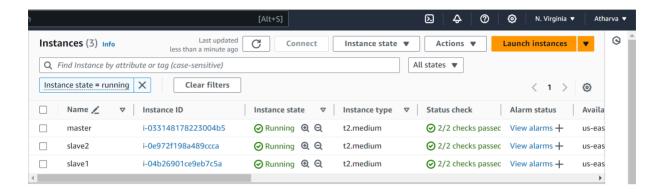
ATHARVA PRABHU D15A 43

EXPERIMENT NO. 3

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

1. Create 3 EC-2 instances with all running on Amazon Linux as OS with inbound SSH allowed. To efficient run kubernetes cluster select instance type of at least t2.medium as kubernetes recommends at least 2 vCPU to run smoothly

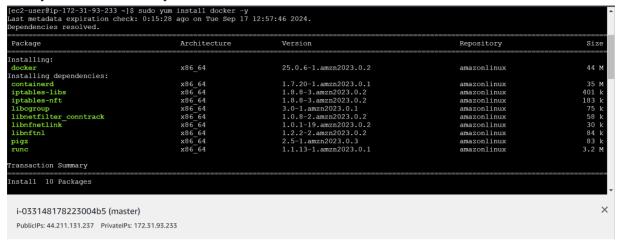


2. SSH into all 3 machines each in separate terminal

I

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

sudo yum install docker -y



Then, configure cgroup in a daemon.json file by using following commands

• 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"
    }
}</li>
```

EOF

- sudo systemctl enable docker
- sudo systemctl daemon-reload
- sudo systemctl restart docker
- docker -v

```
, [ec2-user@ip-172-31-93-233 docker]$ sudo systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
[ec2-user@ip-172-31-93-233 docker]$ sudo systemctl daemon-reload
[ec2-user@ip-172-31-93-233 docker]$ sudo systemctl restart docker
[ec2-user@ip-172-31-93-233 docker]$ docker -v
Docker version 25.0.5, build 5dc9bcc
```

- 4. Install Kubernetes on all 3 machines SELinux needs to be disable before configuring kubelet
- sudo setenforce 0
- sudo sed -i 's/^SELINUX=enforcing\$/SELINUX=permissive/' /etc/selinux/config

```
[ec2-user@ip-172-31-94-214 docker]$ sudo setenforce 0
[ec2-user@ip-172-31-94-214 docker]$ sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[ec2-user@ip-172-31-94-214 docker]$
```

Add kubernetes repository (paste in terminal)

```
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo [kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/r
epomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
```

Type following commands:

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

```
[ec2-user@ip-172-31-93-233 ~]$ sudo yum update
Kubernetes
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-93-233 ~]$ sudo yum install -y kubelet kubeadm kubectl --disableexcludes-kubernetes
Last metadata expiration check: 0:00:16 ago on Wed Sep 18 09:30:01 2024.
Dependencies resolved.

Package Architecture Version Repository Size

Installing:

Kubeadm x86 64 1.30.5-150500.1.1 kubernetes 10 M
kubectl x86 64 1.30.5-150500.1.1 kubernetes 10 M
kubelet x86 64 1.30.5-150500.1.1 kubernetes 10 M
kubenetes x86 64 1.30.5-150500.1.1 kubernetes 10 M
kubernetes 17 M
Installing dependencies:

contrack-tools x86 64 1.4.6-2.amzn2023.0.2 amazonlinux 208 k
kubernetes-cni x86 64 1.4.0-150500.1.1 kubernetes 8.6 M
kubernetes-cni x86 64 1.4.0-150500.1.1 kubernetes 6.7 M
kubernetes-cni x86 64 1.4.0-150500.1.1 kubernetes 8.6 M
kubernetes-cni x86 64 1.4.0-19.amzn2023.0.2 amazonlinux 24 k
libnetfilter_cthelper x86 64 1.0.0-21.amzn2023.0.2 amazonlinux 24 k
libnetfilter_queue x86 64 1.0.5-2.amzn2023.0.2 amazonlinux 30 k
```

After installing Kubernetes, we need to 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
- 5. Perform this ONLY on the Master machine

Initialize kubernetes by typing below command

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

```
[addons] Applied essential addon: kube-proxy
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) $HOME/.kube/config
Alternatively, if you are the root user, you can run:
  export KUBECONFIG=/etc/kubernetes/admin.conf
You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
 https://kubernetes.io/docs/concepts/cluster-administration/addons/
Then you can join any number of worker nodes by running the following on each as root:
kubeadm join 172.31.81.63:6443 --token zh5jbb.a6ty3eujzc51d15d \
        --discovery-token-ca-cert-hash sha256:0822f656bf52a17a2b6686c123f811306f41495ca650a0aed9bf6cd2d2f6f8c5
[ec2-user@ip-172-31-81-63 docker]$ mkdir -p $HOME/.kube
 sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
 sudo chown $(id -u):$(id -g) $HOME/.kube/config
[ec2-user@ip-172-31-81-63 docker]$ |
```

Copy the mkdir and chown commands from the top and execute them

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -q) $HOME/.kube/config
```

Copy this join link and save it in clipboard (copy from your output as it different for each instance)

```
kubeadm join 172.31.81.63:6443 --token zh5jbb.a6ty3eujzc51d15d \
--discovery-token-ca-cert-hash sha256:0822f656bf52a17a2b6686c123f811306f41495ca650a0aed9bf6c d2d2f6f8c5
```

Then, add a common networking plugin called flammel file as mentioned in the code.

kubectl apply -f

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
```

Check the created pod using this command

- kubectl get pods
- 6. Perform this ONLY on the worker machines

Paste the below command on all 2 worker machines

- Sudo yum install iproute-tc -y
- sudo systemctl enable kubelet
- sudo systemctl restart kubelet
- kubeadm join 172.31.81.63:6443 --token zh5jbb.a6ty3eujzc51d15d \
- --discovery-token-ca-cert-hash

sha256:0822f656bf52a17a2b6686c123f811306f41495ca650a0aed9bf6cd2d2f6f8 C5

Now we can see in the master/control node of kubernetes that worker nodes are connected by typing watch kubectl get nodes in the master node instance

```
Every 2.0s: kubectl get nodes
                                STATUS
NAME
                                         ROLES
                                                          AGE
                                                                  VERSION
ip-172-31-81-63.ec2.internal
                                Ready
                                          control-plane
                                                          29m
                                                                  v1.30.4
ip-172-31-87-137.ec2.internal
                                                          5m58s
                                                                  v1.30.4
                                Ready
                                          <none>
ip-172-31-92-18.ec2.internal
                                                          5m53s
                                                                  v1.30.4
                                Ready
                                          <none>
```