

## Experiment 4

**Aim:** To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.

### Procedure:

#### 1. Creation Of EC-2 instance

Create an EC2 AWS Linux instance on AWS .also edit the Security Group Inbound Rules to allow SSH. then select the t2.micro instance type

Name  
my\_instance [Add additional tags](#)

▼ **Application and OS Images (Amazon Machine Image)** [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Q Search our full catalog including 1000s of application and OS images

Recents | My AMIs | **Quick Start**

Amazon Linux macOS Ubuntu Windows Red Hat

aws Mac ubuntu Microsoft Red Hat

[Browse more AMIs](#)  
Including AMIs from AWS, Marketplace and the Community

☒ **Allow SSH traffic from**  
Helps you connect to your instance

Anywhere  
0.0.0.0/0 ▼

☐ **Allow HTTPS traffic from the internet**  
To set up an endpoint, for example when creating a web server

☐ **Allow HTTP traffic from the internet**  
To set up an endpoint, for example when creating a web server

- Thus Kuber named -instance gets created. Then click on Id of that instance then click on connect button you will see this.

## Info

Connect to your instance `i-0f68279e506401ef2` (insty) using any of these options

## EC2 serial console

 i-0f68279e506401ef2 (insty)

- ```
❏ chmod 400 "key2.pem"
```

```
ssh -i "key2.pem" ec2-user@ec2-54-82-44-168.compute-1.amazonaws.com
```

- Chmod 400 “keyname.pem”

ssh -i <keyname>.pem ubuntu@<public\_ip\_address> copy it and then connect it and run the following command for establishing connection.(I have entered this command on git bash where i entered in downloads where server.pem is stored then as the key is not accessible hence we need to change its mode using chmod 400 “key name.pem”. Then use the given command for making connections).

```
Anshi@anshi MINGW64 ~
$ cd Downloads
```

```
Anshi@anshi MINGW64 ~/Downloads
$ chmod 400 "key2.pem"
```

```
Anshi@anshi MINGW64 ~/Downloads
$ ec2-3-85-239-227.compute-1.amazonaws.com
bash: ec2-3-85-239-227.compute-1.amazonaws.com: command not found
```

```
Anshi@anshi MINGW64 ~/Downloads
$ ssh -i "key2.pem" ec2-user@ec2-3-85-239-227.compute-1.amazonaws.com
The authenticity of host 'ec2-3-85-239-227.compute-1.amazonaws.com (3.85.239.227)' can't be established.
ED25519 key fingerprint is SHA256:3ytsjvZbZSc5N7KSAwq0IAh/LRz+zwWqkIlf4gWKjfY.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-85-239-227.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
```



```

graph TD
    Root["~ #"] --> Node1["~ #####"]
    Root --> Node2["~ #####"]
    Node1 --> Node3["~ #####"]
    Node1 --> Node4["~ #"]
    Node2 --> Node5["~ #"]
    Node3 --> Node6["~ #"]
    Node4 --> Node7["~ #"]
    Node5 --> Node8["~ #"]
    Node6 --> Node9["~ #"]
    Node7 --> Node10["~ #"]
    Node8 --> Node11["~ #"]
    Node9 --> Node12["~ #"]
    Node10 --> Node13["~ #"]
    Node11 --> Node14["~ #"]
    Node12 --> Node15["~ #"]
    Node13 --> Node16["~ #"]
    Node14 --> Node17["~ #"]
    Node15 --> Node18["~ #"]
    Node16 --> Node19["~ #"]
    Node17 --> Node20["~ #"]
    Node18 --> Node21["~ #"]
    Node19 --> Node22["~ #"]
    Node20 --> Node23["~ #"]
    Node21 --> Node24["~ #"]
    Node22 --> Node25["~ #"]
    Node23 --> Node26["~ #"]
    Node24 --> Node27["~ #"]
    Node25 --> Node28["~ #"]
    Node26 --> Node29["~ #"]
    Node27 --> Node30["~ #"]
    Node28 --> Node31["~ #"]
    Node29 --> Node32["~ #"]
    Node30 --> Node33["~ #"]
    Node31 --> Node34["~ #"]
    Node32 --> Node35["~ #"]
    Node33 --> Node36["~ #"]
    Node34 --> Node37["~ #"]
    Node35 --> Node38["~ #"]
    Node36 --> Node39["~ #"]
    Node37 --> Node40["~ #"]
    Node38 --> Node41["~ #"]
    Node39 --> Node42["~ #"]
    Node40 --> Node43["~ #"]
    Node41 --> Node44["~ #"]
    Node42 --> Node45["~ #"]
    Node43 --> Node46["~ #"]
    Node44 --> Node47["~ #"]
    Node45 --> Node48["~ #"]
    Node46 --> Node49["~ #"]
    Node47 --> Node50["~ #"]
    Node48 --> Node51["~ #"]
    Node49 --> Node52["~ #"]
    Node50 --> Node53["~ #"]
    Node51 --> Node54["~ #"]
    Node52 --> Node55["~ #"]
    Node53 --> Node56["~ #"]
    Node54 --> Node57["~ #"]
    Node55 --> Node58["~ #"]
    Node56 --> Node59["~ #"]
    Node57 --> Node60["~ #"]
    Node58 --> Node61["~ #"]
    Node59 --> Node62["~ #"]
    Node60 --> Node63["~ #"]
    Node61 --> Node64["~ #"]
    Node62 --> Node65["~ #"]
    Node63 --> Node66["~ #"]
    Node64 --> Node67["~ #"]
    Node65 --> Node68["~ #"]
    Node66 --> Node69["~ #"]
    Node67 --> Node70["~ #"]
    Node68 --> Node71["~ #"]
    Node69 --> Node72["~ #"]
    Node70 --> Node73["~ #"]
    Node71 --> Node74["~ #"]
    Node72 --> Node75["~ #"]
    Node73 --> Node76["~ #"]
    Node74 --> Node77["~ #"]
    Node75 --> Node78["~ #"]
    Node76 --> Node79["~ #"]
    Node77 --> Node80["~ #"]
    Node78 --> Node81["~ #"]
    Node79 --> Node82["~ #"]
    Node80 --> Node83["~ #"]
    Node81 --> Node84["~ #"]
    Node82 --> Node85["~ #"]
    Node83 --> Node86["~ #"]
    Node84 --> Node87["~ #"]
    Node85 --> Node88["~ #"]
    Node86 --> Node89["~ #"]
    Node87 --> Node90["~ #"]
    Node88 --> Node91["~ #"]
    Node89 --> Node92["~ #"]
    Node90 --> Node93["~ #"]
    Node91 --> Node94["~ #"]
    Node92 --> Node95["~ #"]
    Node93 --> Node96["~ #"]
    Node94 --> Node97["~ #"]
    Node95 --> Node98["~ #"]
    Node96 --> Node99["~ #"]
    Node97 --> Node100["~ #"]
    
```

Amazon Linux 2023

<https://aws.amazon.com/linux/amazon-linux-2023>

2. Installation of Docker

- 1. . For installation of Docker into the machines run the following command: `sudo yum install docker -y`

```
~/m/
[ec2-user@ip-172-31-26-174 ~]$ sudo yum install docker -y
Last metadata expiration check: 0:05:13 ago on Fri Sep 13 13:17:25 2024.
Dependencies resolved.
```

| Package                  | Architecture | Version               | Repository  |
|--------------------------|--------------|-----------------------|-------------|
| Installing:<br>docker    | x86_64       | 25.0.6-1.amzn2023.0.2 | amazonlinux |
| Installing dependencies: |              |                       |             |
| containerd               | x86_64       | 1.7.20-1.amzn2023.0.1 | amazonlinux |
| iptables-libs            | x86_64       | 1.8.8-3.amzn2023.0.2  | amazonlinux |
| iptables-nft             | x86_64       | 1.8.8-3.amzn2023.0.2  | amazonlinux |
| libcgroup                | x86_64       | 3.0-1.amzn2023.0.1    | amazonlinux |
| libnetfilter_conntrack   | x86_64       | 1.0.8-2.amzn2023.0.2  | amazonlinux |
| libnftnl                 | x86_64       | 1.0.1-19.amzn2023.0.2 | amazonlinux |
| libnftnl                 | x86_64       | 1.2.2-2.amzn2023.0.2  | amazonlinux |
| pigz                     | x86_64       | 2.5-1.amzn2023.0.3    | amazonlinux |
| runc                     | x86_64       | 1.1.13-1.amzn2023.0.1 | amazonlinux |
| 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
```

```
[ec2-user@ip-172-31-26-174 ~]$ cd /etc/docker
[ec2-user@ip-172-31-26-174 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"
}
- - - - -
```

- Then after this run the following command to enable and start docker and also to load the daemon.json file.

```
sudo systemctl enable docker
```

```
sudo systemctl daemon-reload
```

```
sudo systemctl restart docker
```

```
[ec2-user@ip-172-31-26-174 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-172-31-26-174 docker]$ sudo systemctl daemon-reload
[ec2-user@ip-172-31-26-174 docker]$ sudo systemctl restart docker
[ec2-user@ip-172-31-26-174 docker]$ docker -v
Docker version 25.0.5, build 5dc9bcc
```

- docker -v

```
[ec2-user@ip-172-31-80-126 docker]$ docker -v
Docker version 25.0.5, build 5dc9bcc
```

### 3. Then Install Kubernetes with the following command.

- SELinux needs to be disabled before configuring kubelet thus run the following command

```
sudo setenforce 0
```

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

```
[ec2-user@ip-172-31-26-174 docker]$ sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
```

- Here We are adding kubernetes using the repository whose command is given below. 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/repomd.xml.key
```

```
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
```

```
EOF
```

```
[ec2-user@ip-172-31-26-174 docker]$ sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[ec2-user@ip-172-31-26-174 docker]$ 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/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
[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/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
```

- After that Run following command to make the updation and also to install kubelet, kubeadm, kubectl:

**sudo yum update**

```
[ec2-user@ip-172-31-80-126 docker]$ sudo yum update
Kubernetes
Dependencies resolved.
Nothing to do.
Complete!
```

100 kB/s | 17 kB 00:00

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

```
[ec2-user@ip-172-31-80-126 docker]$ sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Last metadata expiration check: 0:00:10 ago on Fri Sep 13 10:31:17 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  | 17 M  |
| Installing dependencies: |              |                       |             |       |
| conntrack-tools          | x86_64       | 1.4.6-2.amzn2023.0.2  | amazonlinux | 208 k |
| cri-tools                | x86_64       | 1.30.1-150500.1.1     | kubernetes  | 8.6 M |
| kubernetes-cni           | x86_64       | 1.4.0-150500.1.1      | kubernetes  | 6.7 M |
| libnetfilter_cthelper    | x86_64       | 1.0.0-21.amzn2023.0.2 | amazonlinux | 24 k  |
| libnetfilter_cttimeout   | x86_64       | 1.0.0-19.amzn2023.0.2 | amazonlinux | 24 k  |
| libnetfilter_queue       | x86_64       | 1.0.5-2.amzn2023.0.2  | amazonlinux | 30 k  |

Transaction Summary

Install 9 Packages

```
Total
Kubernetes
Importing GPG key 0x9A296436:
  Userid : "isv:kubernetes OES Project <isv:kubernetes@build.opensuse.org>"
  Fingerprint: DE15 8144 86CD 3778 9E87 6E1A 2346 54DA 9A29 6436
  From : https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key
Key imported successfully
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing : 1/1
  Installing : kubernet
  Installing : cri-tools-1.30.1-150500.1.1.x86_64
  Installing : libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64
  Installing : libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64
  Installing : libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64
  Installing : conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64
  Running scriptlet: conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64
  Installing : kubelet-1.30.5-150500.1.1.x86_64
  Running scriptlet: kubelet-1.30.5-150500.1.1.x86_64
  Installing : kubeadm-1.30.5-150500.1.1.x86_64
  Installing : kubectl-1.30.5-150500.1.1.x86_64
  Running scriptlet: kubectl-1.30.5-150500.1.1.x86_64
  Verifying : conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64
  Verifying : libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64
  Verifying : libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64
  Verifying : libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64
  Verifying : cri-tools-1.30.1-150500.1.1.x86_64
  Verifying : kubeadm-1.30.5-150500.1.1.x86_64
  Verifying : kubectl-1.30.5-150500.1.1.x86_64
  Verifying : kubelet-1.30.5-150500.1.1.x86_64
  Verifying : kubernetes-cni-1.4.0-150500.1.1.x86_64
Installed:
  conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64
  kubectl-1.30.5-150500.1.1.x86_64
  libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64
  cri-tools-1.30.1-150500.1.1.x86_64
  kubelet-1.30.5-150500.1.1.x86_64
  kubeadm-1.30.5-150500.1.1.x86_64
  kubernetes-cni-1.4.0-150500.1.1.x86_64
  libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64
Complete!
```

64 MB/s | 53 MB 00:00  
20 kB/s | 1.7 kB 00:00

- After installing Kubernetes, we need to configure internet options to allow bridging.

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

```
[ec2-user@ip-172-31-26-174 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
```

---

#### 4. Initialize the Kubecluster

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

```
[ec2-user@ip-172-31-80-126 docker]$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
I0913 10:32:44.629146 26680 version.go:256] remote version is much newer: v1.31.0; falling back to: stable-1.30
[init] Using Kubernetes version: v1.30.4
[preflight] Running pre-flight checks
```

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 172.31.26.174:6443 --token pv0yyi.xhllqhclfjr50pt8 \
--discovery-token-ca-cert-hash sha256:8293b2f6d29de466bd859007f5adbcdb3a
ecb0c446ba09033d32a5846b3d434f
```

- copy the token and save for future use .

```
kubeadm join 172.31.26.174:6443 --token pv0yyi.xhllqhclfjr50pt8
```

```
\--discovery-token-ca-cert-hash
```

```
sha256:8293b2f6d29de466bd859007f5adbcdb3aecb0c446ba09033d32a5846b3d434f
```

- 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
```

```
[ec2-user@ip-172-31-80-126 docker]$ sudo mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

- Then, 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>

```
[ec2-user@ip-172-31-26-174 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
```

5. Now that the cluster is up and running, we can deploy our nginx server on this cluster. Apply deployment using the following command:

kubectl apply -f <https://k8s.io/examples/pods/simple-pod.yaml>

```
[ec2-user@ip-172-31-26-174 docker]$ kubectl apply -f https://k8s.io/examples/pods/simple-pod.yaml
pod/nginx created
```

Then use **kubectl get pods** to check whether the pod gets created or not.

```
[ec2-user@ip-172-31-26-174 docker]$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
nginx     0/1     Pending   0           12s
```

To convert 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.

- kubectl describe pod nginx

```
[ec2-user@ip-172-31-26-174 docker]$ kubectl describe pod nginx
Name:      nginx
Namespace: default
Priority:   0
Service Account: default
Node:      <none>
Labels:    <none>
Annotations: <none>
Status:    Pending
IP:        <none>
IPs:       <none>
Containers:
  nginx:
    Image:      nginx:1.14.2
    Port:       80/TCP
    Host Port:  0/TCP
    Environment: <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-k4lj6 (ro)
```

```

Conditions:
  Type           Status
  PodScheduled   False
Volumes:
  kube-api-access-k4lj6:
    Type:          Projected (a volume that contains injected data from m
multiple sources)
    TokenExpirationSeconds: 3607
    ConfigMapName:         kube-root-ca.crt
    ConfigMapOptional:     <nil>
    DownwardAPI:          true
QoS Class:           BestEffort
Node-Selectors:      <none>
Tolerations:         node.kubernetes.io/not-ready:NoExecute op=Exists for 3
00s
                   node.kubernetes.io/unreachable:NoExecute op=Exists for
300s
Events:
  Type           Reason              Age   From          Message
  ----           -
Warning         FailedScheduling    7s    default-scheduler  0/1 nodes are available: 1 no
de(s) had untolerated taint {node-role.kubernetes.io/control-plane: }.
preemption: 0/1 nodes are available: 1 Preemption is not helpful for scheduling.

```

- `kubectl taint nodes --all node-role.kubernetes.io/control-plane-`

```

[ec2-user@ip-172-31-26-174 ~]$ kubectl taint nodes --all node-role.kubernetes.io
/control-plane-
node/ip-172-31-26-174.ec2.internal untainted

```

6. Now check pod status is is running perform **kubectl get pods** this command.

```

[ec2-user@ip-172-31-28-70 docker]$ kubectl get pods
NAME      READY   STATUS             RESTARTS   AGE
nginx     0/1     ContainerCreating   0           39s
[ec2-user@ip-172-31-28-70 docker]$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
nginx     1/1     Running   1 (45s ago)  70s

```

7. Lastly, mention the port you want to host. Here i have used localhost 8081 then check it.

`kubectl port-forward nginx 8081:80`

```

[ec2-user@ip-172-31-26-174 ~]$ kubectl port-forward nginx 8081:80
Forwarding from 127.0.0.1:8081 -> 80
Forwarding from [::1]:8081 -> 80

```

## 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://127.0.0.1:8081`



```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\bhush\one drive 2\OneDrive\Desktop\New folder (4)> ssh -i "Master_Ec2_Key.pem" ubuntu@ec2-54-196-129-215.compute-1.amazonaws.com
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

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

System information as of Sun Sep 15 07:58:53 UTC 2024

System load: 0.15               Processes: 152
Usage of /: 55.3% of 6.71GB     Users logged in: 1
Memory usage: 20%              IPv4 address for enX0: 172.31.20.171
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.

132 updates can be applied immediately.
38 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: Sun Sep 15 07:27:23 2024 from 152.58.2.47

```

HTTP/1.1 200 OK

```

ubuntu@ip-172-31-20-171:~$ curl --head http://127.0.0.1:8080
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Sun, 15 Sep 2024 07:59:03 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Tue, 04 Dec 2018 14:44:49 GMT
Connection: keep-alive
ETag: "5c0692e1-264"
Accept-Ranges: bytes

```

If the response is 200 OK and you can see the Nginx server name, your deployment was successful. We have successfully deployed our Nginx server on our EC2 instance.

**Conclusion:** Firstly I created an EC2 AWS Linux instance successfully, then installed docker and kubernetes successfully, then initialized kubernetes which given me token and chown and mkdir command. Then I execute mkdir and chown the command successfully. Then I installed a networking plugin called flannel successfully. Then I tried to deploy nginx which initially gave an error. Then I deployed (simple-pod.yml) nginx successfully and also checked by using the get pods command, then hosted it on localhost 8081 ie http://localhost:8081 successfully.