# Experiment 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.

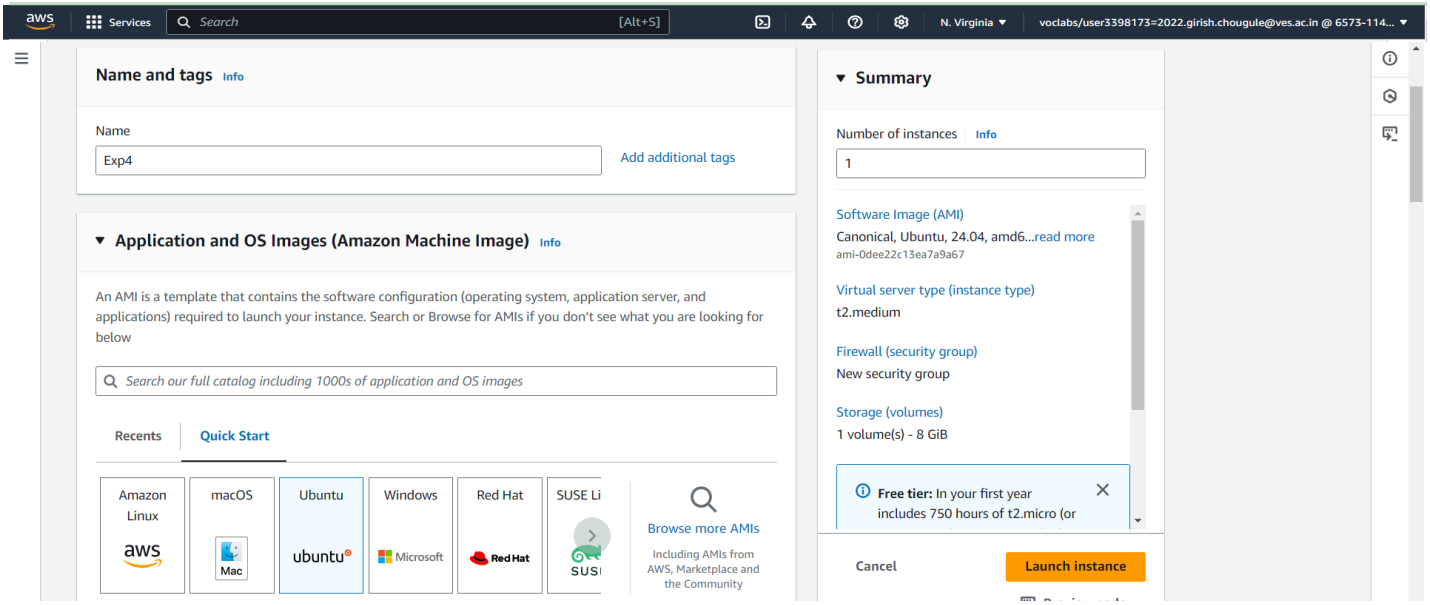
**Necessary Requirements:**

- **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.

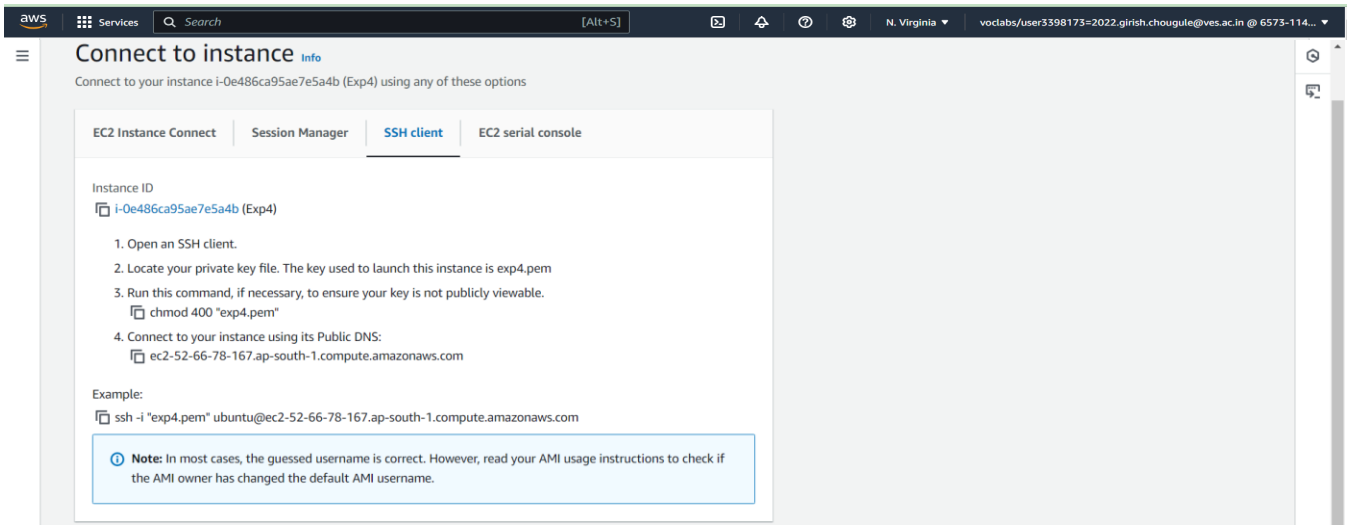
This ensured that the Kubernetes cluster had the necessary resources to function smoothly.

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.

```
C:\Users\khali\Downloads>ssh -i "exp4.pem" ubuntu@ec2-52-66-78-167.ap-south-1.compute.amazonaws.com
The authenticity of host 'ec2-52-66-78-167.ap-south-1.compute.amazonaws.com (52.66.78.167)' can't be established.
ECDSA key fingerprint is SHA256:SO1rxIlxzlT4+NqRCut1U9L5DECvWlsdsobsDqxe/vQ.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-52-66-78-167.ap-south-1.compute.amazonaws.com,52.66.78.167' (ECDSA) to the list of known
 hosts.
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 Sat Oct 12 12:31:48 UTC 2024

  System load:  0.3               Processes:             119
  Usage of /:   22.9% of 6.71GB   Users logged in:       0
  Memory usage: 6%                IPv4 address for enX0: 172.31.37.77
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

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

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

**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-172-31-37-77:~$ sudo snap install docker
docker 24.0.5 from Canonical✓ installed
ubuntu@ip-172-31-37-77:~$ docker --version
Docker version 24.0.5, build ced0996
```

```
ubuntu@ip-172-31-37-77:~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_relea
se -cs) stable"
sudo apt-get update
sudo apt-get install -y docker-ce
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
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.Found existing deb entry in /etc/apt/sources.list.d/docker.list
Updating existing entry instead of using /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-nobl
e.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/docker.list
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Get:4 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
```

**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-172-31-37-77:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /et
c/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
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
sudo systemctl enable --now kubelet
ubuntu@ip-172-31-37-77:~$ echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/s
table:/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/ /
```

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

```
ubuntu@ip-172-31-37-77:~$ sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
sudo systemctl enable --now kubelet
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Get:4 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Err:4 https://download.docker.com/linux/ubuntu noble InRelease
  The following signatures couldn't be verified because the public key is not available: NO_PUBKEY 7EA0A9C3F273FCD8
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease [1186 B]
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  Packages [4865 B]
Reading package lists... Done
W: GPG error: https://download.docker.com/linux/ubuntu noble InRelease: The following signatures couldn't be verified be
cause the public key is not available: NO_PUBKEY 7EA0A9C3F273FCD8
E: The repository 'https://download.docker.com/linux/ubuntu noble 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.
```

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

```
ubuntu@ip-172-31-37-77:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
sudo: kubeadm: command not found
```

**Now We have got an error.**
**So we have to perform some additional commands as follow.**

**sudo apt-get install -y containerd**
**sudo mkdir -p /etc/containerd**
**sudo containerd config default | sudo tee /etc/containerd/config.toml**
**sudo systemctl restart containerd**
**sudo systemctl enable containerd**
**sudo systemctl status containerd**

**sudo apt-get install -y socat**

```
ubuntu@ip-172-31-37-77:~$ sudo apt-get install -y containerd
mkdir -p /etc/containerdsudo mkdir -p /etc/containerd
tee /etc/containerd/config.toml| sudo
sudo systemctl restart containerd
sudo systemctl enable containerd
sudo systemctl status containerd
Reading package lists... Done
 install -y socat
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  runc
The following NEW packages will be installed:
  containerd runc
0 upgraded, 2 newly installed, 0 to remove and 12 not upgraded.
Need to get 47.2 MB of archives.
After this operation, 179 MB of additional disk space will be used.
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 runc amd64 1.1.12-0ubuntu3.1 [8599 kB]
Get:2 http://ap-south-1.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 (60.9 MB/s)
Selecting previously unselected package runc.
(Reading database ... 67836 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) ...
Setting up runc (1.1.12-0ubuntu3.1) ...
Setting up containerd (1.7.12-0ubuntu4.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /usr/lib/systemd/system/containerd.service.
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
**sudo kubeadm init --pod-network-cidr=10.244.0.0/16**

```
ubuntu@ip-172-31-37-77:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.1
[preflight] Running pre-flight checks
        [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'
W1012 12:59:46.685154    6007 checks.go:846] 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 to use "registry.k8s.io/pause:3.10" as the CRI san
dbox 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-172-31-37-77 kubernetes kubernetes.default kubernetes.default
.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 172.31.37.77]
[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-172-31-37-77 localhost] and IPs [172.31.37.77 127.0.0.1 ::1
]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-37-77 localhost] and IPs [172.31.37.77 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
```

```
[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 172.31.37.77:6443 --token qa8xqc.zffmf5d8x9urjmqd \
        --discovery-token-ca-cert-hash sha256:ea1a050f59e0bb9d92a5b36f4ff507c5b281c10a5f3937bbefc2f154232c86cd
```

**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**

```
ubuntu@ip-172-31-37-77:~$ mkdir -p $HOME/.kube
p -i /etubuntu@ip-172-31-37-77:~$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $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-172-31-37-77:~$ kubectl apply -f kube-flannel-legacy.yml
serviceaccount/flannel unchanged
configmap/kube-flannel-cfg configured
Warning: spec.template.spec.nodeSelector[beta.kubernetes.io/arch]: deprecated since v1.14; use "kubernetes.io/arch" ins
tead
daemonset.apps/kube-flannel-ds created
```

```
ubuntu@ip-172-31-37-77:~$ kubectl get daemonsets -n kube-system
NAME              DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR                   AGE
kube-flannel-ds   0         0         0       0            0           beta.kubernetes.io/arch=amd64   2m46s
kube-proxy        1         1         0       1            0           kubernetes.io/os=linux          23m
```

**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 get pods**

```
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-2nwj8    0/1     Pending   0          41m
nginx-deployment-d556bf558-vbnn6    0/1     Pending   0          41m
```

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

**kubectl port-forward $POD_NAME 8080:80**

```
ubuntu@ip-172-31-87-78:~$ kubectl port-forward $POD_NAME 8080:80
error: unable to forward port because pod is not running. Current status=Pending
```

**Note : We have faced an error as pod status is pending so make it running run below commands then again run above 2 commands.**

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

**kubectl get nodes**

```
ubuntu@ip-172-31-87-78:~$ kubectl taint nodes --all node-role.kubernetes.io/control-plane-node/ip-172-31-20-171 untainted
error: at least one taint update is required
ubuntu@ip-172-31-87-78:~$ kubectl get nodes
NAME              STATUS   ROLES           AGE   VERSION
ip-172-31-87-78   Ready    control-plane   63m   v1.31.1
```

**kubectl get pods**

```
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-2nwj8    0/1     Pending   0          41m
nginx-deployment-d556bf558-vbnn6    0/1     Pending   0          41m
```

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

**kubectl port-forward $POD_NAME 8080:80**

```
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-vz8rv    1/1     Running   0          3m4s
nginx-deployment-d556bf558-wz5wc    1/1     Running   0          3m4s
```

**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**

```
ubuntu@ip-172-31-20-171:~$ curl --head http://127.0.0.1:8080
HTTP/1.1 200 OK
```

**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.