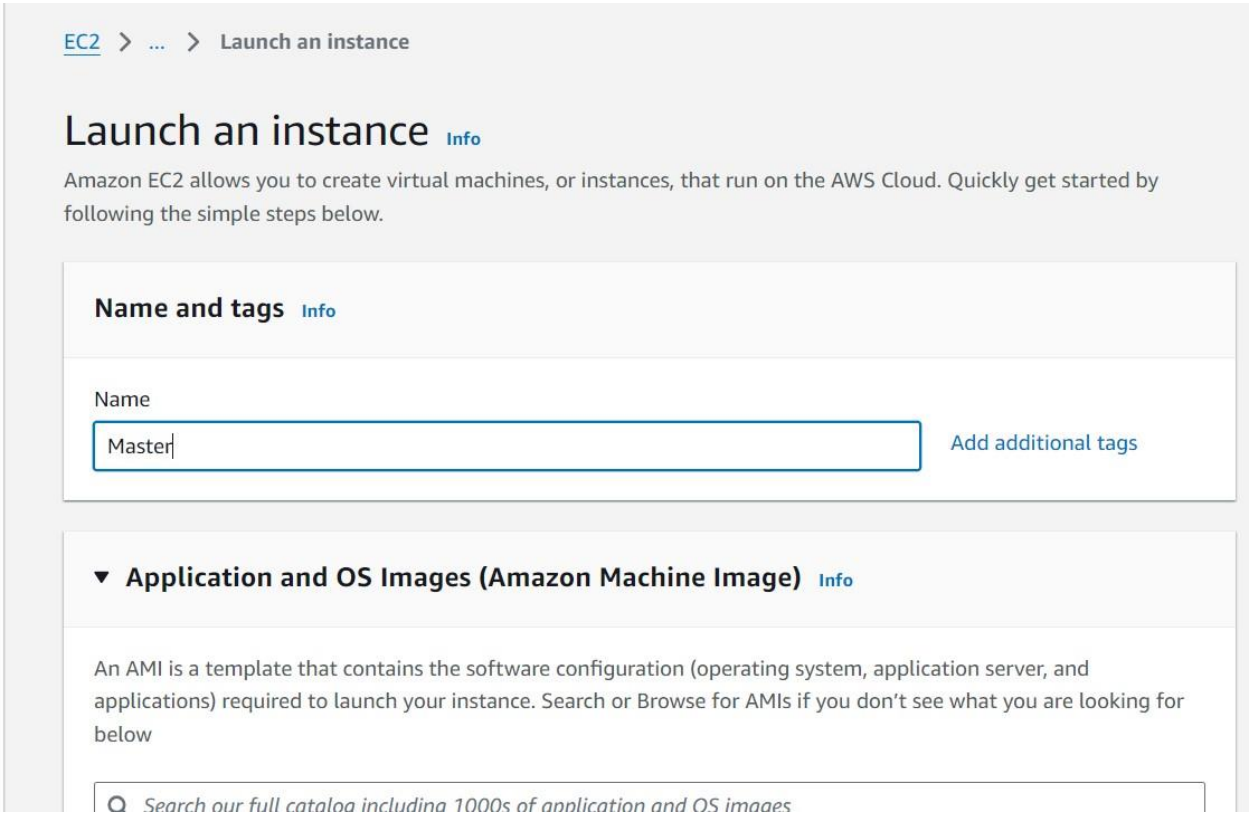
CaseStudy

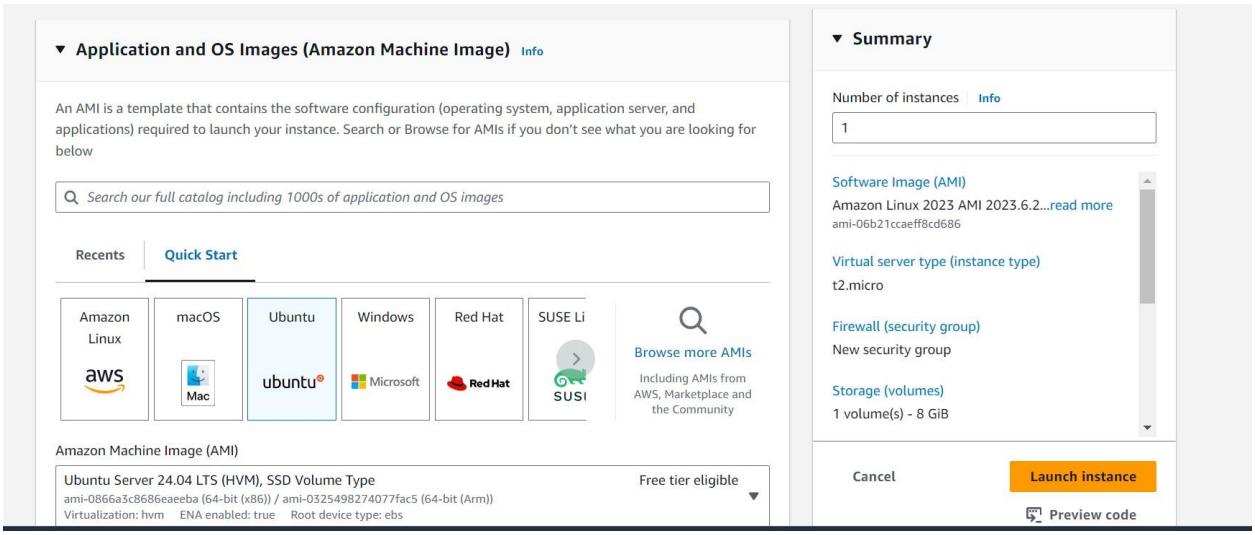
Kubernetes Deployment

A Kubernetes Deployment is used to tell Kubernetes how to create or modify instances of the pods that hold a containerized application. Deployments can scale the number of replica pods, enable the rollout of updated code in a controlled manner, or roll back to an earlier deployment version if necessary.

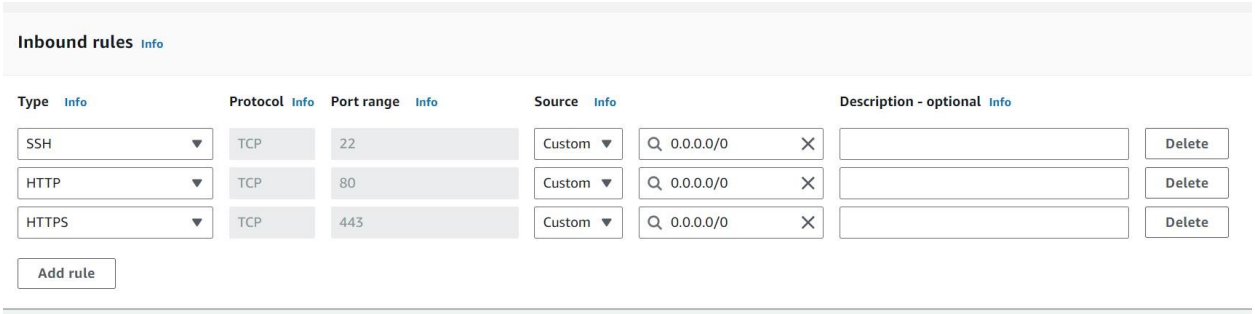
Steps:

1. Create an EC2 Ubuntu Instance on AWS.





2. Edit the Security Group Inbound Rules to allow SSH



3. SSH into the machine **ssh -i <keyname>.pem ubuntu@<public_ip_address>**

```
USER@DESKTOP-LI6Q7A6 MINGW64 ~
$ cd Downloads/

USER@DESKTOP-LI6Q7A6 MINGW64 ~/Downloads
$ ssh -i "sahil13.pem" ubuntu@ec2-18-117-238-194.us-east-2.compute.amazonaws.com
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1016-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Wed Oct 23 02:27:54 UTC 2024

  System load:  0.02              Processes:             152
  Usage of /:   57.9% of 6.71GB   Users logged in:       0
  Memory usage: 10%               IPv4 address for enX0: 172.31.27.63
  Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

32 updates can be applied immediately.
21 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


Last login: Tue Oct 22 18:56:31 2024 from 106.220.92.200
ubuntu@ip-172-31-27-63:~$
```

Step 4: Run the below commands to install and setup Docker. curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add - curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

```
ubuntu@ip-172-31-27-63:~$ sudo apt install docker.io -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 slirp4netns
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  bridge-utils dns-root-data dnsmasq-base ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-buildx docker-compose-v2 docker-doc rinse zfs-fuse | zfsutils
The following packages will be REMOVED:
  docker-ce-cli
The following NEW packages will be installed:
  bridge-utils dns-root-data dnsmasq-base docker.io ubuntu-fan
0 upgraded, 5 newly installed, 1 to remove and 27 not upgraded.
Need to get 29.5 MB of archives.
After this operation, 68.6 MB of additional disk space will be used.
Get:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble/main amd64 bridge-utils amd64 1.7.1-1ubuntu2 [33.9 kB]
Get:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble/main amd64 dns-root-data all 2023112702~willsync1 [4450 B]
Get:3 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble/main amd64 dnsmasq-base amd64 2.90-2build2 [375 kB]
Get:4 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 docker.io amd64 24.0.7-0ubuntu4.1 [29.1 MB]
Get:5 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 ubuntu-fan all 0.12.16 [35.2 kB]
Fetched 29.5 MB in 0s (83.5 MB/s)
Preconfiguring packages ...
(Reading database ... 68248 files and directories currently installed.)
```

sudo apt-get install -y docker-ce

```
ubuntu@ip-172-31-27-63:~$ sudo apt install docker-ce
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  containerd.io docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 li
  slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-plugin l
  slirp4netns
0 upgraded, 10 newly installed, 0 to remove and 27 not upgraded.
Need to get 123 MB of archives.
After this operation, 442 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]
Get:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libltdl7 amd64 2.4.7-7build1 [40.3 kB]
Get:3 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libslirp0 amd64 4.7.0-1ubuntu3 [63.8 kB]
Get:4 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 slirp4netns amd64 1.2.1-1build2 [34.
Get:5 https://download.docker.com/linux/ubuntu noble/stable amd64 containerd.io amd64 1.7.22-1 [29.5 MB]
Get:6 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-buildx-plugin amd64 0.17.1-1~ubuntu.2
Get:7 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-ce-cli amd64 5:27.3.1-1~ubuntu.24.04~
Get:8 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-ce amd64 5:27.3.1-1~ubuntu.24.04~nobl
Get:9 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-ce-rootless-extras amd64 5:27.3.1-1~u
B]
Get:10 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-compose-plugin amd64 2.29.7-1~ubuntu
Fetched 123 MB in 2s (79.0 MB/s)
Selecting previously unselected package pigz.
(Reading database ... 67840 files and directories currently installed.)
Preparing to unpack .../0-pigz_2.8-1_amd64.deb ...
Unpacking pigz (2.8-1)
```

sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF

```
ubuntu@ip-172-31-27-63:~$ sudo mkdir -p /etc/docker
ubuntu@ip-172-31-27-63:~$ cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
  "exec-opts": ["native.cgroupdriver=systemd"]
}
```

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

```
ubuntu@ip-172-31-27-63:~$ sudo systemctl enable docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
ubuntu@ip-172-31-27-63:~$ sudo systemctl daemon-reload
ubuntu@ip-172-31-27-63:~$ sudo systemctl restart docker
```

Step 5: Run the below command to install Kubernets. curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

```
ubuntu@ip-172-31-80-240:~$ echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sud
o tee /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
ubuntu@ip-172-31-80-240:~$
```

sudo apt-get update sudo apt-get install -y kubelet kubeadm kubectl sudo apt-mark hold kubelet kubeadm kubectl

```
ubuntu@ip-172-31-27-63:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearm
or -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
ubuntu@ip-172-31-27-63:~$ echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.i
o/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
ubuntu@ip-172-31-27-63:~$ sudo apt-get update
Hit:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease [1186 B
]
Get:6 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  Packages [4865 B]
Get:8 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [13.5 kB]
Fetched 146 kB in 1s (192 kB/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring
(/etc/apt/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
ubuntu@ip-172-31-27-63:~$ sudo apt-get install -y kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

sudo systemctl enable --now kubelet

```
ubuntu@ip-172-31-27-63:~$ sudo systemctl enable --now kubelet
ubuntu@ip-172-31-27-63:~$ sudo systemctl enable --now kubelet
ubuntu@ip-172-31-27-63:~$ ^C
```

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

```
ubuntu@ip-172-31-27-63:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.1
[preflight] Running pre-flight checks
W1022 18:40:01.161701    6128 checks.go:1080] [preflight] WARNING: Couldn't create the interface used for talki
ng to the container runtime: failed to create new CRI runtime service: validate service connection: validate CR
I v1 runtime API for endpoint "unix:///var/run/containerd/containerd.sock": rpc error: code = Unimplemented des
c = unknown service runtime.v1.RuntimeService
        [WARNING FileExisting-socat]: socat 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'
error execution phase preflight: [preflight] Some fatal errors occurred:
failed to create new CRI runtime service: validate service connection: validate CRI v1 runtime API for endpoint
 "unix:///var/run/containerd/containerd.sock": rpc error: code = Unimplemented desc = unknown service runtime.v
1.RuntimeService[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-prefl
ight-errors=...`
To see the stack trace of this error execute with --v=5 or higher
```

sudo apt-get install -y containerd

```
ubuntu@ip-172-31-27-63:~$ sudo apt-get install -y containerd
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz
  slirp4netns
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  runc
The following packages will be REMOVED:
  containerd.io docker-ce
The following NEW packages will be installed:
  containerd runc
0 upgraded, 2 newly installed, 2 to remove and 27 not upgraded.
Need to get 47.2 MB of archives.
After this operation, 53.1 MB disk space will be freed.
Get:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 runc amd64 1.1.12-0ubuntu3.1 [859
9 kB]
Get:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 containerd amd64 1.7.12-0ubuntu4.
1 [38.6 MB]
Fetched 47.2 MB in 1s (57.1 MB/s)
(Reading database ... 68163 files and directories currently installed.)
Removing docker-ce (5:27.3.1-1~ubuntu.24.04~noble) ...
Removing containerd.io (1.7.22-1) ...
Selecting previously unselected package runc.
(Reading database ... 68143 files and directories currently installed.)
Preparing to unpack .../runc_1.1.12-0ubuntu3.1_amd64.deb ...
Unpacking runc (1.1.12-0ubuntu3.1) ...
Selecting previously unselected package containerd.
Preparing to unpack .../containerd_1.7.12-0ubuntu4.1_amd64.deb ...
Unpacking containerd (1.7.12-0ubuntu4.1) ...
```

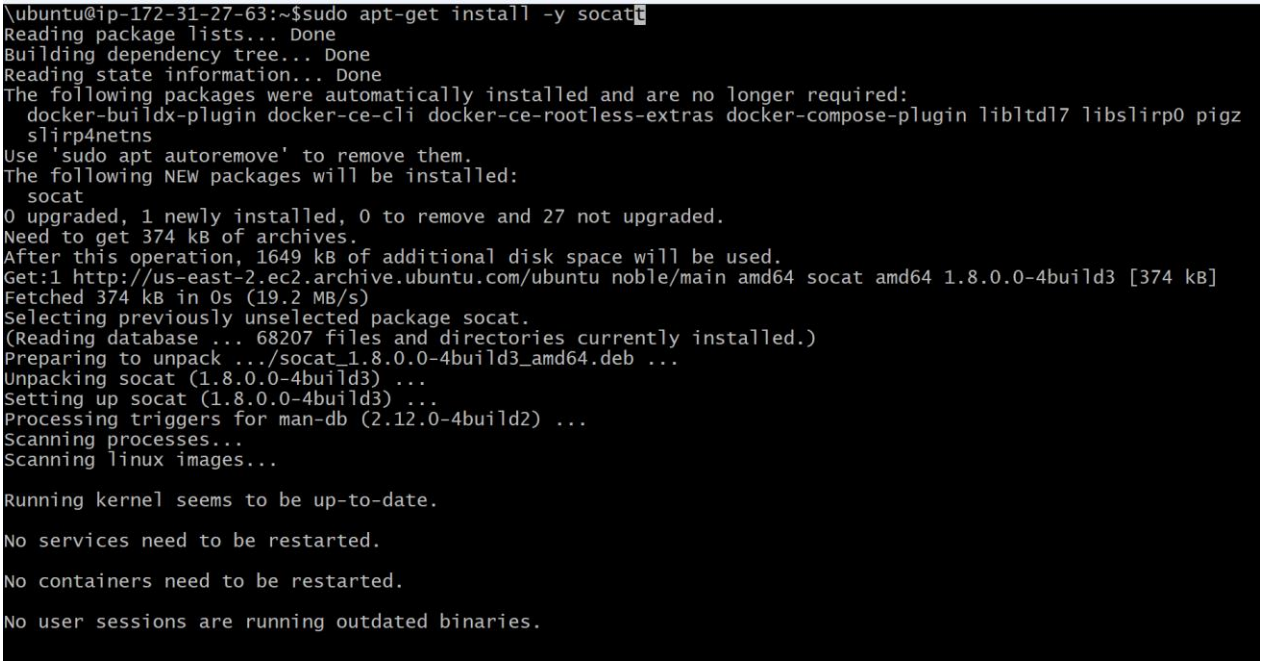
sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml

```
ubuntu@ip-172-31-27-63:~$ sudo containerd config default | sudo tee /etc/containerd/config.toml
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
temp = ""
version = 2

[cgroup]
  path = ""

[debug]
  address = ""
  format = ""
  gid = 0
  level = ""
  uid = 0

[grpc]
  address = "/run/containerd/containerd.sock"
  gid = 0
  max_recv_message_size = 16777216
  max_send_message_size = 16777216
  tcp_address = ""
  tcp_tls_ca = ""
  tcp_tls_cert = ""
  tcp_tls_key = ""
  uid = 0
```

sudo systemctl restart containerd sudo
systemctl enable containerd sudo
systemctl status containerd

```
ubuntu@ip-172-31-27-63:~$ sudo systemctl status containerd
● containerd.service - containerd container runtime
     Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: enabled)
     Active: active (running) since Tue 2024-10-22 18:29:06 UTC; 11min ago
       Docs: https://containerd.io
   Main PID: 3266 (containerd)
      Tasks: 7
     Memory: 12.6M (peak: 13.5M)
        CPU: 547ms
     CGroup: /system.slice/containerd.service
             └─3266 /usr/bin/containerd
```

sudo apt-get install -y socat



Step 6: Initialize the Kubecluster

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

```
ubuntu@ip-172-31-27-63:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.1
[preflight] Running pre-flight checks
[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'
W1022 18:48:32.092564    7343 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the
 container runtime is inconsistent with that used by kubeadm.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 "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [ip-172-31-27-63 kubernetes kubernetes.default kubernete
s.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 172.31.27.63]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [ip-172-31-27-63 localhost] and IPs [172.31.27.63 127.
0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-27-63 localhost] and IPs [172.31.27.63 127.0.
0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "super-admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
```

```
ubuntu@ip-172-31-80-240:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.1
[preflight] Running pre-flight checks
[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'
W1020 10:07:42.067638    6134 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the container runtime is inconsistent wi
h that used by kubeadm.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 "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [ip-172-31-80-240 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc
cluster.local] and IPs [10.96.0.1 172.31.80.240]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [ip-172-31-80-240 localhost] and IPs [172.31.80.240 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-80-240 localhost] and IPs [172.31.80.240 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "super-admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
```

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

```
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
```

Add a common networking plugin called flannel as mentioned in the code.
kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

```
ubuntu@ip-172-31-27-63:~$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentatio
n/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
```

kubectl apply -f https://k8s.io/examples/application/deployment.yaml

```
ubuntu@ip-172-31-27-63:~$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
```

kubectl get pods

```
ubuntu@ip-172-31-27-63:~$ kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-6h7w5   0/1     Pending   0          24s
nginx-deployment-d556bf558-sb25q   0/1     Pending   0          24s
```

POD_NAME =$(kubectl get pods -l app=nginx -o jsonpath = “{.item[0].metadata.name}”)

```
ubuntu@ip-172-31-27-63:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8080:80
error: unable to forward port because pod is not running. Current status=Pending
```

kubectl get nodes

```
ubuntu@ip-172-31-80-240:~$  kubectl get nodes
NAME               STATUS   ROLES           AGE     VERSION
ip-172-31-80-240   Ready    control-plane   3m46s   v1.31.1
```

POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8080:80

```
ubuntu@ip-172-31-80-240:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
ubuntu@ip-172-31-80-240:~$ kubectl port-forward $POD_NAME 8080:80
error: unable to forward port because pod is not running. Current status=Pending
```

command kubectl tain tnodes--all node-role.kubernetes.io/control-plane-node/ip-172-3120-171 untainted

```
ubuntu@ip-172-31-80-240:~$ command kubectl taint nodes --all node-role.kubernetes.io/control-plane:NoSchedule-
node/ip-172-31-80-240 untainted
```

kubectl get nodes

```
ubuntu@ip-172-31-80-240:~$ kubectl get nodes
NAME               STATUS   ROLES           AGE     VERSION
ip-172-31-80-240   Ready    control-plane   9m52s   v1.31.1
```

kubectl get pods

```
ubuntu@ip-172-31-27-63:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-6h7w5    1/1     Running   0          3m5s
nginx-deployment-d556bf558-sb25q    1/1     Running   0          3m5s
```

POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8080:80

```
ubuntu@ip-172-31-27-63:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
Handling connection for 8080
```

Step 8: Verify your deployment
Open up a new terminal and ssh to your EC2 instance.
Then, use this curl command to check if the Nginx server is running. curl –head http://127.0.0.1:8080

```
USER@DESKTOP-LI6Q7A6 MINGW64 ~/Downloads
$ curl --head http://127.0.0.1:8080
HTTP/1.1 403 Forbidden
Date: Tue, 22 Oct 2024 18:56:01 GMT
X-Content-Type-Options: nosniff
Set-Cookie: JSESSIONID.f8c738e8=node01e1osv5kd7rrakww61tadvkny0.node0; Path=/; HttpOnly
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Content-Type: text/html;charset=utf-8
X-Hudson: 1.395
X-Jenkins: 2.462.2
X-Jenkins-Session: b9e73b11
Transfer-Encoding: chunked
Server: Jetty(10.0.20)
```

kubectl get services

```
ubuntu@ip-172-31-27-63:~$ kubectl get services
NAME               TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes         ClusterIP   10.96.0.1        <none>        443/TCP        15m
nginx-deployment   NodePort    10.100.120.151   <none>        80:32531/TCP   5m23s
```

kubectl create deployment nginx –-image=nginx

```
ubuntu@ip-172-31-27-63:~$ kubectl create deployment nginx --image=nginx
deployment.apps/nginx created
```

kubectl get deployments

```
ubuntu@ip-172-31-27-63:~$ kubectl get deployments
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx              1/1     1            1           40s
nginx-deployment   2/2     2            2           14m
```

kubectl expose deployment nginx --type=NodePort --port=80

```
ubuntu@ip-172-31-27-63:~$ kubectl expose deployment nginx --type=NodePort --port=80
service/nginx exposed
```

Nginx server is running successfully on the EC2 instance, and it's accessible locally via localhost on port 31801.

curl http://127.0.0.1/31801

```
ubuntu@ip-172-31-80-240:~$ curl http://localhost:31301
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installe
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Conclusion:

In this experiment, we successfully set up Kubernetes and Docker on an AWS EC2 Ubuntu instance, configured the necessary settings, and initialized a Kubernetes cluster. We deployed an Nginx server using a Kubernetes Deployment and implemented the Flannel networking plugin for pod communication. By checking the pod status and forwarding ports, we were able to access the Nginx server locally. The successful `200 OK` response from the `curl` command confirmed that the deployment was functioning correctly. This setup highlighted key Kubernetes operations, such as cluster management, application deployment, and verification, demonstrating the effectiveness of Kubernetes in orchestrating containerized applications efficiently.