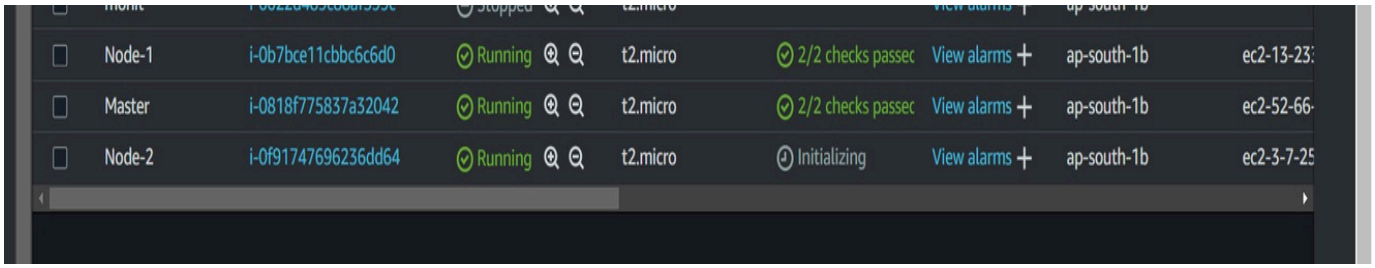
Experiment:3

Name: Shivpratik Hande **Class:D15c Roll no.14**

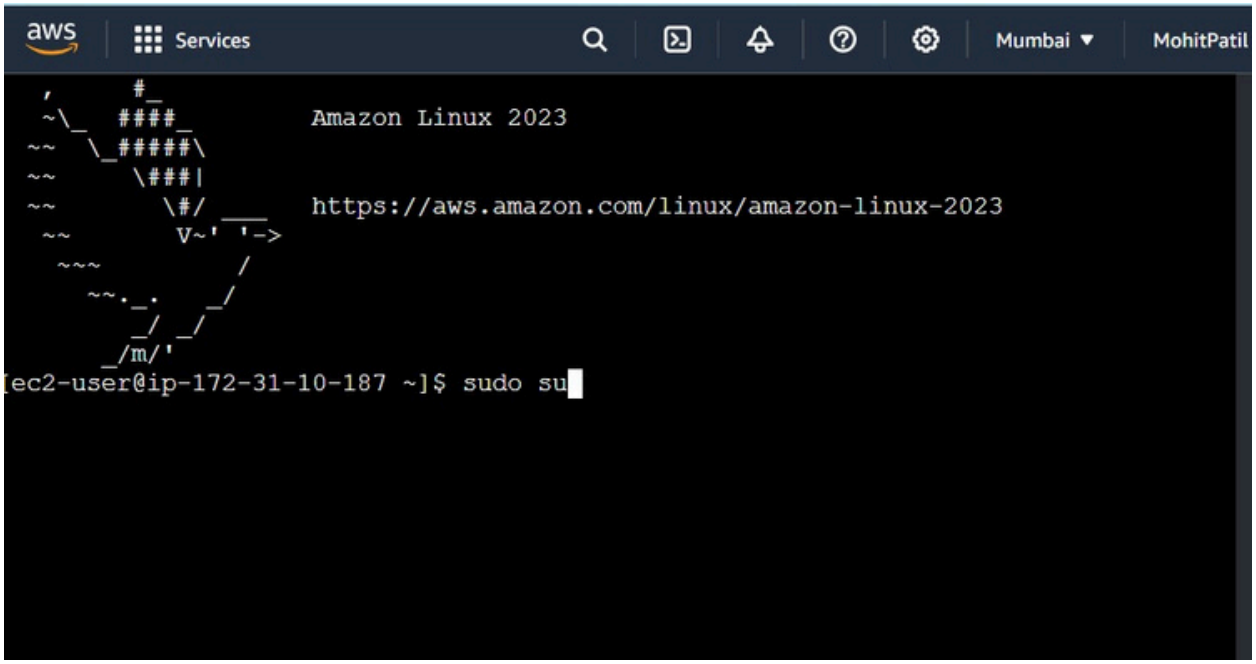
Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.

1. Create 3 EC2 Ubuntu Instances on AWS.



1. Now click on connect to instance, then click on SSH client.

2. Now copy the ssh from the example and paste it on command prompt.(I used gitbash)



3. After this type on all 3 machines Yum install docker -y

```
    ~~~        /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-10-245 ~]$ sudo su
[root@ip-172-31-10-245 ec2-user]# yum install docker -y
```

4. To start the docker on master and slave perform this command: Systemctl start docker

```
  containerd-1.7.20-1.amzn2023.0.1.x86_64              docker-25.0.6-1.amzn2023.0.2
.x86_64            iptables-libs-1.8.8-3.amzn2023.0.2.x86_64
  iptables-nft-1.8.8-3.amzn2023.0.2.x86_64             libcgroup-3.0-1.amzn2023.0.1
.x86_64            libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64
  libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64            libnftnl-1.2.2-2.amzn2023.0.
2.x86_64           pigz-2.5-1.amzn2023.0.3.x86_64
  runc-1.1.13-1.amzn2023.0.1.x86_64

Complete!
[root@ip-172-31-10-245 ec2-user]#
```

i-0b7bce11cbbc6c6d0 (Node-1)

PublicIPs: 13.233.152.101 PrivateIPs: 172.31.10.245

```
Complete!
[root@ip-172-31-10-187 ec2-user]#  systemctl start docker
```

i-0818f775837a32042 (Master)

PublicIPs: 52.66.241.212 PrivateIPs: 172.31.10.187

EXTRA:

To check if docker is installed or not

Docker -v

```
[root@ip-172-31-84-37 ec2-user]# systemctl start docker
[root@ip-172-31-84-37 ec2-user]# sudo su
[root@ip-172-31-84-37 ec2-user]# yum repolist
repo id                                                    repo name
amazonlinux                                                Amazon Linux 2023 repository
kernel-livepatch                                           Amazon Linux 2023 Kernel Livepatch repository
[root@ip-172-31-84-37 ec2-user]# docker --version
Docker version 25.0.5, build 5dc9bcc
```

5. Now to install kubeadm on master and Nodes :

Installing kubeadm:Go the official documentation

off kubeadm.

```
[ec2-user@ip-172-31-14-163 ~]$ sudo service docker start
Redirecting to /bin/systemctl start docker.service
[ec2-user@ip-172-31-14-163 ~]$ sudo systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
[ec2-user@ip-172-31-14-163 ~]$ sudo systemctl start docker
[ec2-user@ip-172-31-14-163 ~]$
```

/ Installing kubeadm

Installing kubeadm

This page shows how to install the `kubeadm` toolbox. For information on how to create a cluster with kubeadm once you have performed this installation process, see the Creating a cluster with kubeadm page.

This installation guide is for Kubernetes v1.31. If you want to use a different Kubernetes version, please refer to the following pages instead:

- Installing kubeadm (Kubernetes v1.30)
- Installing kubeadm (Kubernetes v1.29)
- Installing kubeadm (Kubernetes v1.28)
- Installing kubeadm (Kubernetes v1.27)

Before you begin

- A compatible Linux host. The Kubernetes project provides generic instructions for Linux distributions based on Debian and Red Hat, and

Edi
Cre
Cre
Prir

Before y
Verify th
unique
Check n
Check r
Swap co
Installin
Installin
Configu
Trouble
What's r

6. Scroll down and select Red Hat based distributions:

Note:
There's a dedicated package repository for each Kubernetes minor version. If you want to install a minor version other than v1.31, please see the installation guide for your desired minor version.

Debian-based distributions | Red Hat-based distributions

Without a package manager

1. Set SELinux to `permissive` mode:

   These instructions are for Kubernetes 1.31.

```
# Set SELinux in permissive mode (effectively disabling it)
sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
```

Caution:

- Setting SELinux in permissive mode by running `setenforce 0` and `sed ...` effectively disables it. This is required to allow containers to access the host filesystem; for example, some cluster network plugins require that. You have to do this until SELinux support is improved in the kubelet.
- You can leave SELinux enabled if you know how to configure it but it may require

Creat
Creat
Print

Before yo
Verify the
unique fo
Check net
Check req
Swap con
Installing
Installing
Configurir
Troublesh
What's ne

7.Now copy the command on all 3 machines:

1. Set SELinux to `permissive` mode:

   These instructions are for Kubernetes 1.31.

```
# Set SELinux in permissive mode (effectively disabling it)
sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
```

```
# This overwrites any existing configuration in /etc/yum.repos.d/kubernetes.repo
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
```

3. Install kubelet, kubeadm and kubectl:

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

8.Run yum repolist command for checking the repositorires

```
        _/ _/
      _/m/'
Last login: Wed Sep 18 14:58:05 2024 from 13.233.177.3
[ec2-user@ip-172-31-10-187 ~]$ sudo su
[root@ip-172-31-10-187 ec2-user]# yum repolist
repo id                                                          repo name
amazonlinux                                                      Amazon Linux
 2023 repository
kernel-livepatch                                                 Amazon Linux
 2023 Kernel Livepatch repository
[root@ip-172-31-10-187 ec2-user]#
```

```
8/9
  Verifying         : kubernetes-cni-1.5.1-150500.1.1.x86_64

9/9

Installed:
  conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64            cri-tools-1.31.1-150500
.1.1.x86_64                        kubeadm-1.31.1-150500.1.1.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_queue-1.0.5-2.amzn2023.0.2.x86_64


Complete!
[root@ip-172-31-10-187 ec2-user]# sudo systemctl enable --now kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /
usr/lib/systemd/system/kubelet.service.
[root@ip-172-31-10-187 ec2-user]#
```

i-0818f775837a32042 (Master)

PublicIPs: 52.66.241.212 PrivateIPs: 172.31.10.187

```
Complete!
[root@ip-172-31-10-187 ec2-user]# sudo systemctl enable --now kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /usr/lib/systemd/system/kubelet.service.
[root@ip-172-31-10-187 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-172-31-10-187 ec2-user]#
```

i-0818f775837a32042 (Master)

PublicIPs: 52.66.241.212 PrivateIPs: 172.31.10.187

EXTRA :

Got error in initializing Kubernetes

```
[root@ip-172-31-31-240 ec2-user]# kubeadm init
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
W0908 11:25:45.820964    2320 checks.go:1080] [preflight] WARNING: Couldn't create the interface used for talking t
RI runtime service: validate service connection: validate CRI v1 runtime API for endpoint "unix:///var/run/containe
ble desc = connection error: desc = "transport: Error while dialing: dial unix /var/run/containerd/containerd.sock:
        [WARNING FileExisting-tc]: tc not found in system path
error execution phase preflight: [preflight] Some fatal errors occurred:
        [ERROR FileContent--proc-sys-net-ipv4-ip_forward]: /proc/sys/net/ipv4/ip_forward contents are not set to 1
[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`
To see the stack trace of this error execute with --v=5 or higher
```

9.Copy paste the commands in all three instances

```
[root@ip-172-31-10-187 ec2-user]# kubeadm init
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
        [WARNING FileExisting-socat]: socat not found in system path
        [WARNING FileExisting-tc]: tc not found in system path
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action beforehand using 'kubeadm config images pull'
W0918 15:44:45.531044    3429 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the container runtime is inconsistent with that used b
y kubeadm.It is recommended to use "registry.k8s.io/pause:3.10" as the CRI sandbox image.
```

```
  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 172.31.10.187:6443 --token v5qt9c.8kc71gb6xwil18co \
        --discovery-token-ca-cert-hash sha256:b9765a8ba4dc546e8263bd7b7531f78b4b8582c9757a9371e00ab90a71a38785
[root@ip-172-31-10-187 ec2-user]#   mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
[root@ip-172-31-10-187 ec2-user]#   export KUBECONFIG=/etc/kubernetes/admin.conf
[root@ip-172-31-10-187 ec2-user]#
```

i-0818f775837a32042 (Master)

PublicIPs: 13.126.147.65 PrivateIPs: 172.31.10.187

10.After pasting the connection link in the nodes run the kublect get nodes command to view the connected nodes successfully

```
ubuntu@ip-172-31-17-23:~$ kubectl get nodes
NAME                STATUS   ROLES           AGE     VERSION
ip-172-31-17-23     Ready    control-plane   3m56s   v1.29.0
ip-172-31-18-12     Ready    <none>          37s     v1.29.0
ip-172-31-26-153    Ready    <none>          24s     v1.29.0
ubuntu@ip-172-31-17-23:~$ kubectl get nodes
NAME                STATUS   ROLES           AGE     VERSION
ip-172-31-17-23     Ready    control-plane   9m34s   v1.29.0
ip-172-31-18-12     Ready    <none>          6m15s   v1.29.0
ip-172-31-26-153    Ready    <none>          6m2s    v1.29.0
ubuntu@ip-172-31-17-23:~$
```

Conclusion:

In this setup, we established a Kubernetes cluster using three AWS EC2 instances, successfully deploying Docker and Kubernetes components on each. While the master node is operational and the initial configuration is complete, the worker nodes are encountering challenges when attempting to join the cluster. These issues appear to stem from configuration or networking problems. To finalize the cluster setup and ensure its proper functionality, further troubleshooting on the worker nodes is necessary to resolve these connectivity issues. Once these challenges are addressed, the cluster will be fully operational, allowing for efficient management and scaling of containerized applications across the instances.