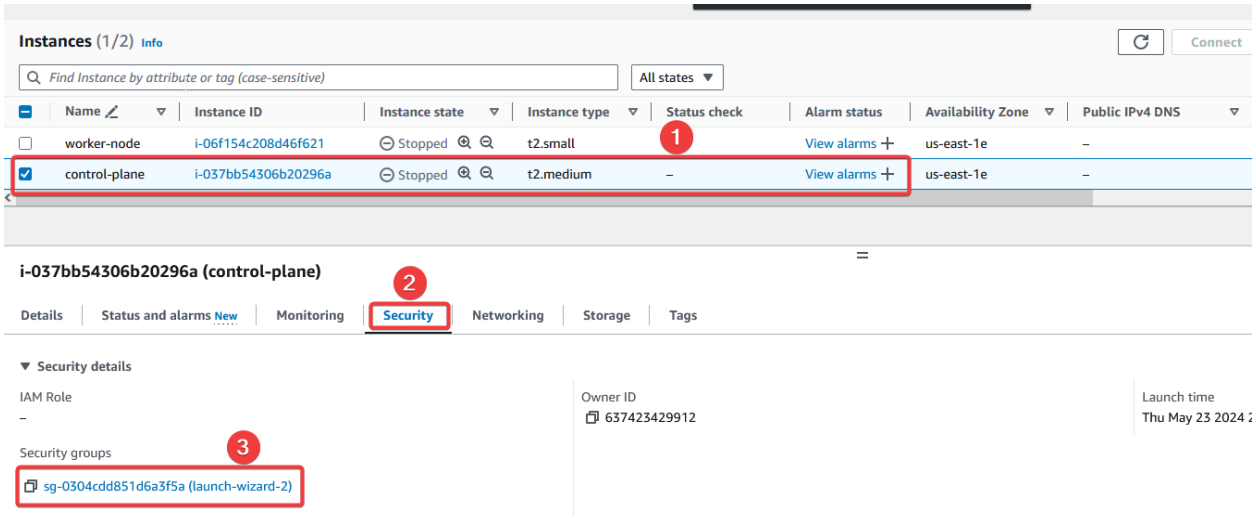
Now, we need to add ports to our firewall or AWS security group. The Kubernetes master node requires the following ports:

TCP Ports: 6443, 2379, 2380, 10250, 10251, 10252, 10257, 10259, 179

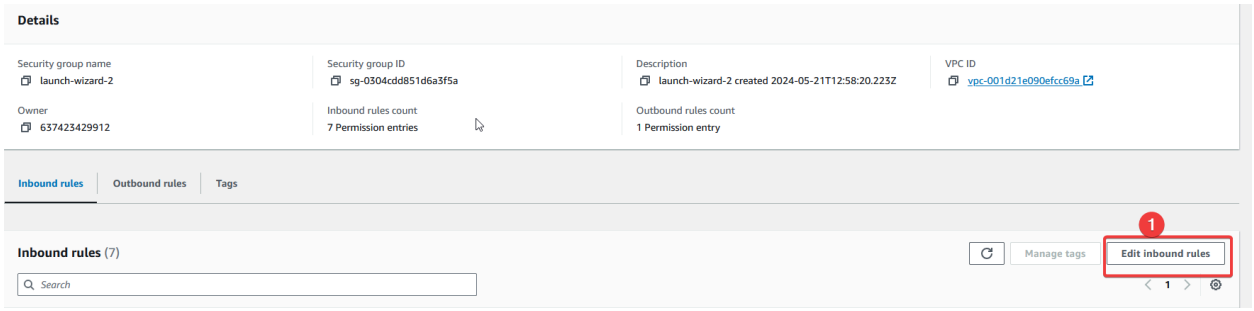
UDP Port: 4789

In my case, I'm using an AWS security group to allow Kubernetes ports.

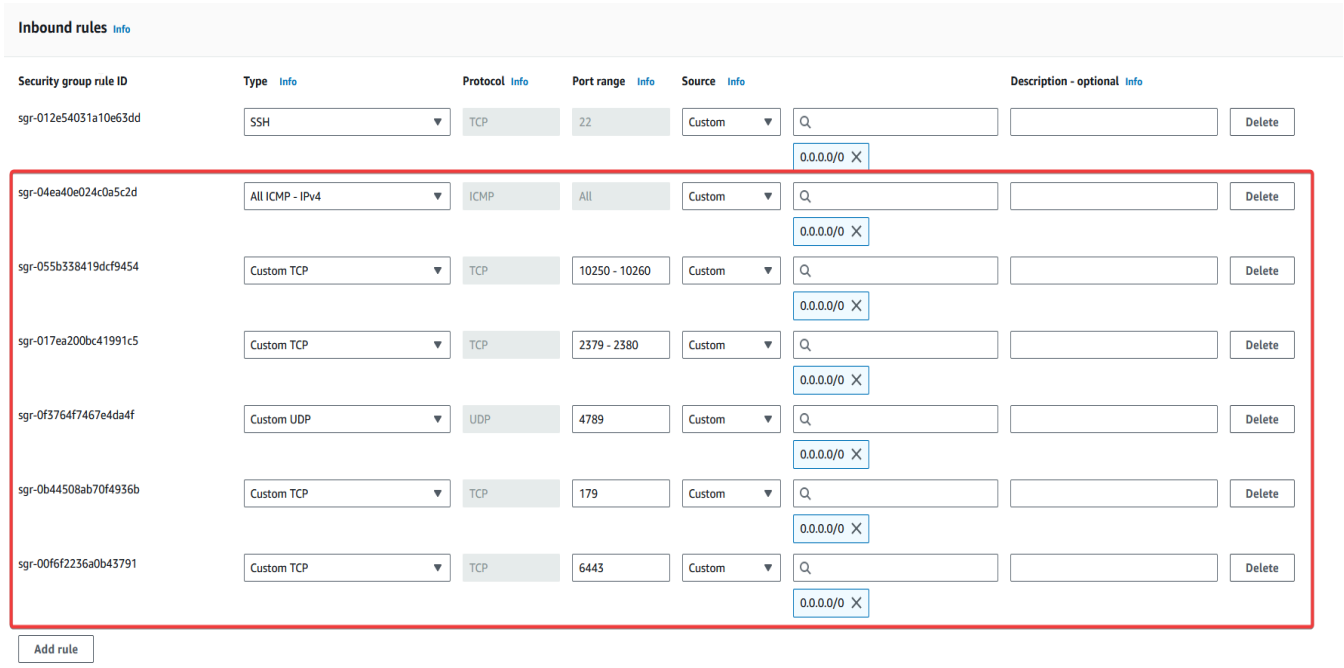
To do this, navigate to your AWS web console, select your master node, and click on "Security". Then, in the security section, click on the security group associated with your master node.



After selecting the security group, you'll need to edit the inbound rules. To do this, click on "Edit rules".



Add all of these rules



We have completed all the prerequisites for setting up the Kubernetes cluster. Now, we'll proceed with setting up the container runtime and Kubernetes packages such as kubeadm, kubelet, and kubectl.

We'll be using CRI-O as the container runtime. Here are the steps to set it up:

Install dependencies for adding the repositories

`apt-get update`

`apt-get install -y software-properties-common curl`

Add the CRI-O repository

curl -fsSL https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/Release.key | gpg --dearmor -o /etc/apt/keyrings/cri-o-apt-keyring.gpg

echo "deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg] https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/ /" | tee /etc/apt/sources.list.d/cri-o.list

Add the Kubernetes repository

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

Install the packages

apt-get update
apt-get install -y cri-o kubelet kubeadm kubectl

Now start and enable cri-o

systemctl start crio.service
systemctl status crio

Run the following command to disable automatic updates for Kubernetes packages kubelet, kubeadm, and kubectl. This prevents version skew between Kubernetes packages.

apt-mark hold kubelet kubeadm kubectl

Adding user in your master node

[root@k8master ~]# useradd kiosk
[root@k8master ~]# passwd kiosk
Changing password for user kiosk.
New password:
BAD PASSWORD:
The password is shorter than 8 characters Retype new password:
passwd: all authentication tokens updated successfully.

Giving sudo access to kiosk user

[root@k8master ~]# vim /etc/sudoers

==========

All configurations for the master are done. Now, we move towards worker node configurations. In my case, only 1 worker node is available. Repeat all of these steps for the worker node.

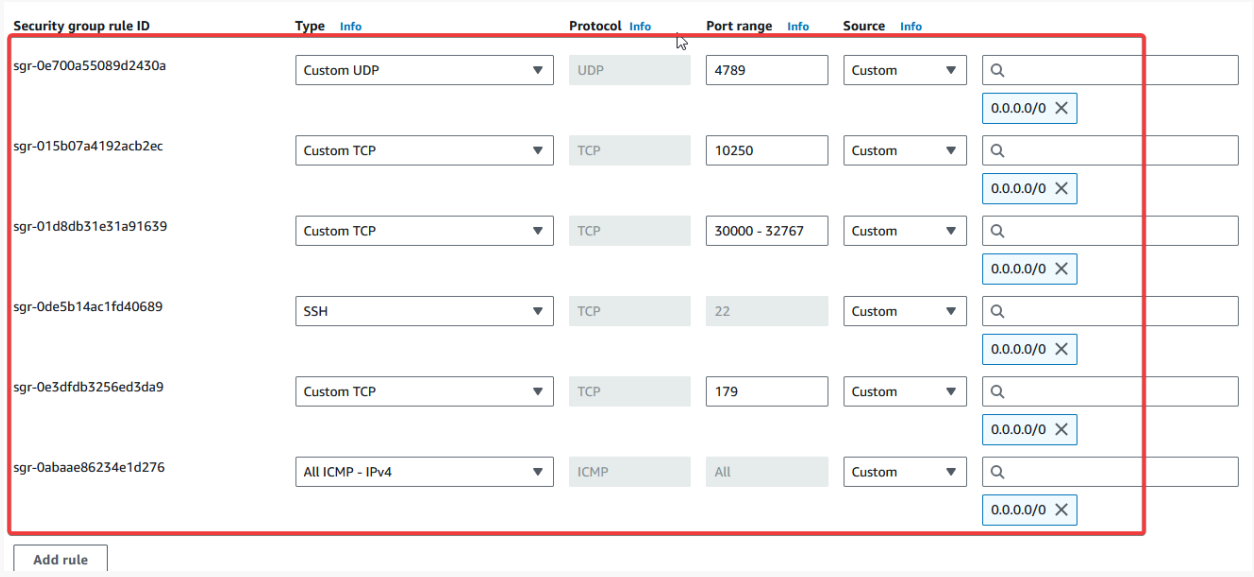
==========

NOTE:- All the steps are the same for the worker node. The only modification required is adjusting the security group rules. Additionally, you do not need to add the 'kiosk' user in the worker node.

Next, we need to add ports to the firewall for the worker nodes. Kubernetes requires the following ports:

TCP Ports: 179, 10250, 30000-32767

UDP Port: 4789



With all configurations completed for both the master and worker nodes, it's time to initialize our Kubernetes master node.

To initialize your Kubernetes master node, run the following command on your master node:

[root@k8master ~]# kubeadm init

After completing this command you will get this type of output on your terminal

```
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 192.168.43.156:6443 --token epiv7r.q6nkchryz72ar9jl \
        --discovery-token-ca-cert-hash sha256:e14c31f6d21c9495b4ddf941d9cacbb63ac7220e
2d364d8baee624af422d0713

Save this output to your notepad file this is very important of do not forget to save this
```

Before joining worker nodes run this commands on master node as kiosk user

```
[kiosk@k8master ~]$ mkdir -p $HOME/.kube

[kiosk@k8master ~]$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

[kiosk@k8master ~]$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

After this, it's time to run the `kubeadm join` command on your worker node.

Once your worker node successfully joins the master node, you can switch to the master node and run the command kubectl get nodes using a normal user account.

Now, we'll configure the CNI plugin for Kubernetes. In my case, I'm using the Calico CNI plugin. Head to your master node and follow these steps.

To set up the Calico plugin, follow these steps on your master node:

Download Calico release v3.26.3 from this link: Calico Release v3.26.3, or run the following command with the 'kiosk' user on your master node:

[kiosk@k8master ~]$ wget
https://github.com/projectcalico/calico/releases/download/v3.26.3/release-v3.26.3.tgz

After downloading, extract the file using the following command:

[kiosk@k8master ~]$ tar -xvf /home/kiosk/release-v3.26.3.tgz

Verify that the file is extracted by running:

[kiosk@k8master ~]$ ls

release-v3.26.3 release-v3.26.3.tgz

Navigate to the extracted directory:

[kiosk@k8master ~]$ cd release-v3.26.3/

Inside the directory, you'll find 'bin', 'images', and 'manifests'.

Navigate to the 'manifests' directory:

[kiosk@k8master release-v3.26.3]$ cd manifests/

Verify your location:

[kiosk@k8master manifests]$ pwd

/home/kiosk/release-v3.26.3/manifests

Finally, apply the Calico YAML manifest file on the master node using the following command. Make sure not to run this command on the worker node

[kiosk@k8master manifests]$ kubectl apply -f /home/kiosk/release-v3.26.3/manifests/calico.yaml

Run these commands to verify your cluster:

[kiosk@k8master]$ kubectl get pods -A

[kiosk@k8master]$ kubectl get nodes

NAME	STATUS	ROLES	AGE	VERSION
k8master.example.com	Ready	master	72m	v1.28.4
worker1.example.com	Ready	<none>	45m	v1.28.4

==========Cluster setup complete. Ready to deploy!==========