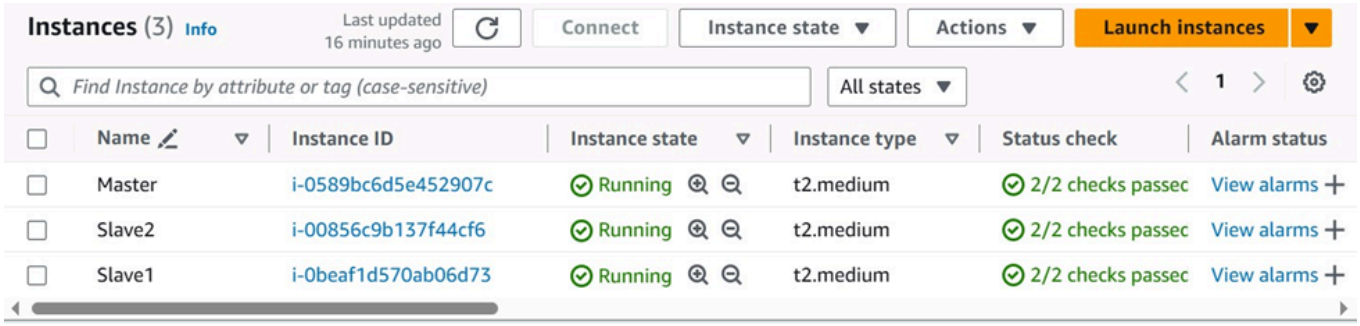
Name:Shreyash Kamat Div:D15C Roll no:22

Advanced DevOps Lab
Experiment:3

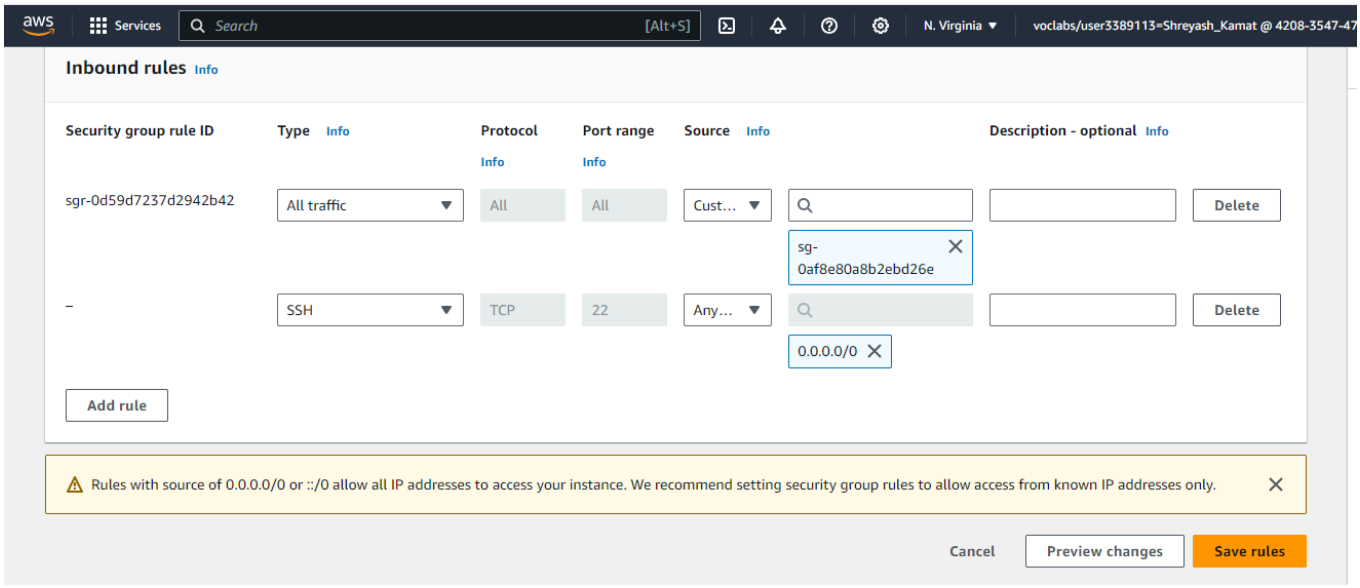
Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.

Reference: https://www.youtube.com/watch?v=Cz7hSJNq2GU

Step 1: : Create 3 EC2 instances (1 master and 2 slaves). Select SSH option in the inbound rules. Created a key pair to be used commonly between all 3 instances created. I selected AWS Linux as my operating system and enabled t2 medium option for kubernetes cluster to run smoothly



2. Edit the Security Group Inbound Rules to allow SSH



3. SSH into all 3 machines

ssh -i <keyname>.pem ubuntu@<public_ip_address>

```
quantum@machine ~/Downloads ssh -i "ec2-ubuntu.pem" ec2-user@ec2-3-88-111-183.compute-1.amazonaws.com
The authenticity of host 'ec2-3-88-111-183.compute-1.amazonaws.com (3.88.111.183)' can't be established.
ED25519 key fingerprint is SHA256:pQu+xs9foYbY3de1twjZcVVAOzmGwGv6PHmVruF/Q1s.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-88-111-183.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
```

4. From now on, until mentioned, perform these steps on all 3 machines.

Install Docker

```
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
```

```
[ec2-user@ip-172-31-22-31 ~]$ yum install docker -y
Error: This command has to be run with superuser privileges (under the root user on most systems).
[ec2-user@ip-172-31-22-31 ~]$ sudo su
[root@ip-172-31-22-31 ec2-user]# yum install docker -y
Last metadata expiration check: 0:03:24 ago on Sat Sep 14 14:54:57 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture    Version                 Repository    Size
================================================================================
Installing:
 docker                  x86_64          25.0.6-1.amzn2023.0.2   amazonlinux   44 M
Installing dependencies:
 containerd              x86_64          1.7.20-1.amzn2023.0.1   amazonlinux   35 M
 iptables-libs           x86_64          1.8.8-3.amzn2023.0.2    amazonlinux  401 k
 iptables-nft            x86_64          1.8.8-3.amzn2023.0.2    amazonlinux  183 k
 libcgroup               x86_64          3.0-1.amzn2023.0.1      amazonlinux   75 k
 libnetfilter_conntrack  x86_64          1.0.8-2.amzn2023.0.2    amazonlinux   58 k
 libnfnetlink            x86_64          1.0.1-19.amzn2023.0.2   amazonlinux   30 k
 libnftnl                x86_64          1.2.2-2.amzn2023.0.2    amazonlinux   84 k
 pigz                    x86_64          2.5-1.amzn2023.0.3      amazonlinux   83 k
 runc                    x86_64          1.1.13-1.amzn2023.0.1   amazonlinux  3.2 M

Transaction Summary
================================================================================
Install  10 Packages

Total download size: 84 M
Installed size: 317 M
```

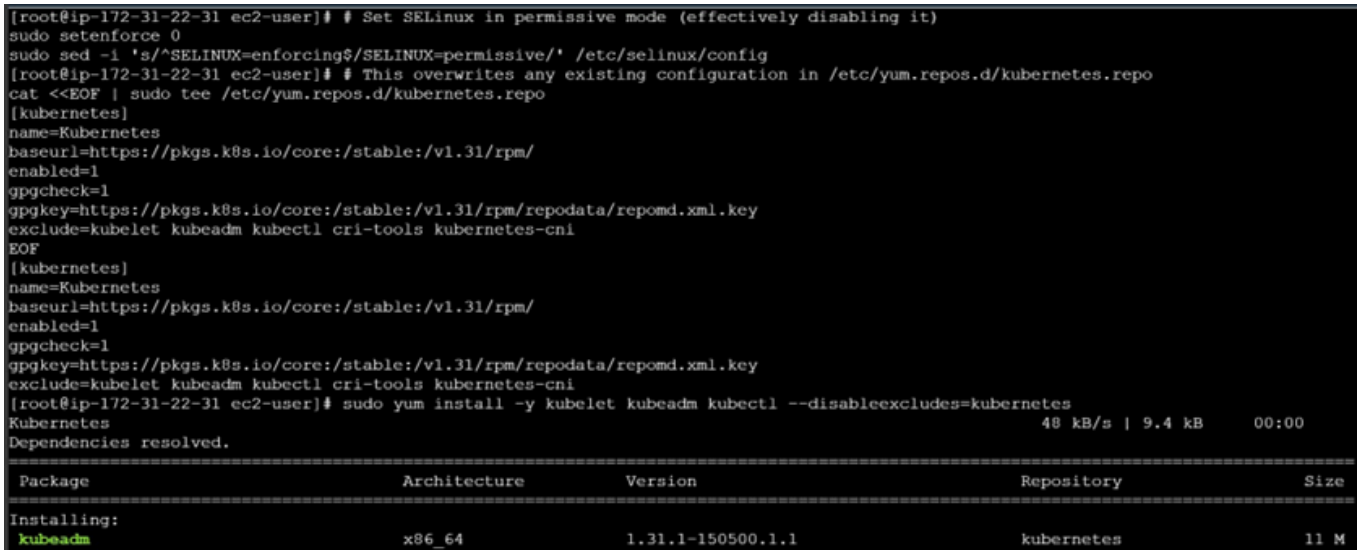
Then, configure cgroup in a daemon.json file.

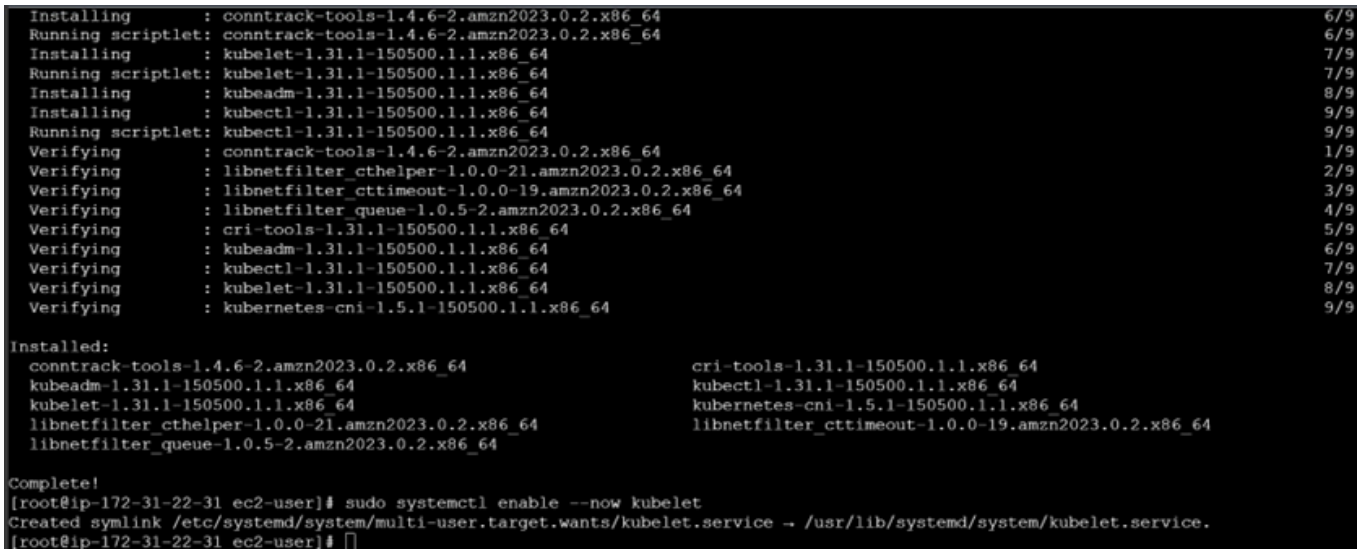
```
cd /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
```

```
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF
sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart
docker
```

Install Kubernetes on all 3 machines

```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg |
sudo apt-key add -
cat << EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb https://apt.kubernetes.io/ kubernetes-xenial main EOF
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
```



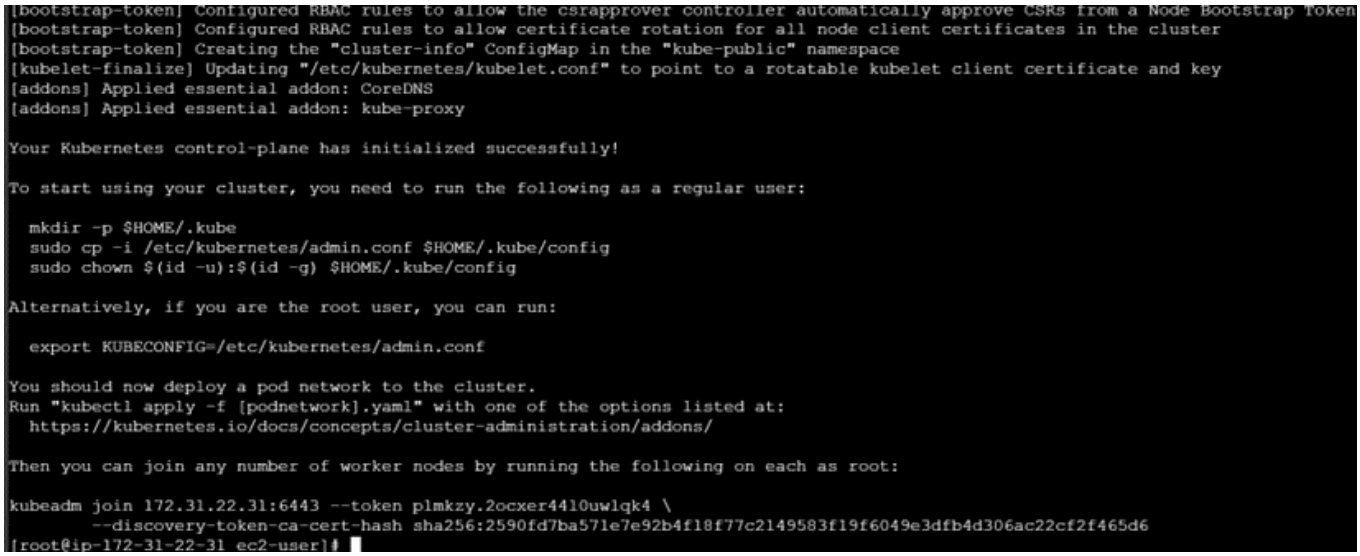


After installing Kubernetes,we need to configure internet options to allow bridging.

- Sudo swapoff-a
- echo"net.bridge.bridge-nf-call-iptables=1"|sudotee-a/etc/sysctl.conf
- Sudo sysctl-p

5. Perform this **ONLY on the Master machine**

Run command …kubeadm init with the proper network pod, here it is --pod-network-cidr= 10.244.0.0/16 to initialize kubernetes



We are supposed to add a networking plugin named flaggen with the help of the command mentioned in the console output i.e kubectl apply -f

Name:Shreyash Kamat Div:D15C Roll no:22

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

```
[ec2-user@ip-172-31-81-63 docker]$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
```

By running kubectl get nodes, we get to see the nodes that are currently connected to the master node

6. Perform this **ONLY on the worker machines**

Run the following commands to ensure a smooth and secure joining to the master node

Paste the below command on all 2 worker machines

- sudo yum install iproute-tc-y
- sudo systemctl enable kubelet
- sudo systemctl restart kubelet

Then we are supposed to run the join command that was generated in the console output of our master machine

kubeadm join 172.31.22.31:6443 --token gyakv9.hktjpt5usstl5u3y \ --discovery-token-ca-cert-hash sha256:2590fd7ba571e7e92b4f18f77c2149583f19f6049e3dfb4d306ac22cf2f465d6

```
[root@ip-172-31-23-217 ec2-user]# kubeadm join 172.31.22.31:6443 --token gyakv9.hktjpt5usstl5u3y \
        --discovery-token-ca-cert-hash sha256:2590fd7ba571e7e92b4f18f77c2149583f19f6049e3dfb4d306ac22cf2f465d6
[preflight] Running pre-flight checks
```

Post which we are supposed to get the output that our worker nodes have been successfully connected to master node.

Unfortunately, on running the join command i was not able to produce anything beyond ‘Running pre-filght checks’ which can be seen in the above image

And thus, could not execute the last step of this experiment

Conclusion: In this experiment, we set up a connection between a local machine and an EC2 instance using SSH. After facing issues like timeouts and permission problems, we learned how to check for common causes such as incorrect security group settings, improper key permissions, and network issues. By resolving these, we successfully connected to the EC2 instance. This experiment helped us understand the steps required for remote server access and troubleshooting.