Experiment no-4

Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy

Your First Kubernetes Application.

Theory:
Kubernetes, originally developed by Google, is an open-source container orchestration platform. It automates the deployment, scaling, and management of containerized applications, ensuring high availability and fault tolerance. Kubernetes is now the industry standard for container orchestration and is governed by the Cloud Native Computing Foundation (CNCF), with contributions from major cloud
and software providers like Google, AWS, Microsoft, IBM, Intel, Cisco, and Red Hat.

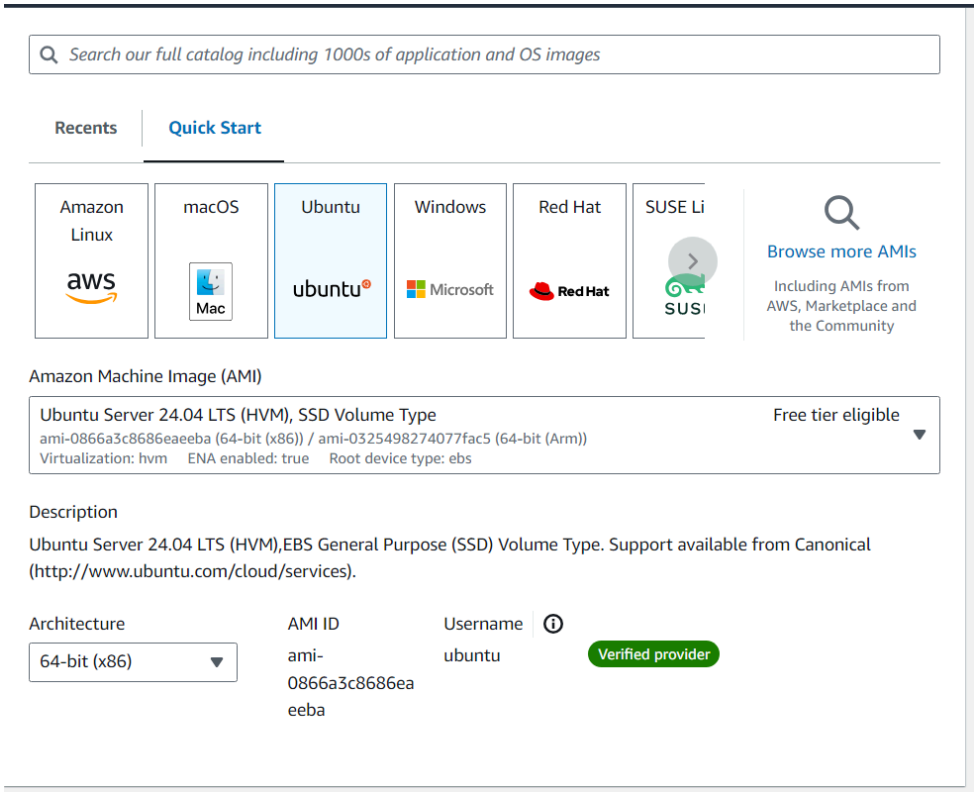
Kubernetes Deployment: Is a resource in Kubernetes that provides declarative updates for Pods and ReplicaSets. With a Deployment, you can define how many replicas of a pod should run, roll out new versions of an application, and roll back to previous versions if necessary. It ensures that the desired number of pod replicas are running at all times.

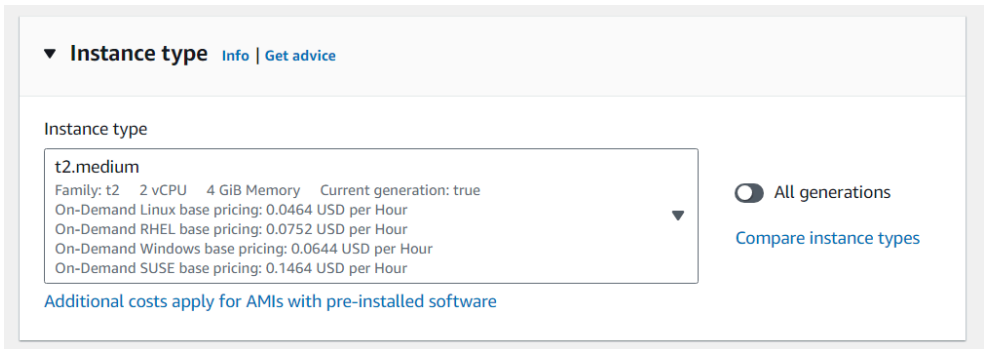
Necessary Requirements:
● EC2 Instance: The experiment required launching a t2.medium EC2 instance with 2 CPUs, as
Kubernetes demands sufficient resources for effective functioning.
● Minimum Requirements:
○ Instance Type: t2.medium
○ CPUs: 2
○ Memory: Adequate for container orchestration.

This ensured that the Kubernetes cluster had the necessary resources to function smoothly.
Note:
AWS Personal Account is preferred but we can also perform it on AWS Academy(adding some ignores in the command if any error occurs in below as the below experiment is performed on Personal Account

If You are using AWS Academy Account Errors you will face in kubeadm init command so you have to
add some ignores with this command.

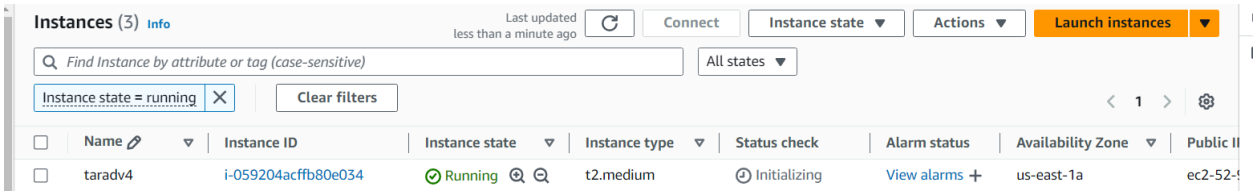
Step 1: Log in to your AWS Academy/personal account and launch a new Ec2 Instance. Select Ubuntu as AMI and t2.medium as Instance Type, create a key of type RSA with .pem extension,
and move the downloaded key to the new folder.
Note: A minimum of 2 CPUs are required so Please select t2.medium and do not forget to stop the
instance after the experiment because it is not available in the free tier.





Step 2

: After creating the instance click on Connect the instance and navigate to SSH Client.



Step 3: Now open the folder in the terminal where our .pem key is stored and paste the Example
command (starting with ssh -i .....) in the terminal.(ssh -i "Master_Ec2_Key.pem" ubuntu@ec2-54-196-129-215.compute-1.amazonaws.com)

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-172-31-95-62:~$ 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
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBFit2ioBEADhWpZ8/wvZ6hUTiXOwQHXMAlaFHcPH9hAtr4F1y2+OYdbtMuth
lqqwp028AqyY+PRfVMtSYMbjuQuu5byyKR01BbqYhuS3jtqQmljZ/bJvXqnmiVXh
38UuLa+z077PxyxQhu5BbqntTPQMfiyqEiU+BKbq2WmANUKQf+1AmZY/IruOXbnq
L4C1+gJ8vfmXQt99npCaxEjaNRVYfOS8QcixNzHUYnb6emjlANyEVlZzeqo7XKl7
UrwV5inawTSzWNvtjEjj4nJL8NsLwscpLPQUhTQ+7BbQXAwAmeHCUTQIvvWXqw0N
cmhh4HgeQscQHYgOJjjDVfoY5MucvglbIgCqfzAHW9jxmRL4qbMZj+b1XoePEtht
ku4bIQN1X5P07fNWzlgaRL5Z4POXDDZTlIQ/El58j9kp4bnWRCJW0lya+f8ocodo
vZZ+Doi+fy4D5ZGrL4XEcIQP/Lv5uFyf+kQtl/94VFYVJOleAv8W92KdgDkhTcTD
G7c0tIkVEKNUq48b3aQ64NOZQW7fVjfoKwEZdOqPE72Pa45jrZzvUFxSpdiNk2tZ
XYukHjlxxEgBdC/J3cMMNRE1F4NCA3ApfV1Y7/hTeOnmDuDYwr9/obA8t016Yljj
q5rdkywPf4JF8mXUW5eCN1vAFHxeg9ZWemhBtQmGxXnw9M+z6hWwc6ahmwARAQAB
tCtEb2NrZXIgUmVsZWFzZSAoQ0UgZGViKSA8ZG9ja2VyQGRvY2tlci5jb20+iQI3
BBMBCgAhBQJYrefAAhsvBQsJCAcDBRUKCQgLBRYCAwEAAh4BAheAAAoJEI2BgDwO
v82IsskP/iQZo68flDQmNvn8X5XTd6RRaUH33kXYXquT6NkHJciS7E2gTJmqvMqd
```

```
ubuntu@ip-172-31-95-62:~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable"
Repository: 'deb [arch=amd64] https://download.docker.com/linux/ubuntu noble stable'
Description:
Archive for codename: noble components: stable
More info: https://download.docker.com/linux/ubuntu
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
Adding deb entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
```

sudo apt-get update
sudo apt-get install -y docker-ce

```
ubuntu@ip-172-31-95-62:~$ sudo apt-get update
sudo apt-get install -y docker-ce
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 https://download.docker.com/linux/ubuntu noble InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
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 secti
 in apt-key(8) for details.
Reading package lists... Done
Building dependency tree... Done
```

```
Setting up docker-buildx-plugin (0.17.1-1~ubuntu.24.04~noble) ...
Setting up containerd.io (1.7.22-1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /usr/lib/systemd/system/containerd.service.
Setting up docker-compose-plugin (2.29.7-1~ubuntu.24.04~noble) ...
Setting up libltdl7:amd64 (2.4.7-7build1) ...
Setting up docker-ce-cli (5:27.3.1-1~ubuntu.24.04~noble) ...
Setting up libslirp0:amd64 (4.7.0-1ubuntu3) ...
Setting up pigz (2.8-1) ...
Setting up docker-ce-rootless-extras (5:27.3.1-1~ubuntu.24.04~noble) ...
Setting up slirp4netns (1.2.1-1build2) ...
Setting up docker-ce (5:27.3.1-1~ubuntu.24.04~noble) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.3) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-95-62:~$
```

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

```
  GNU nano 7.2                                                              /
{
  "exec-opts": ["native.cgroupdriver=systemd"]
}
```

sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart 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-172-31-95-62:~$ sudo systemctl daemon-reload
ubuntu@ip-172-31-95-62:~$ sudo systemctl restart docker
ubuntu@ip-172-31-95-62:~$
```

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-172-31-95-62:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gp
ubuntu@ip-172-31-95-62:~$ 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
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
ubuntu@ip-172-31-95-62:~$
```

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

```
ubuntu@ip-172-31-95-62:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu noble InRelease
Err:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb $(lsb_release InRelease
  403  Forbidden [IP: 108.138.64.44 443]
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION sectio
 in apt-key(8) for details.
E: Failed to fetch https://pkgs.k8s.io/core:/stable:/v1.31/deb/dists/$(lsb_release/InRelease  403  Forbidden [IP: 108.138.64.44 443]
E: The repository 'https://pkgs.k8s.io/core:/stable:/v1.31/deb $(lsb_release InRelease' is not signed.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
ubuntu@ip-172-31-95-62:~$
```

```
ubuntu@ip-172-31-81-58:~$ 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-172-31-81-58:~$ sudo systemctl daemon-reload
ubuntu@ip-172-31-81-58:~$ sudo systemctl restart docker
ubuntu@ip-172-31-81-58:~$
```

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-172-31-81-58:~$ sudo mkdir -p /etc/apt/keyrings/
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-
apt-keyring.gpg
ubuntu@ip-172-31-81-58:~$ 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
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
ubuntu@ip-172-31-81-58:~$
```

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

```
ubuntu@ip-172-31-81-58:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease
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-172-31-81-58:~$ sudo apt-get install -y kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
kubelet is already the newest version (1.31.1-1.1).
kubeadm is already the newest version (1.31.1-1.1).
kubectl is already the newest version (1.31.1-1.1).
0 upgraded, 0 newly installed, 0 to remove and 12 not upgraded.
ubuntu@ip-172-31-81-58:~$ sudo apt-mark hold kubelet kubeadm kubectl
kubelet was already set on hold.
kubeadm was already set on hold.
kubectl was already set on hold.
ubuntu@ip-172-31-81-58:~$
```

sudo systemctl enable --now kubelet
sudo kubeadm init --pod-network-cidr=10.244.0.0/16

```
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 172.31.81.58:6443 --token xo7et0.1j18rho6zn4kr5dw \
        --discovery-token-ca-cert-hash sha256:90ccf368d2fbd70e046a09b4f0a8be84de4b4e91760180fe716d788a13afd5c8
ubuntu@ip-172-31-81-58:~$
```

sudo apt-get install -y containerd

```
ubuntu@ip-172-31-81-58:~$ 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 12 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-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 runc amd64 1.1.12-0ubuntu3.1 [8599 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 containerd amd64 1.7.12-0ubuntu4.1 [38.6 M
B]
Fetched 47.2 MB in 1s (51.3 MB/s)
(Reading database ... 68202 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 ... 68182 files and directories currently installed.)
Preparing to unpack .../runc_1.1.12-0ubuntu3.1_amd64.deb ...
Unpacking runc (1.1.12-0ubuntu3.1) ...
```

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

```
ubuntu@ip-172-31-81-58:~$ 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 12 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-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 runc amd64 1.1.12-0ubuntu3.1 [8599 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 containerd amd64 1.7.12-0ubuntu4.1 [38.6 M
B]
Fetched 47.2 MB in 1s (51.3 MB/s)
(Reading database ... 68202 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 ... 68182 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 ...
```

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

```
ubuntu@ip-172-31-81-58:~$ sudo systemctl restart containerd
sudo systemctl enable containerd
sudo systemctl status containerd
● containerd.service - containerd container runtime
     Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: enabled)
     Active: active (running) since Sat 2024-10-12 17:41:02 UTC; 340ms ago
       Docs: https://containerd.io
   Main PID: 65436 (containerd)
      Tasks: 76
     Memory: 848.3M (peak: 882.1M)
        CPU: 192ms
     CGroup: /system.slice/containerd.service
             ├─59222 /usr/bin/containerd-shim-runc-v2 -namespace k8s.io -id 9215d752e3e0d11cebf7c06808e1da15801d4bd4e3e>
             ├─59254 /usr/bin/containerd-shim-runc-v2 -namespace k8s.io -id 68fe30e46f67f161694c173dce279a90659af718e5e>
             ├─59255 /usr/bin/containerd-shim-runc-v2 -namespace k8s.io -id d78e94e5b8cae6461d40ba39b9511b0f364cc8149c2>
             ├─59269 /usr/bin/containerd-shim-runc-v2 -namespace k8s.io -id a62b6834d6bf50383b11e7d9125f19a207e618f818c>
             ├─60502 /usr/bin/containerd-shim-runc-v2 -namespace k8s.io -id 6b11b124d1ef987cdf30ea3b420fdb856f3704e7182>
             └─65436 /usr/bin/containerd

Oct 12 17:41:02 ip-172-31-81-58 containerd[65436]: time="2024-10-12T17:41:02.705286131Z" level=info msg="Start subscrib>
Oct 12 17:41:02 ip-172-31-81-58 containerd[65436]: time="2024-10-12T17:41:02.705339862Z" level=info msg=serving... addr>
Oct 12 17:41:02 ip-172-31-81-58 containerd[65436]: time="2024-10-12T17:41:02.705485592Z" level=info msg=serving... addr>
Oct 12 17:41:02 ip-172-31-81-58 containerd[65436]: time="2024-10-12T17:41:02.705438787Z" level=info msg="Start recoveri>
Oct 12 17:41:02 ip-172-31-81-58 containerd[65436]: time="2024-10-12T17:41:02.754249626Z" level=info msg="Start event mo>
Oct 12 17:41:02 ip-172-31-81-58 containerd[65436]: time="2024-10-12T17:41:02.754288030Z" level=info msg="Start snapshot>
Oct 12 17:41:02 ip-172-31-81-58 containerd[65436]: time="2024-10-12T17:41:02.754298717Z" level=info msg="Start cni netw>
Oct 12 17:41:02 ip-172-31-81-58 containerd[65436]: time="2024-10-12T17:41:02.754305487Z" level=info msg="Start streamin>
Oct 12 17:41:02 ip-172-31-81-58 containerd[65436]: time="2024-10-12T17:41:02.754367810Z" level=info msg="containerd suc>
Oct 12 17:41:02 ip-172-31-81-58 systemd[1]: Started containerd.service - containerd container runtime.
lines 1-26/26 (END)
```

sudo apt-get install -y socat

```
ubuntu@ip-172-31-81-58:~$ sudo apt-get install -y socat
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
socat is already the newest version (1.8.0.0-4build3).
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.
0 upgraded, 0 newly installed, 0 to remove and 12 not upgraded.
ubuntu@ip-172-31-81-58:~$
```

Step 6: Initialize the Kubecluster
sudo kubeadm init --pod-network-cidr=10.244.0.0/16

```
ubuntu@ip-172-31-81-58:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
I1012 17:46:30.745964   75542 version.go:256] remote version is much newer: v1.31.1; falling back to: stable-1.27
[init] Using Kubernetes version: v1.27.16
[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 in beforehand using 'kubeadm config images pull'
W1012 17:46:31.007101   75542 images.go:80] could not find officially supported version of etcd for Kubernetes v1.27.16,
 falling back to the nearest etcd version (3.5.7-0)
W1012 17:46:31.089344   75542 checks.go:835] detected that the sandbox image "registry.k8s.io/pause:3.8" of the containe
r runtime is inconsistent with that used by kubeadm. It is recommended that using "registry.k8s.io/pause:3.9" as the CRI
 sandbox image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Using existing ca certificate authority
[certs] Using existing apiserver certificate and key on disk
[certs] Using existing apiserver-kubelet-client certificate and key on disk
[certs] Using existing front-proxy-ca certificate authority
[certs] Using existing front-proxy-client certificate and key on disk
[certs] Using existing etcd/ca certificate authority
[certs] Using existing etcd/server certificate and key on disk
[certs] Using existing etcd/peer certificate and key on disk
[certs] Using existing etcd/healthcheck-client certificate and key on disk
[certs] Using existing apiserver-etcd-client certificate and key on disk
[certs] Using the existing "sa" key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Using existing kubeconfig file: "/etc/kubernetes/admin.conf"
[kubeconfig] Using existing kubeconfig file: "/etc/kubernetes/kubelet.conf"
```

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

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-172-31-81-58:~$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-fl
annel.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
ubuntu@ip-172-31-81-58:~$
```

Step 7: Now that the cluster is up and running, we can deploy our nginx server on this cluster.Apply this deployment file using this command to create a deployment

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

```
ubuntu@ip-172-31-81-58:~$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
ubuntu@ip-172-31-81-58:~$
```

kubectl get pods

```
ubuntu@ip-172-31-81-58:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-cbdccf466-blrxb    0/1     Pending   0          24s
nginx-deployment-cbdccf466-nj5cm    0/1     Pending   0          24s
ubuntu@ip-172-31-81-58:~$
```

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

```
ubuntu@ip-172-31-81-58:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
ubuntu@ip-172-31-81-58:~$ kubectl port-forward $POD_NAME 8080:80
error: unable to forward port because pod is not running. Current status=Pending
ubuntu@ip-172-31-81-58:~$
```

kubectl taint nodes --all node-role.kubernetes.io/control-plane-node/ip-172-31-20-171 untainted
kubectl get nodes

```
ubuntu@ip-172-31-81-58:~$ kubectl taint nodes --all node-role.kubernetes.io/control-plane-
node/ip-172-31-81-58 untainted
ubuntu@ip-172-31-81-58:~$ kubectl get nodes
NAME              STATUS   ROLES           AGE    VERSION
ip-172-31-81-58   Ready    control-plane   6m4s   v1.31.1
ubuntu@ip-172-31-81-58:~$
```

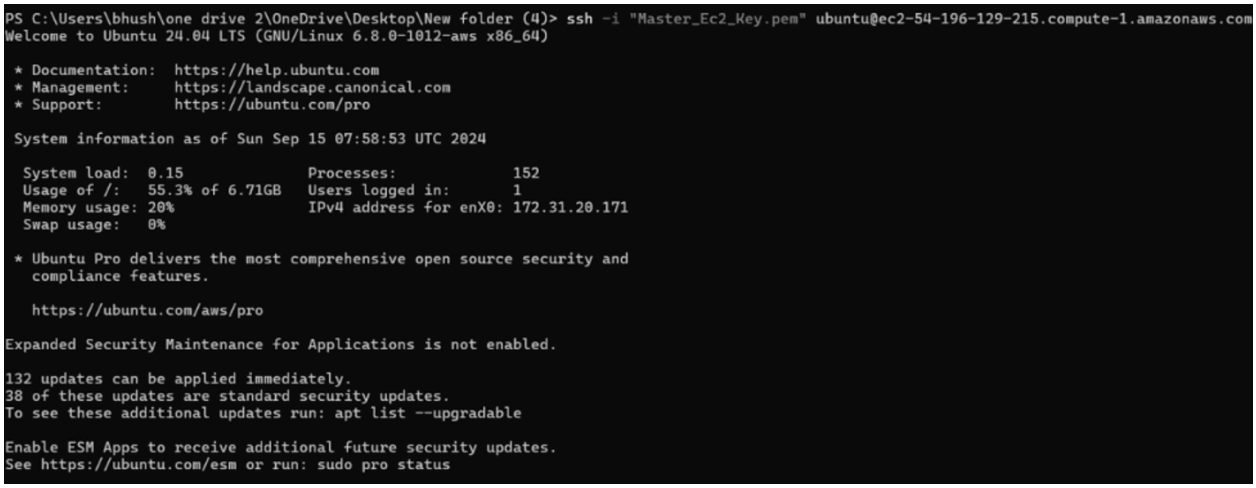
kubectl get pods

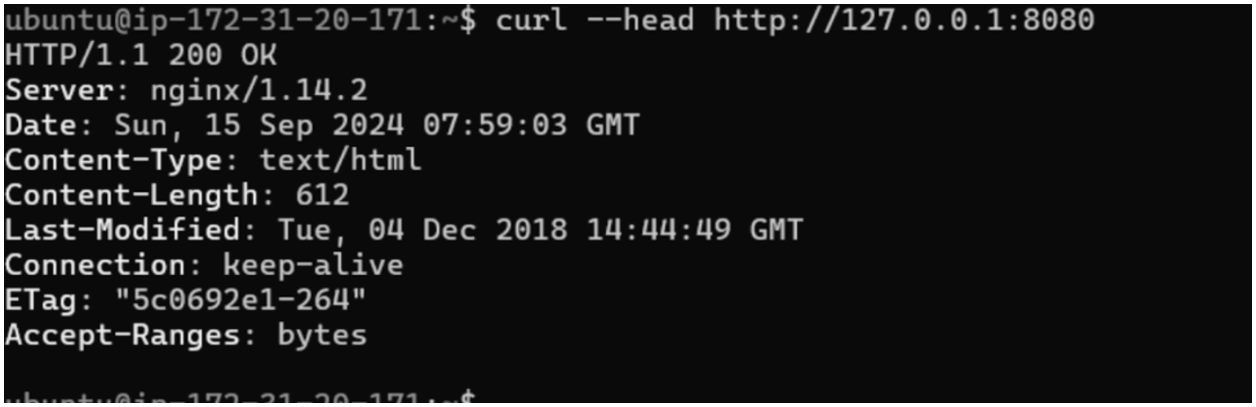
```
ubuntu@ip-172-31-81-58:~$ kubectl get pods
NAME                                READY   STATUS              RESTARTS   AGE
nginx-deployment-cbdccf466-blrxb    0/1     ContainerCreating   0          3m47s
nginx-deployment-cbdccf466-nj5cm    0/1     ContainerCreating   0          3m47s
ubuntu@ip-172-31-81-58:~$
```

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

```
ubuntu@ip-172-31-20-171:~$ 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





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:
In this experiment, we successfully installed Kubernetes on an EC2 instance and deployed an Nginx server using Kubectl commands. During the process, we encountered two main errors: the Kubernetes pod was initially in a pending state, which was resolved by removing the control-plane taint using kubectl taint nodes --all, and we also faced an issue with the missing containerd runtime, which was fixed by installing and starting containerd. We used a t2.medium EC2 instance with 2 CPUs to meet the necessary resource requirements for the Kubernetes setup and deployment.