

Advanced DevOps Lab

Experiment:3

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

Steps:

- ## 1. Create 3 EC2 Ubuntu Instances on AWS.

Instances (1/3) Info

Last updated less than a minute ago

Refresh

Connect

Instance state ▾

Actions ▾



















Launch instances ▾

ⓘ

Find Instance by attribute or tag (case-sensitive)

All states ▾

< 1 > ⓘ

<input type="checkbox"/>	Name 	Instance ID	Instance state 	Instance type 	Status check 	Alarm status	Availability Zone 	Public IPv4 DNS 	Public IPv4 ... 	Elastic IP	IF
<input type="checkbox"/>	worker-2	i-0565519f0e4f7c049	 Running 	t2.micro	 2/2 checks passed	View alarms 	us-east-1a	ec2-44-203-151-154.co...	44.203.151.154	-	-
<input checked="" type="checkbox"/>	Master	i-06d8202bcfb192d3b	 Terminated 	t2.micro	-	View alarms 	us-east-1a	-	-	-	-
<input type="checkbox"/>	worker-1	i-0da69d1cb88393d8b	 Running 	t2.micro	 2/2 checks passed	View alarms 	us-east-1a	ec2-54-164-90-206.co...	54.164.90.206	-	-

(Name 1 as Master, the other 2 as worker-1 and worker-2)

- ## -2. Edit the Security Group Inbound Rules to allow SSH



- ### 3. SSH into all 3 machines `ssh -i <keyname>.pem`

ubuntu@<public ip address>

```
The authenticity of host 'ec2-44-203-151-154.compute-1.amazonaws.com (44.203.151.154)' can't be established.  
ED25519 key fingerprint is SHA256:4cidwEvWyyoqWE0gGmsDmJx2S1xkZVTU1bDzmdDlc.  
This key is not known by any other names.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added 'ec2-44-203-151-154.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
```

[illegible]

```
[ec2-user@ip-172-31-95-91 ~]$ ssh -i "newkey.pem" ec2-user@ec2-54-164-90-206.compute-1.amazonaws.com
Warning: Identity file newkey.pem not accessible: No such file or directory.
The authenticity of host 'ec2-54-164-90-206.compute-1.amazonaws.com (172.31.88.50)' can't be established.
ED25519 key fingerprint is SHA256:RRoS21NvNq9JLCAJhDKUn6fIRCRu1+VtNbkjyV05M/.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-164-90-206.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
ec2-user@ec2-54-164-90-206.compute-1.amazonaws.com: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[ec2-user@ip-172-31-95-91 ~]$
```

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

`sudo yum install docker -y`

```
[ec2-user@ip-172-31-92-18 ~]$ sudo yum install docker -y
Last metadata expiration check: 0:09:56 ago on Wed Sep 11 15:19:39 2024.
Dependencies resolved.
```

```
=====
Package                                         Architecture
=====
Installing:
docker                                         x86_64
Installing dependencies:
containerd                                     x86_64
iptables-libs                                 x86_64
iptables-nft                                  x86_64
libcgroup                                     x86_64
libnetfilter_conntrack                        x86_64
libnfnetlink                                  x86_64
libnftnl                                       x86_64
pigz                                           x86_64
runc                                           x86_64
```

Transaction Summary

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"  
}  
EOF
```

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

Install Kubernetes on all 3 machines

SELinux needs to be disabled before configuring kubelet

- `sudo setenforce 0`
- `sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config`

```
[ec2-user@ip-172-31-81-63 docker]$ sudo setenforce 0  
[ec2-user@ip-172-31-81-63 docker]$ sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
```

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/repo  
md.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-172-31-81-63 docker]$ sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Last metadata expiration check: 0:01:34 ago on Wed Sep 11 15:39:05 2024.
Dependencies resolved.
=====
Package                                Architecture                               Version
=====
Installing:
kubeadm                                x86_64                                    1.30.4-150500.1.1
kubectl                                x86_64                                    1.30.4-150500.1.1
kubelet                                x86_64                                    1.30.4-150500.1.1
Installing dependencies:
conntrack-tools                        x86_64                                    1.4.6-2.amzn2023.0.2
cri-tools                              x86_64                                    1.30.1-150500.1.1
kubernetes-cni                         x86_64                                    1.4.0-150500.1.1
libnetfilter_cthelper                  x86_64                                    1.0.0-21.amzn2023.0.2
libnetfilter_cttimeout                 x86_64                                    1.0.0-19.amzn2023.0.2
libnetfilter_queue                     x86_64                                    1.0.5-2.amzn2023.0.2
socat                                  x86_64                                    1.7.4.2-1.amzn2023.0.2
Transaction Summary
=====
Install 10 Packages
```

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`

1. 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 -g) $HOME/.kube/config
```

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

My personal join key:

```
kubeadm join 172.31.92.157:6443 --token x4sw6q.sbckmhm5gkoubquv \
--discovery-token-ca-cert-hash
sha256:24c005691fcab2260667ee43384d46afd4b2b27401e82c01550798a0d8f98950
```

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

D15C

4

~~This step gives~~
~~error:~~

kubectl apply -f

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