Name: Shejal Nilesh Tiwari Div/Roll no: D15C/57

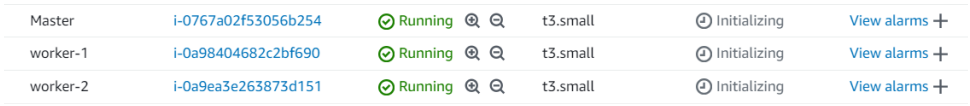
EXPERIMENT NO: 3

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

STEPS :

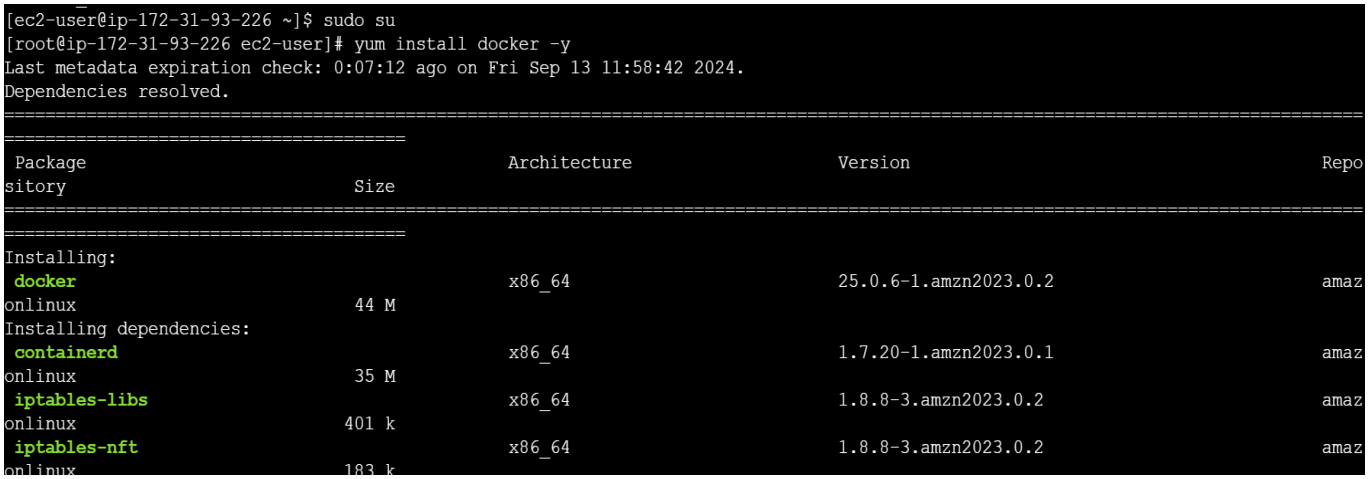
1.**Create Instances:**

We initiated the creation of three virtual machines or instances, naming them Master, Worker1, and Worker2. These instances will act as the nodes in our Kubernetes cluster.



Master	i-0767a02f53056b254	Running	t3.small	Initializing	View alarms +
worker-1	i-0a98404682c2bf690	Running	t3.small	Initializing	View alarms +
worker-2	i-0a9ea3e263873d151	Running	t3.small	Initializing	View alarms +

2.**Install Docker**:
On each instance, execute the following commands to gain superuser privileges and install Docker:
sudo su
yum install docker -y
This step ensures that Docker, a container runtime, is installed on Master, Worker1, and Worker2. Docker is essential for running the containerized applications that Kubernetes will orchestrate.



```
[ec2-user@ip-172-31-93-226 ~]$ sudo su
[root@ip-172-31-93-226 ec2-user]# yum install docker -y
Last metadata expiration check: 0:07:12 ago on Fri Sep 13 11:58:42 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
```

3.Start Docker:

Start the Docker service on each instance to ensure it is running:

```
systemctl start docker
```

- This command activates the Docker service on Master, Worker1, and Worker2, enabling them to run and manage containers.

```
[root@ip-172-31-93-226 ec2-user]# systemctl start docker
[root@ip-172-31-93-226 ec2-user]#
```

4.**Install kubeadm:**
Prepare the system for Kubernetes installation by configuring SELinux and adding the Kubernetes repository:

```
sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
```

Install Kubernetes components:

```
sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
sudo systemctl enable --now kubelet
```

These steps configure SELinux to run in permissive mode, add the Kubernetes repository, and install Kubernetes tools (kubelet, kubeadm, and kubectl) on Master, Worker1, and Worker2. The kubelet service is also enabled and started to ensure it runs on boot.

```
Installed:
  conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64              cri-tools-1.31.1-150500.1.1.x86_64                      kubeadm-1.31.1-15
0500.1.1.x86_64
  kubectl-1.31.1-150500.1.1.x86_64                         kubelet-1.31.1-150500.1.1.x86_64                        kubernetes-cni-1.
5.1-150500.1.1.x86_64
  libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64       libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64     libnetfilter_queu
e-1.0.5-2.amzn2023.0.2.x86_64

Complete!
[root@ip-172-31-84-46 ec2-user]# sudo systemctl enable --now kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /usr/lib/systemd/system/kubelet.service.
```

5.Confirm Repository:

Verify that the Kubernetes repository is correctly configured by listing all enabled repositories:
bash
Copy code
yum repo list
Execute this command on Master, Worker1, and Worker2 to confirm that the Kubernetes repository has been added and is available for package installations.

```
[root@ip-172-31-84-46 ec2-user]# yum repolist
repo id                                                    repo name
amazonlinux                                                Amazon Linux 2023 repository
kernel-livepatch                                           Amazon Linux 2023 Kernel Livepatch repository
kubernetes                                                 Kubernetes
[root@ip-172-31-84-46 ec2-user]#
```

6.Initialize kubeadm on Master Node:

On the Master node, initialize the Kubernetes cluster using kubeadm. This process sets up the Kubernetes control plane and generates commands for joining worker nodes:
Copy the commands displayed in the output of the initialization process to configure permissions and obtain the join token. This includes a join command link needed for worker nodes to connect to the master.

```
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 172.31.84.46:6443 --token jl06ac.t7cdzxf0x5eddmsl \
        --discovery-token-ca-cert-hash sha256:4a152b913ee4b60dc2126d55f631b86d0dafb7d58132416c4f32f0668ac553be
[root@ip-172-31-84-46 ec2-user]#
```

In the screenshot you can see the commands written in the 3rd 4th and 5th line
Copy that command , this command is used to add right permission to the user
Also copy the 7th line , here it is the credential for the user .
Also copy the last 2 lines it is a link used to join the nodes

7.Join Worker Nodes:

On Worker1 and Worker2, use the join command provided by the Master node to connect them to the Kubernetes cluster:

kubeadm join 172.31.84.46:6443 --token jl06ac.t7cdzxf0x5eddmsl \
    --discovery-token-ca-cert-hash
sha256:4a152b913ee4b60dc2126d55f631b86d0dafb7d58132416c4f32f0668ac553be
This command allows Worker1 and Worker2 to register themselves with the Master node and become part of the Kubernetes cluster.
Verify Nodes:

Name: Shejal Nilesh Tiwari Div/Roll no: D15C/57

On the Master node, check the status of all nodes in the cluster to ensure that Worker1 and Worker2 have successfully joined:

kubectl get nodes

This command lists all nodes and their status within the Kubernetes cluster, helping to confirm successful node registration and connectivity.

```
[root@ip-172-31-84-46 ec2-user]# kubectl get node
NAME                           STATUS     ROLES           AGE   VERSION
ip-172-31-84-46.ec2.internal   NotReady   control-plane   56s   v1.31.1
[root@ip-172-31-84-46 ec2-user]# kubectl get node
The connection to the server 172.31.84.46:6443 was refused - did you specify the right host or port?
[root@ip-172-31-84-46 ec2-user]# kubectl get node
NAME                           STATUS     ROLES           AGE    VERSION
ip-172-31-84-46.ec2.internal   NotReady   control-plane   5m1s   v1.31.1
[root@ip-172-31-84-46 ec2-user]#
```

And this was the output when i tried connecting it with worker 1 and worker 2

```
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Fri Sep 13 12:04:33 2024 from 18.206.107.28
[ec2-user@ip-172-31-87-149 ~]$ sudo su
[root@ip-172-31-87-149 ec2-user]# kubeadm join 172.31.84.46:6443 --token qq448c.8c0zquwywz3eqt0b        --discovery-token-ca-cert-has
h sha256:49cc770e646aab704883a3d9fb30e6146fdd83e782ebd5ec3194dacee59f2257
[preflight] Running pre-flight checks
        [WARNING FileExisting-socat]: socat not found in system path
        [WARNING FileExisting-tc]: tc not found in system path
```

So there was a failure in connecting the worker 1 and 2 with the master even if i provided the joining link.
The Cloud shell didn't proceeds further.

CONCLUSION :
We created and configured three instances (Master, Worker1, and Worker2), installed Docker and Kubernetes on all nodes, and initialized the Kubernetes control plane on the Master node. Despite providing the join command to Worker1 and Worker2, they failed to connect to the Master node.

Connection Refused Errors: The Kubernetes components are failing to connect to the API server at `https://172.31.84.46:6443`, resulting in "connection refused" errors.
No Running Containers: The `docker ps -a` command shows no containers, which means the Kubernetes components are not running in Docker.

CrashLoopBackOff: There are errors indicating that the containers for Kubernetes components are restarting repeatedly but failing to start properly.
Network Plugin Not Ready: The error message "Network plugin returns error: cni plugin not initialized" suggests issues with the network plugin required for Kubernetes.