Name: Krushikesh Sunil Shelar **2024-2025** **D15-C** **Roll No:51**

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.

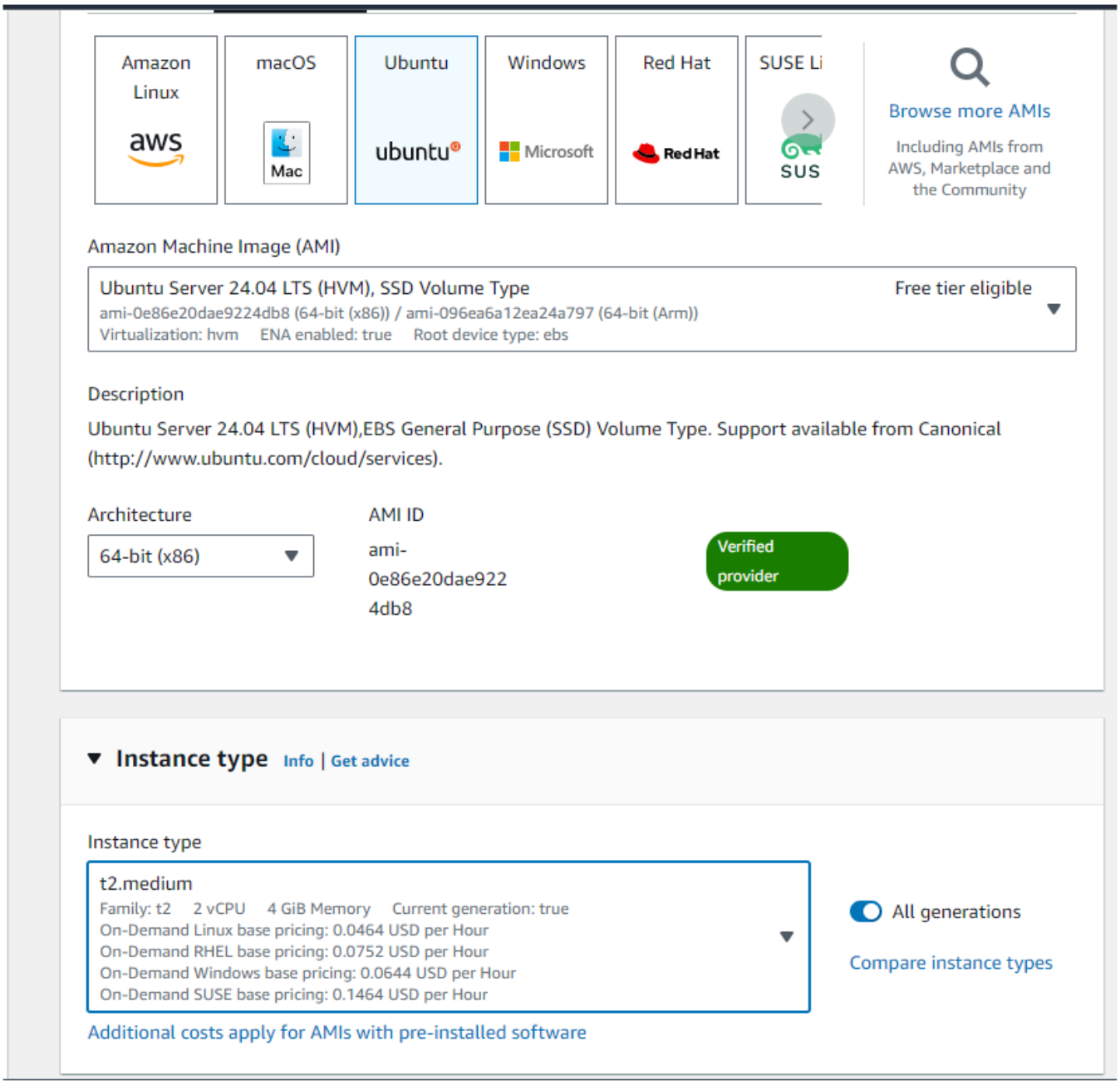
Kubernetes Deployment: Is a resource in Kubernetes that provides declarative updates for Pods and ReplicaSets. With a Deployment, you can define how many replicas of a pod should run, roll out new versions of an application, and roll back to previous versions if necessary. It ensures that the desired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

Necessary Requirements:

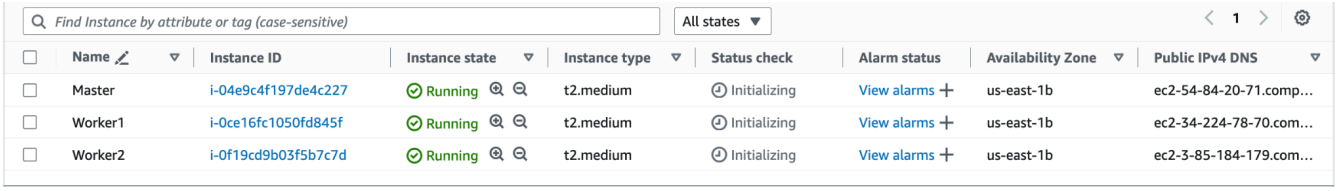
- **EC2 Instance:** The experiment required launching a t2.medium EC2 instance with 2 CPUs, as Kubernetes demands sufficient resources for effective functioning.
- **Minimum Requirements:**
    - **Instance Type:** t2.medium
    - **CPUs:** 2
    - **Memory:** Adequate for container orchestration.

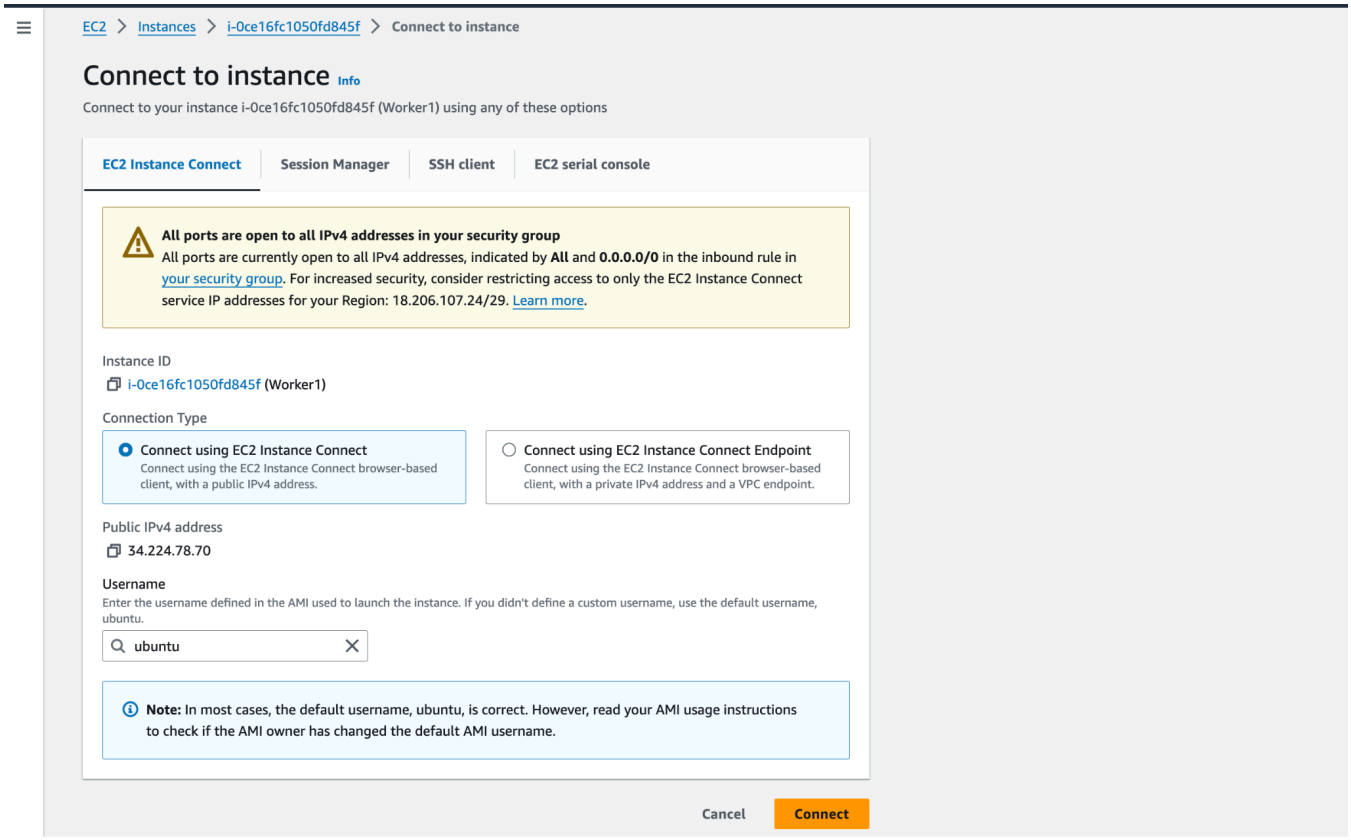
This ensured that the Kubernetes cluster had the necessary resources to function smoothly.

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 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.

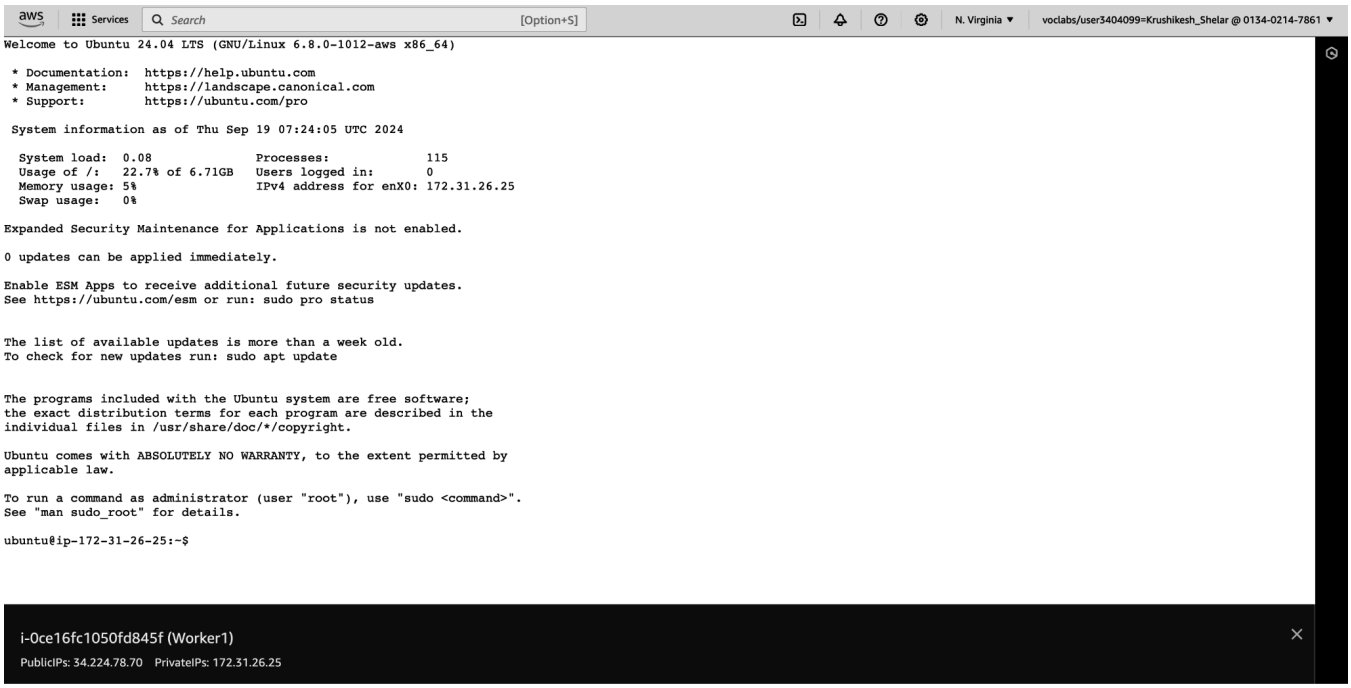


Step 2: After creating the instance click on Connect the instance and navigate to SSH Client.





Step 3: Successful Connection:



Step 4: Run on Master the below commands to install and setup Docker in Master.

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-172-31-17-92:~$ 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"
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
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]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:7 https://download.docker.com/linux/ubuntu noble/stable amd64 Packages [13.8 kB]
```

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

```
Setting up pigz (2.8-1) ...
Setting up docker-ce-rootless-extras (5:27.2.1-1~ubuntu.24.04~noble) ...
Setting up slirp4netns (1.2.1-1build2) ...
Setting up docker-ce (5:27.2.1-1~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 (2.39-0ubuntu8.2) ...
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-172-31-17-92:~$ docker --version
Docker version 27.2.1, build 9e34c9b
```

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

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

```
ubuntu@ip-172-31-17-92:~$ docker --version
Docker version 27.2.1, build 9e34c9b
ubuntu@ip-172-31-17-92:~$ sudo mkdir -p /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
    "exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
    "exec-opts": ["native.cgroupdriver=systemd"]
}
ubuntu@ip-172-31-17-92:~$ 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
```

Step 5: Run the below command to install Kubernetes.
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-172-31-20-171:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyri
ng.gpg
ubuntu@ip-172-31-20-171:~$ echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /et
c/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-172-31-27-176:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.3
1/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-k
eyring.gpg

echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://p
kgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/ku
bernetes.list

deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8
s.io/core:/stable:/v1.31/deb/ /
ubuntu@ip-172-31-27-176:~$
```

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

```
Setting up kubernetes-cni (1.5.1-1.1) ...
Setting up kubeadm (1.31.1-1.1) ...
Setting up kubelet (1.31.1-1.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
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.
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
ubuntu@ip-172-31-17-92:~$
```

sudo systemctl enable --now kubelet
sudo apt-get install -y containerd

```
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) ...
Setting up runc (1.1.12-0ubuntu3.1) ...
Setting up containerd (1.7.12-0ubuntu4.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
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-172-31-17-92:~$
```

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

```
ubuntu@ip-172-31-27-176:~$ sudo mkdir -p /etc/containerd
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
```

```
[timeouts]
  "io.containerd.timeout.bolt.open" = "0s"
  "io.containerd.timeout.metrics.shimstats" = "2s"
  "io.containerd.timeout.shim.cleanup" = "5s"
  "io.containerd.timeout.shim.load" = "5s"
  "io.containerd.timeout.shim.shutdown" = "3s"
  "io.containerd.timeout.task.state" = "2s"

[ttrpc]
  address = ""
  gid = 0
  uid = 0
```

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

```
ubuntu@ip-172-31-17-92:~$ 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 Thu 2024-09-19 07:47:29 UTC; 269ms ago
       Docs: https://containerd.io
   Main PID: 5079 (containerd)
      Tasks: 8
     Memory: 13.3M (peak: 14.0M)
        CPU: 69ms
     CGroup: /system.slice/containerd.service
             └─5079 /usr/bin/containerd

Sep 19 07:47:29 ip-172-31-17-92 containerd[5079]: time="2024-09-19T07:47:29.274330566Z" level=info msg="Start subscribing containerd event"
Sep 19 07:47:29 ip-172-31-17-92 containerd[5079]: time="2024-09-19T07:47:29.274370852Z" level=info msg="Start recovering state"
Sep 19 07:47:29 ip-172-31-17-92 containerd[5079]: time="2024-09-19T07:47:29.274417909Z" level=info msg="Start event monitor"
Sep 19 07:47:29 ip-172-31-17-92 containerd[5079]: time="2024-09-19T07:47:29.274428071Z" level=info msg=serving... address=/run/containerd/containerd.sock.ttrpc
Sep 19 07:47:29 ip-172-31-17-92 containerd[5079]: time="2024-09-19T07:47:29.274430045Z" level=info msg="Start snapshots syncer"
Sep 19 07:47:29 ip-172-31-17-92 containerd[5079]: time="2024-09-19T07:47:29.274458201Z" level=info msg="Start cni network conf syncer for default"
Sep 19 07:47:29 ip-172-31-17-92 containerd[5079]: time="2024-09-19T07:47:29.274461808Z" level=info msg=serving... address=/run/containerd/containerd.sock
Sep 19 07:47:29 ip-172-31-17-92 containerd[5079]: time="2024-09-19T07:47:29.274465481Z" level=info msg="Start streaming server"
Sep 19 07:47:29 ip-172-31-17-92 containerd[5079]: time="2024-09-19T07:47:29.274564095Z" level=info msg="containerd successfully booted in 0.028249s"
Sep 19 07:47:29 ip-172-31-17-92 systemd[1]: Started containerd.service - containerd container runtime.
```

sudo apt-get install -y socat

```
ubuntu@ip-172-31-17-92:~$  sudo apt-get install -y socat
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 NEW packages will be installed:
  socat
0 upgraded, 1 newly installed, 0 to remove and 133 not upgraded.
Need to get 374 kB of archives.
After this operation, 1649 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 socat amd64 1.8.0.0-4build3 [374 kB]
Fetched 374 kB in 0s (10.8 MB/s)
Selecting previously unselected package socat.
(Reading database ... 68108 files and directories currently installed.)
Preparing to unpack .../socat_1.8.0.0-4build3_amd64.deb ...
Unpacking socat (1.8.0.0-4build3) ...
Setting up socat (1.8.0.0-4build3) ...
Processing triggers for man-db (2.12.0-4build2) ...
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.
```

Step 6: Initialize the Kubecluster .Now Perform this Command only for Master.

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

Run this command on master and also copy and save the Join command from above.

```
[mark-control-plane] Marking the node ip-172-31-17-92 as control-plane by adding the taints [node-role.kubernetes.io/control-pla
[bootstrap-token] Using token: ikfl3a.g5zi9o3q6ac1cw2a
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to get nodes
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certific
[bootstrap-token] Configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap To
[bootstrap-token] Configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[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 172.31.17.92:6443 --token ikfl3a.g5zi9o3q6ac1cw2a \
        --discovery-token-ca-cert-hash sha256:43d27d0955ab782c8877f38d04e1146411c2c510ff75b82f8212bbf7f50db3dd
```

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

```
kubeadm join 172.31.17.92:6443 --token ikfl3a.g5zi9o3q6ac1cw2a \
        --discovery-token-ca-cert-hash sha256:43d27d0955ab782c8877f38d04e1146411c2c510ff75b82f8212bbf7f50db3dd
ubuntu@ip-172-31-17-92:~$ 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-172-31-17-92:~$ 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
```

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-172-31-17-92:~$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
```

kubectl get pods

```
ubuntu@ip-172-31-17-92:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-2s7ks    1/1     Running   0          24s
nginx-deployment-d556bf558-r8cpz    1/1     Running   0          24s
```

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

kubectl port-forward $POD_NAME 8080:80

```
ubuntu@ip-172-31-17-92:~$ kubectl port-forward $POD_NAME 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
```

```
^Cubuntu@ip-172-31-17-92:~kubectl taint nodes --all node-role.kubernetes.io/control-plane-node/ip-172-31-20-171 untainted

kubectl get nodes
error: at least one taint update is required
NAME              STATUS   ROLES           AGE   VERSION
ip-172-31-17-92   Ready    control-plane   41m   v1.31.1
ip-172-31-19-166  Ready    worker2         34m   v1.31.1
ip-172-31-26-25   Ready    worker1         35m   v1.31.1
```

kubectl get pods

```
ubuntu@ip-172-31-17-92:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-2s7ks    1/1     Running   0          4m2s
nginx-deployment-d556bf558-r8cpz    1/1     Running   0          4m2s
```

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

kubectl port-forward $POD_NAME 8080:80

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

```
Last login: Thu Sep 19 07:21:00 2024 from 18.206.107.27
ubuntu@ip-172-31-17-92:~$  curl --head http://127.0.0.1:8080
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Thu, 19 Sep 2024 08:35:44 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:
In this experiment, we successfully installed Kubernetes on an EC2 instance and deployed an Nginx server using Kubectl commands. We did not encounter any major errors during the process. We efficiently managed the setup by ensuring all components were correctly configured from the start. To meet the necessary resource requirements for the Kubernetes environment, we used a t2.medium EC2 instance with 2 CPUs, which facilitated smooth deployment and operation.