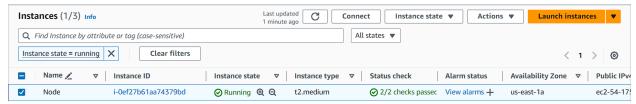
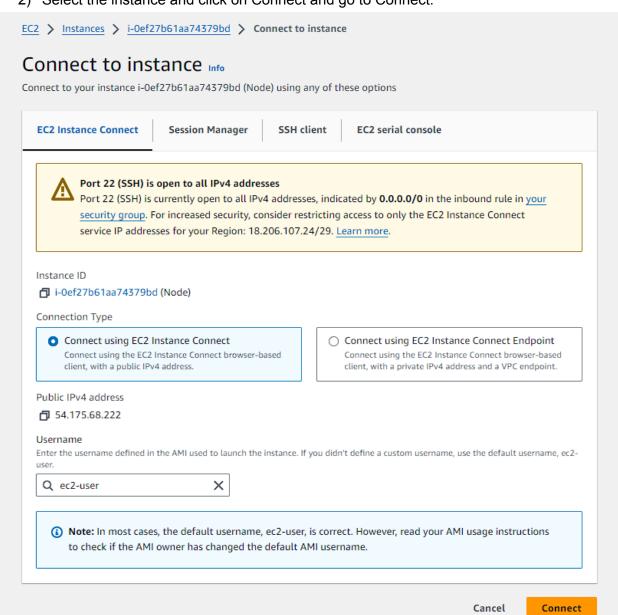
Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy your first Kubernetes Application.

1) Create an EC2 instance and allow SSH traffic to connect.



Select the instance and click on Connect and go to Connect.



3) EC2 server is connected.

4) To install docker run the following command:

sudo yum install docker -y

```
[ec2-user@ip-172-31-22-128 ~]$ sudo yum install docker -y
Last metadata expiration check: 2:24:08 ago on Sat Sep 14 12:44:19 2024.
Package docker-25.0.6-1.amzn2023.0.2.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```

5) Configure cgroup in daemon.json file using the following commands 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</p>

```
[ec2-user@ip-172-31-30-107 ~]$ cd /etc/docker
[ec2-user@ip-172-31-30-107 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"
}</pre>
```

6) Run the following command after this: sudo systemctl enable docker sudo systemctl daemon-reload sudo systemctl restart docker

```
[ec2-user@ip-172-31-30-107 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.
```

7) Check for installation of docker by docker -v command.

```
[ec2-user@ip-172-31-30-107 docker]$ docker -v
Docker version 25.0.5, build 5dc9bcc
```

## **Install Kubernetes**

- 8) Disable SELinux before configuring kubelet
  - sudo setenforce 0
  - sudo sed -i 's/^SELINUX-enforcing\$/SELINUX=permissive/' /etc/selinux/config

```
[ec2-user@ip-172-31-30-107 docker]$ sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
```

 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.k
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
```

10) Run the commands to update and install kubernetes packages sudo yum update sudo yum install -y kubelet kubeadm kubectl --disableexcludes kubernetes

```
ec2-user@ip-172-31-31-241 docker]$ sudo yum install -y kubelet kubeadm k
ast metadata expiration check: 0:00:46 ago on Sat Sep 14 16:39:07 2024.
ependencies resolved.
                                                                                     Architecture
                                                                                                                                                                                                                                                                                                           Size
                                                                                                                                             Version
                                                                                                                                                                                                                                        Repository
 stalling:
                                                                                     x86_64
x86_64
x86_64
                                                                                                                                             1.31.1-150500.1.1
1.31.1-150500.1.1
1.31.1-150500.1.1
 stalling dependencies:
ransaction Summary
 otal download size: 51 M
stalled size: 269 M
wwnloading Packages:
```

11) 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-172-31-31-241 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
```

## 12) Initialize the kubecluster

sudo kubeadm init --pod-network-cidr-10.244.0.0/16

```
sudo kubeadm init --pod-network-cidr-10.244.0.0/16

[ec2-user@ip-172-31-31-241 docker]$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16

[init] Using Kubernetes version: v1.31.0

[preflight] Running pre-flight checks

[MARNING FileExisting-socat]: socat not found in system path

[MARNING FileExisting-socat]: socat not found in system path

[MARNING FileExisting-socat]: socat not found in system path

[MARNING Service-Kubelet]: kubelet service is not enabled, please run 'systemctl enable kubelet.service'

[preflight] Pulling images required for setting up a Kubernetes cluster

[preflight] You can also perform this action beforehand using 'kubeadm config images pull'

[preflight] You can also perform this action beforehand using 'kubeadm config images pull'

[wold 16:43:42.619918 27:90 checks.go:245] detected that the sandbox image "registry.k8s.io/pause:3.8" of the container runtime is inconsistent with that used by yubeadm.It is recommended to use "registry.k8s.io/pause:3.10" as the CRI sandbox image.

[certs] Using certificateDir folder "/etc/kubernetes/pki"

[certs] Generating "apiserver" certificate and key

[certs] Generating "apiserver" certificate and key

[certs] Generating "apiserver" certificate and key

[certs] Generating "apiserver"-kubelet-client" certificate and key

[certs] Generating "front-proxy-client" certificate and key

[certs] Generating "front-proxy-client" certificate and key

[certs] Generating "front-proxy-client" certificate and key

[certs] Generating "etcd/server" certificate and key

[certs] Generating "etcd/peer" certificate and key

[certs] Generating "a
```

13) Copy the Join Command-

```
Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.31.31.241:6443 --token x6zuwa.bafjhtm4fqxp4yi8 \
--discovery-token-ca-cert-hash sha256:ad0a2b0eb8318975f42be705f750f923adc01bd102ebd7d93401df81429ef5a5
```

kubeadm join 172.31.31.241:6443 --token x6zuwa.bafjhtm4fqxp4yi8 \

--discovery-token-ca-cert-hash

sha256:ad0a2b0eb8318975f42be705f750f923adc01bd102ebd7d93401df81429ef5a5

14) Run the following commands

```
[ec2-user@ip-172-31-31-241 docker] mkdir -p $HOME/.kube sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config sudo chown $(id -u):$(id -g) $HOME/.kube/config cp: overwrite '/home/ec2-user/.kube/config'? yes
```

15) Add a common network plugin called Flannel as mentioned in the code below: kubectl apply -f

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

```
c2-user@ip-172-31-20-245 dockekubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
mespace/kube-flannel created
usterrole.rbac.authorization.k@s.io/flannel created
usterrolebinding.rbac.authorization.k@s.io/flannel created
rviceaccount/flannel created
nfigmap/kube-flannel-cfg created
emonset.apps/kube-flannel-ds created
co-user@ip-172-31-20-245 docker]$
```

16) Check for creation of pods

```
[ec2-user@ip-172-31-20-245 ~]$ kubectl get pods
NAME READY STATUS RESTARTS AGE
nginx 0/1 Pending 0 80s
```

17) To change the state from pending to running, use the following command kubectl describe pod nginx

This command will help to describe the pods it gives reason for failure as it shows the untolerated taints which need to be untainted.

```
nginx:
Image:
                     nginx:1.14.2
    Port:
Host Port:
    Mounts:
        /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-dmncs (ro)
  enditions:
 Type Status
PodScheduled False
  kube-api-access-dmncs:
    Type:
                                    Projected (a volume that contains injected data from multiple sources)
     TokenExpirationSeconds: 3607
    ConfigMapOptional:
DownwardAPI:
                                   <ni1>
                                   BestEffort
 ode-Selectors:
                                   node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
 olerations:
 vents:
                                                                    Message
 Warning FailedScheduling 2m47s default-scheduler 0/1 nodes are available: 1 node(s) had untolerated taint (node-role.kubernete: o/control-plane: ). preemption: 0/1 nodes are available: 1 Preemption is not helpful for scheduling.
[ec2-user@ip-172-31-20-245 docker]$ kubectl taint nodes ip-172-31-20-245.ec2.internal node-role.kubernetes.io/control-plane-
node/ip-172-31-20-245.ec2.internal untainted
```

## 18) Check the status of pods.

```
[ec2-user@ip-172-31-20-245 docker]$ kubectl get pods

NAME READY STATUS RESTARTS AGE

nginx 1/1 Running 0 4m3s
```

```
[ec2-user@ip-172-31-20-245 docker]$ kubectl port-forward nginx 8081:80 Forwarding from 127.0.0.1:8081 -> 80 Forwarding from [::1]:8081 -> 80
```

## Conclusion:

We successfully configured Kubernetes environment on Amazon Linux EC2 instance. We installed Docker and adjusted its settings to use systemd for cgroup management. Then, we installed Kubernetes. We desabled SELinux, and added the Kubernetes repository. Then we installed the necessary components. Then added a network plugin and deployed on Nginx server. We also addressed issues related to pod scheduling and port forwarding, ensuring the Nginx pod.