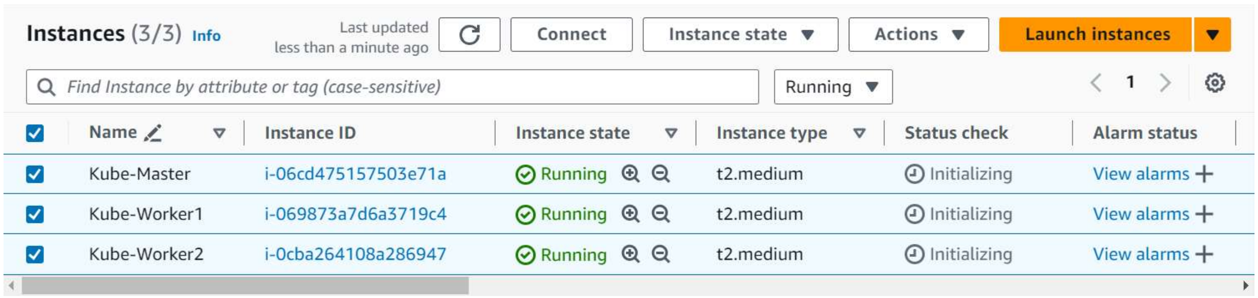
Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machine

1. Create 3 ec2 instances with an OS as Amazon Linux. Select the instance type as t2.medium.



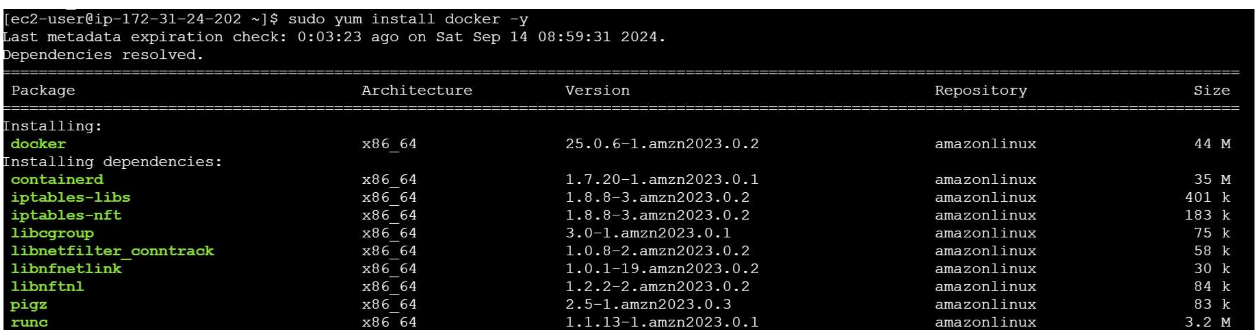
Instances (3/3) Info — Last updated less than a minute ago — Connect — Instance state — Actions — Launch instances

Find Instance by attribute or tag (case-sensitive) — Running

Name	Instance ID	Instance state	Instance type	Status check	Alarm status
Kube-Master	i-06cd475157503e71a	Running	t2.medium	Initializing	View alarms +
Kube-Worker1	i-069873a7d6a3719c4	Running	t2.medium	Initializing	View alarms +
Kube-Worker2	i-0cba264108a286947	Running	t2.medium	Initializing	View alarms +

From now onwards perform the steps on all three instance unless mentioned otherwise.

2. Install docker with the command:
   sudo yum install docker -y



```
[ec2-user@ip-172-31-24-202 ~]$ sudo yum install docker -y
Last metadata expiration check: 0:03:23 ago on Sat Sep 14 08:59:31 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
```

3. Configure cgroup in deamon.json file using the following commands.

- cd /etc/docker
- 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

```
[ec2-user@ip-172-31-24-202 ~]$ cd /etc/docker
[ec2-user@ip-172-31-24-202 docker]$ 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
{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
[ec2-user@ip-172-31-24-202 docker]$
```

4. Enable docker
    - sudo systemctl enable docker
    - sudo systemctl daemon-reload
    - sudo systemctl restart docker

```
[ec2-user@ip-172-31-24-202 docker]$ sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
[ec2-user@ip-172-31-24-202 docker]$ docker -v
Docker version 25.0.5, build 5dc9bcc
[ec2-user@ip-172-31-24-202 docker]$
```

5. Install Kubernetes
    I. Disable SELinux before configuring kubelet
    sudo setenforce 0
    sudo sed -i ‘s/^SELINUX=enforcing$/SELINUX=permissive/’ /etc/selinux/config

```
[ec2-user@ip-172-31-24-202 docker]$ sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[ec2-user@ip-172-31-24-202 docker]$
```

    II. Add kubernetes repository
    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

```
[ec2-user@ip-172-31-24-202 docker]$ 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
[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
[ec2-user@ip-172-31-24-202 docker]$
```

III. Run the commands to install kubernetes packages
sudo yum update
sudo yum install -y kubelet kubeadm kubectl
--disableexcludes=kubernetes

```
  Installing       : libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64                                    4/9
  Installing       : libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64                                     5/9
  Installing       : conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64                                            6/9
  Running scriptlet: conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64                                            6/9
  Installing       : kubelet-1.31.1-150500.1.1.x86_64                                                       7/9
  Running scriptlet: kubelet-1.31.1-150500.1.1.x86_64                                                       7/9
  Installing       : kubeadm-1.31.1-150500.1.1.x86_64                                                       8/9
  Installing       : kubectl-1.31.1-150500.1.1.x86_64                                                       9/9
  Running scriptlet: kubectl-1.31.1-150500.1.1.x86_64                                                       9/9
  Verifying        : conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64                                            1/9
  Verifying        : libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64                                     2/9
  Verifying        : libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64                                    3/9
  Verifying        : libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64                                         4/9
  Verifying        : cri-tools-1.31.1-150500.1.1.x86_64                                                     5/9
  Verifying        : kubeadm-1.31.1-150500.1.1.x86_64                                                       6/9
  Verifying        : kubectl-1.31.1-150500.1.1.x86_64                                                       7/9
  Verifying        : kubelet-1.31.1-150500.1.1.x86_64                                                       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!
[ec2-user@ip-172-31-24-202 docker]$
```

IV. Configure internet options to allow bridging
- sudo swapoff -a

- echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
- sudo sysctl -p

```
[ec2-user@ip-172-31-24-202 docker]$ sudo swapoff -a
echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p
net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-iptables = 1
[ec2-user@ip-172-31-24-202 docker]$
```

6. Perform the following only on master machine

   sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=all

```
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 172.31.24.202:6443 --token gxk2tb.nyjtgrmr2smoni5h \
        --discovery-token-ca-cert-hash sha256:1344a4ef2f818653de2aa22de8906da096356edf67eadc0c3fbd7fd92d4c04a6
```

   Save the join command in notepad
   Copy the commands given and run it.

```
[ec2-user@ip-172-31-24-202 ~]$ mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
[ec2-user@ip-172-31-24-202 ~]$
```

   Add the networking plugin (Flammel file) using the following command.

kubectl apply -f
https://github.com/flannel-io/flannel/releases/latest/download/kube-flannel.yml

```
[ec2-user@ip-172-31-24-202 ~]$ kubectl apply -f https://github.com/flannel-io/flannel/releases/latest/download/kube-flannel.yml
namespace/kube-flannel created
serviceaccount/flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
[ec2-user@ip-172-31-24-202 ~]$
```

7. Perform the following commands in worker node only.

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

```
[ec2-user@ip-172-31-30-225 ~]$ sudo yum install iproute-tc-y
sudo systemctl enable kubelet
sudo systemctl restart kubelet
Last metadata expiration check: 0:47:11 ago on Sat Sep 14 09:29:30 2024.
No match for argument: iproute-tc-y
Error: Unable to find a match: iproute-tc-y
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /usr/lib/systemd/system/kubelet.service.
```

Run the join command saved in notepad previously.

```
[ec2-user@ip-172-31-30-225 ~]$ sudo kubeadm join 172.31.24.202:6443 --token clj8yw.nk67cs21cfmgn8g4 \
        --discovery-token-ca-cert-hash sha256:1344a4ef2f818653de2aa22de8906da096356edf67eadc0c3fbd7fd92d4c04a6
[preflight] Running pre-flight checks
        [WARNING FileExisting-socat]: socat not found in system path
        [WARNING FileExisting-tc]: tc not found in system path
```

The following output indicates that the command was successfully run, but its execution is not getting completed. The node faces a delay in joining the cluster. This could be due to reasons like Network Connectivity issue or some misconfiguration in the cluster configuration.

Conclusion:

The experiment involved setting up a Kubernetes cluster using three Amazon Linux EC2 instances, and installing Kubernetes components on each instance. The master node was initialized with kubeadm, and a Flannel network plugin was applied for pod networking. The worker nodes were configured to join the cluster using a join command generated during the initialization. We successfully understood the process of setting up a Kubernetes Cluster on EC2 instance, however, a delay in worker nodes joining the master node was observed, which could be attributed to possible network connectivity issues or configuration errors