

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.

Steps:

Log in to your AWS Academy/personal account.

1. Create security group

create security group with following configuration lets name it as exp4.

EC2 > Security Groups > Create security group

Create security group info

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

Basic details

Security group name info
exp4SecurityGroup
Name cannot be edited after creation.

Description info
security for exp4

VPC info
vpc-07b6966cbfa88ee3

Inbound rules info

Type <small>info</small>	Protocol <small>info</small>	Port range <small>info</small>	Source <small>info</small>	Description - optional <small>info</small>	
SSH	TCP	22	Anywhere-I...	0.0.0.0/0	Delete
Custom TCP	TCP	6443	Anywhere-I...	0.0.0.0/0	Delete
All traffic	All	All	Anywhere-I...	0.0.0.0/0	Delete

Add rule

click on create security group.

RWS Services [Alt+S]

EC2 Dashboard
EC2 Global View
Events
Console-to-Code [Preview](#)

▼ Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations

▼ Images
AMIs
AMI Catalog

▼ Elastic Block Store
Volumes
Snapshots
Lifecycle Manager

▼ Network & Security
Security Groups

Security group (sg-0f87a692e5cdada58 | exp4SecurityGroup) was created successfully

Details

EC2 > Security Groups > sg-0f87a692e5cdada58 - exp4SecurityGroup

sg-0f87a692e5cdada58 - exp4SecurityGroup

Actions

Details

Security group name exp4SecurityGroup	Security group ID sg-0f87a692e5cdada58	Description security for exp4	VPC ID vpc-07b6966cbfa88ee3
Owner 209322483715	Inbound rules count 3 Permission entries	Outbound rules count 1 Permission entry	

Inbound rules | Outbound rules | Tags

Inbound rules (3)

Search

	Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
<input type="checkbox"/>	-	sg-0493fce359abad547	IPv4	SSH	TCP	22	0.0.0.0/0	-
<input type="checkbox"/>	-	exp-0f87a692e5cdada58	IPv4	All traffic	All	All	0.0.0.0/0	-

2. Create Instance

Launch an ec2 instance.

☰

Name and tags [Info](#)

Name

exp4

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

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

Recents

Quick Start

Amazon Linux

aws

macOS

Mac

Ubuntu

ubuntu

Windows

Microsoft

Red Hat

Red Hat

SUSE Linux

SUSE

🔍

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type

ami-0e86e20dae9224db8 (64-bit (x86)) / ami-096ea6a12ea24a797 (64-bit (Arm))

Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible ▼

Description

Ubuntu Server 24.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Architecture

AMI ID

Username

Verified provider

64-bit (x86) ▼

ami-0e86e20dae9224db8

ubuntu

Verified provider

Select Ubuntu 22.04 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.

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type
ami-0e86e20dae9224db8 (64-bit (x86)) / ami-096ea6a12ea24a797 (64-bit (Arm))
Virtualization: hvm FFA enabled: true Root device type: ebs

Free tier eligible

Q

t2.small
Family: t2 1 vCPU 2 GiB Memory Current generation: true
On-Demand Windows base pricing: 0.032 USD per Hour
On-Demand Linux base pricing: 0.023 USD per Hour
On-Demand RHEL base pricing: 0.0376 USD per Hour
On-Demand SUSE base pricing: 0.053 USD per Hour

t2.medium
Family: t2 2 vCPU 4 GiB Memory Current generation: true
On-Demand Linux base pricing: 0.0464 USD per Hour
On-Demand RHEL base pricing: 0.0752 USD per Hour
On-Demand Windows base pricing: 0.0644 USD per Hour
On-Demand SUSE base pricing: 0.1464 USD per Hour

t2.large
Family: t2 2 vCPU 8 GiB Memory Current generation: true
On-Demand Windows base pricing: 0.1208 USD per Hour
On-Demand RHEL base pricing: 0.1216 USD per Hour
On-Demand SUSE base pricing: 0.1928 USD per Hour
On-Demand Linux base pricing: 0.0928 USD per Hour

t2.xlarge
Family: t2 4 vCPU 16 GiB Memory Current generation: true

t2.micro
Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Windows base pricing: 0.0162 USD per Hour
On-Demand SUSE base pricing: 0.0116 USD per Hour
On-Demand RHEL base pricing: 0.026 USD per Hour
On-Demand Linux base pricing: 0.0116 USD per Hour

Free tier eligible

Available from Canonical

1

Softw
Cano
ami-0

Virtu
t2.mi

Firew
New

Stora
1 vol

All generations

Compare instance types

Additional costs apply for AMIs with pre-installed software

▼ Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the 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.


Create a key-pair to login to the machine remotely and then select this newly generated key-pair.

▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

exp4 ▼

 [Create new key pair](#)

▼ Network settings [Info](#)

Edit

Network [Info](#)

vpc-07b6966cbfba88ee3

Subnet [Info](#)

No preference (Default subnet in any availability zone)

Select existing security group and select the security group we created at the start.

▼ Network settings

Info

Edit

Network

Info

vpc-07b6966cbfba88ee3

Subnet

Info

No preference (Default subnet in any availability zone)

Auto-assign public IP

Info

Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups)

Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

○ Create security group

● Select existing security group

Common security groups

Info

Select security groups

exp4SecurityGroup sg-0f87a692e5cdada58

×

VPC: vpc-07b6966cbfba88ee3

Compare security group rules

Security groups that you add or remove here will be added to or removed from all your network interfaces.

▼ Configure storage

Info

Advanced

Launch the instance.

3. Connect to the instance.

Select the instance created

click on Connect the instance and navigate to SSH Client.

Instances (1/1)

Info

Last updated less than a minute ago

Refresh

Connect

Inst

Find Instance by attribute or tag (case-sensitive)

All states

Instance ID = i-09426999c36422602

×

Clear filters

✓	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
✓	exp4	i-09426999c36422602	Running	t2.medium	Initializing	View alarms	us-east-1b

Copy the command that comes to your dashboard at the bottom.

EC2 > Instances > i-09426999c36422602 > Connect to instance

Connect to instance [Info](#)

Connect to your instance i-09426999c36422602 (exp4) using any of these options

EC2 Instance Connect

Session Manager

SSH client

EC2 serial console

Instance ID

i-09426999c36422602 (exp4)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is exp4.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
 `chmod 400 "exp4.pem"`
4. Connect to your instance using its Public DNS:
 `ec2-44-212-57-152.compute-1.amazonaws.com`

Command copied

`ssh -i "exp4.pem" ubuntu@ec2-44-212-57-152.compute-1.amazonaws.com`

Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

Cancel

Now copy the path to the file where our .pem key is stored and replace the pem file in the command copied from the ssh dashboard.

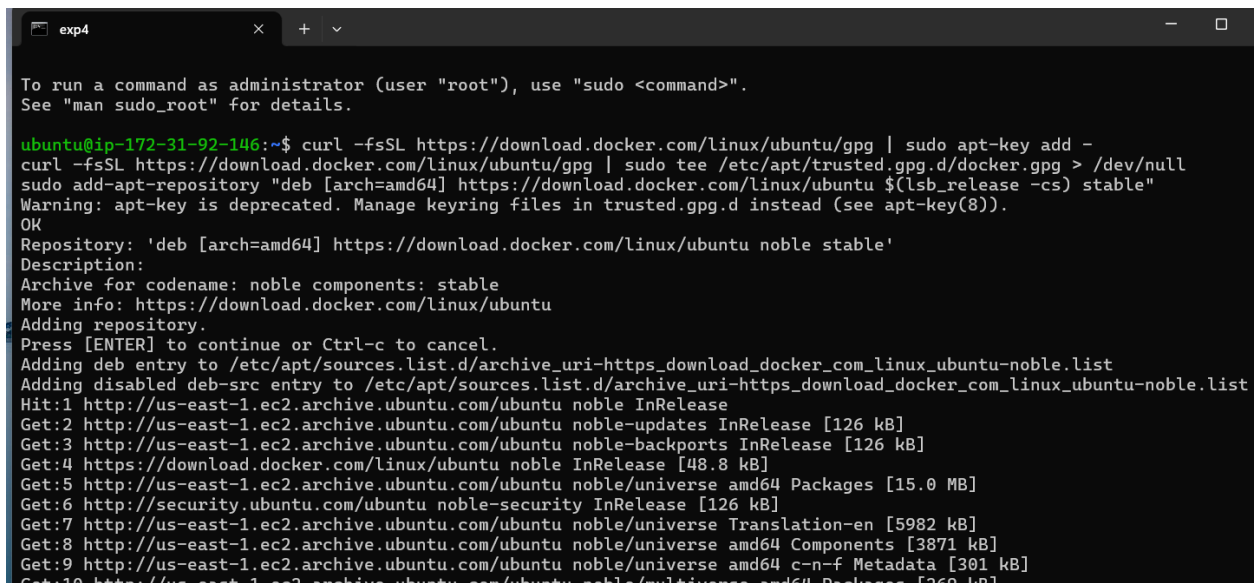
```
C:\Users\Lenovo>ssh -i "C:\Users\Lenovo\Downloads\exp4.pem" ubuntu@ec2-44-212-57-152.compute-1.amazonaws.com
The authenticity of host 'ec2-44-212-57-152.compute-1.amazonaws.com (44.212.57.152)' can't be established.
ED25519 key fingerprint is SHA256:SYWntsQatiMJ2x6vE4Nabz7KWXCSDPgjer2N22WJ7eU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes|
```

4. Install and set-up Docker

Run the following commands:

1. We have to install and setup Docker. Run these commands

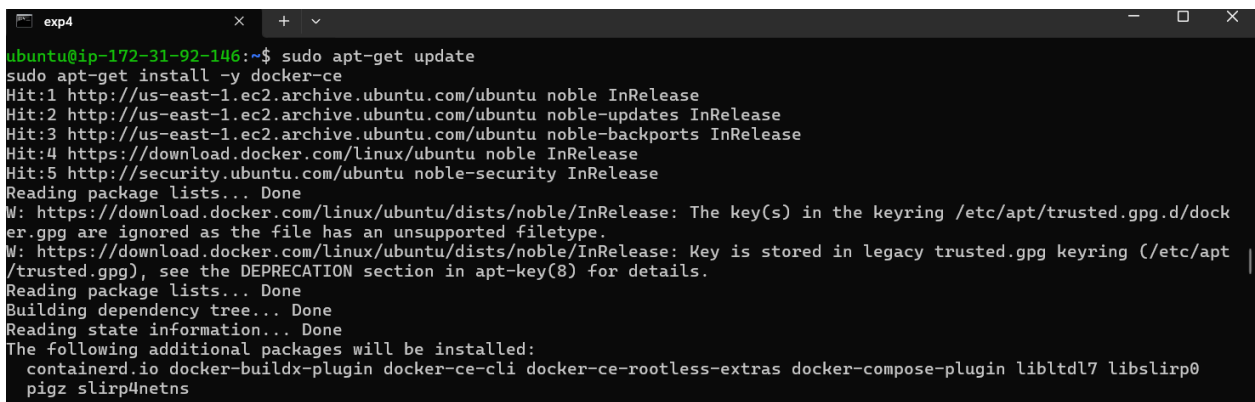
```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -  
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee  
/etc/apt/trusted.gpg.d/docker.gpg > /dev/null  
sudo add-apt-repository "deb [arch=amd64]  
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```



```
exp4  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
  
ubuntu@ip-172-31-92-146:~$ 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 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]  
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]  
Get:6 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]  
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]  
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]  
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]  
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
```

2. Update

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



```
exp4  
ubuntu@ip-172-31-92-146:~$ 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 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  
Reading package lists... Done  
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: The key(s) in the keyring /etc/apt/trusted.gpg.d/docker.gpg are ignored as the file has an unsupported filetype.  
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.  
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 libslirp0  
  pigz slirp4netns
```



```
exp4
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 (2.39-0ubuntu8.2) ...
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-172-31-92-146:~$
```

3. Configure Docker to use the `systemd` cgroup driver by creating the necessary configuration file in the `/etc/docker` directory.

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

```
exp4
ubuntu@ip-172-31-92-146:~$ 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-172-31-92-146:~$
```

4. Restart and enable docker:

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

```
ubuntu@ip-172-31-92-146:~$ 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-172-31-92-146:~$
```

5. Set-up Kubernetes

1. Add the Kubernetes signing key and repository to your APT sources for package installation.

```
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
```

```
exp4
ubuntu@ip-172-31-92-146:~$ 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
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
ubuntu@ip-172-31-92-146:~$
```

2. Update APT package lists, install Kubernetes tools ([kubelet](#), [kubeadm](#), [kubctl](#)), and mark them to prevent automatic updates.

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

```
exp4
ubuntu@ip-172-31-92-146:~$ sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
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
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]
Fetched 6051 B in 0s (12.3 kB/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: The key(s) in the keyring /etc/apt/trusted.gpg.d/docker.gpg are ignored as the file has an unsupported filetype.
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.
Reading package lists... Done
Building dependency tree... Done
```

```
exp4
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-172-31-92-146:~$ |
```

6. Initialize the kubecuster

1. Enable and start the `kubelet` service, then initialize the Kubernetes cluster with a specified pod network CIDR.

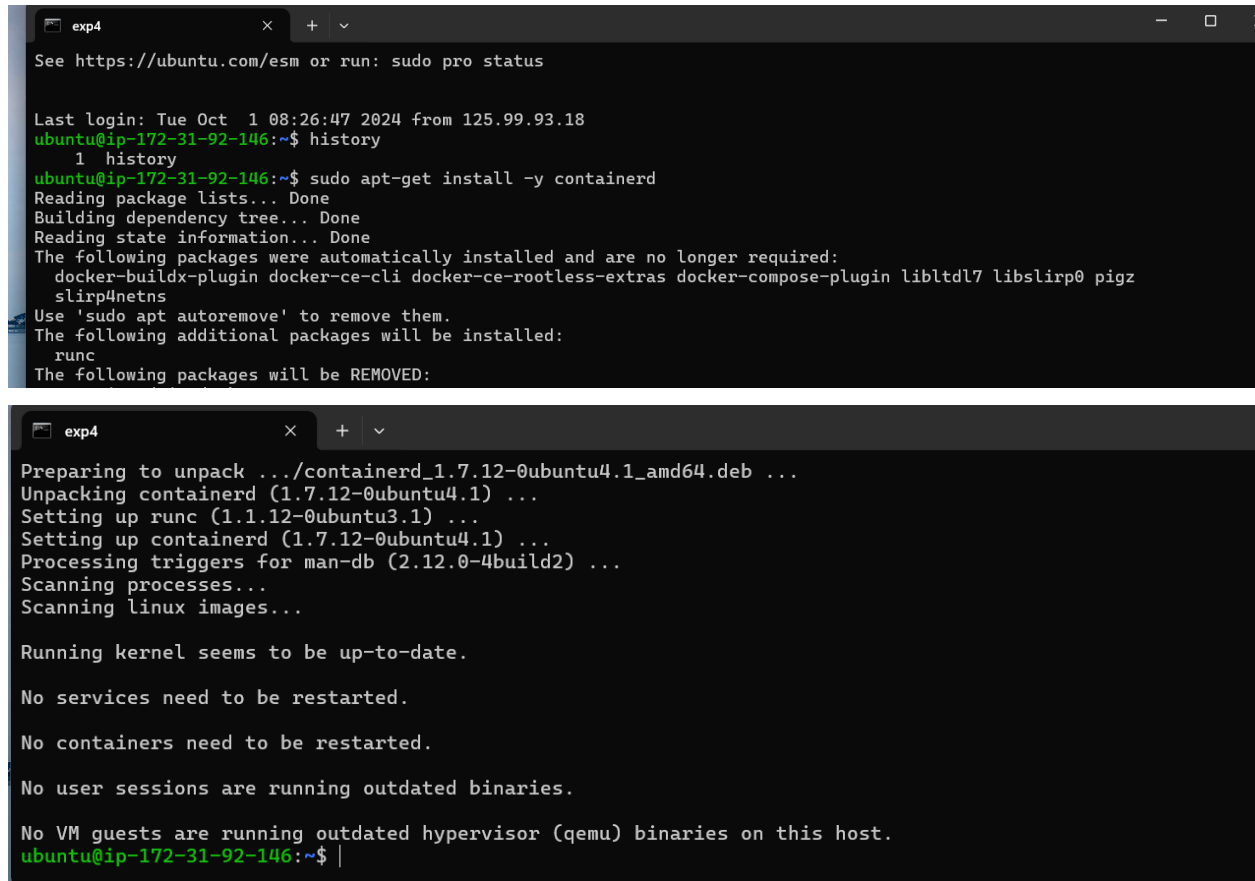
```
sudo systemctl enable --now kubelet
```

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

```
exp4
ubuntu@ip-172-31-92-146:~$ sudo systemctl enable --now kubelet
sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
W1001 08:58:04.356900 4249 checks.go:1080] [preflight] WARNING: Couldn't create the interface used for talking to the
container runtime: failed to create new CRI runtime service: validate service connection: validate CRI v1 runtime API f
or endpoint "unix:///var/run/containerd/containerd.sock": rpc error: code = Unimplemented desc = 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.v1.RuntimeService[p
reflight] If you know what you are doing, you can make a check non-fatal with '--ignore-preflight-errors=...'
To see the stack trace of this error execute with --v=5 or higher
ubuntu@ip-172-31-92-146:~$ |
```

Here, we encounter an error as a few of the dependencies for running the command are not installed. So, run the following commands

```
sudo apt-get install -y containerd
```



The image shows two terminal windows. The top window shows the command `sudo apt-get install -y containerd` being executed, which lists several packages to be removed and installed. The bottom window shows the output of `sudo systemctl status containerd`, indicating that the service is active and running.

```
exp4
See https://ubuntu.com/esm or run: sudo pro status

Last login: Tue Oct  1 08:26:47 2024 from 125.99.93.18
ubuntu@ip-172-31-92-146:~$ history
1  history
ubuntu@ip-172-31-92-146:~$ 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:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz
  slirp4netns

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-172-31-92-146:~$
```

2. Create the `/etc/containerd` directory and generate the default `containerd` configuration file (`config.toml`).

```
sudo mkdir -p /etc/containerd
```

```
sudo containerd config default | sudo tee /etc/containerd/config.toml
```

```
exp4
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-92-146:~$ 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 = ""

[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
ubuntu@ip-172-31-92-146:~$
```

3. Restart, enable, and check the status of the `containerd` service.

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

```
exp4
ubuntu@ip-172-31-92-146:~$ 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 Tue 2024-10-01 09:18:05 UTC; 274ms ago
     Docs: https://containerd.io
   Main PID: 5581 (containerd)
    Tasks: 7
   Memory: 13.6M (peak: 13.9M)
      CPU: 71ms
   CGroup: /system.slice/containerd.service
           └─5581 /usr/bin/containerd

Oct 01 09:18:05 ip-172-31-92-146 containerd[5581]: time="2024-10-01T09:18:05.013214307Z" level=info msg=serving... addr=
Oct 01 09:18:05 ip-172-31-92-146 containerd[5581]: time="2024-10-01T09:18:05.013246107Z" level=info msg=serving... addr=
Oct 01 09:18:05 ip-172-31-92-146 containerd[5581]: time="2024-10-01T09:18:05.013300897Z" level=info msg="Start subscrib
Oct 01 09:18:05 ip-172-31-92-146 containerd[5581]: time="2024-10-01T09:18:05.013325717Z" level=info msg="Start recoveri>
```

4. Install the **socat** package using APT with no prompts.

```
sudo apt-get install -y socat
```

```
exp4
ubuntu@ip-172-31-92-146:~$ 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 143 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 (9246 kB/s)
Selecting previously unselected package socat.
dpkg: warning: files list only for packages generated by the --build flag.
```

5. Initialize the Kubernetes cluster with a specified pod network CIDR of **10.244.0.0/16**.

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

```
exp4
ubuntu@ip-172-31-92-146:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.0
[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'
W1001 09:21:47.290415 5902 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-172-31-92-146 kubernetess kubernetess.default kubernetess.default.svc kubernetess.default.svc.cluster.local] and IPs [10.96.0.1 172.31.92.146]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
```

```
exp4
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 172.31.92.146:6443 --token vg6cqy.kx64j8i2bp7qf776 \
--discovery-token-ca-cert-hash sha256:812f3da588c8ecd9e96cf40a0ea5d99360e518299e5ec7b026f8e228c2017904
ubuntu@ip-172-31-92-146:~$
```

6. Deploy the Flannel network add-on for Kubernetes by applying the specified YAML configuration file from the provided URL.

kubectl apply -f

<https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml>

7. Connect nginx server to pod.

```
ip-172-31-81-27.ec2.internal notready control-plane 15m v1.31.1
[ec2-user@ip-172-31-81-27 bin]$ kubectl get nodes
NAME                                STATUS    ROLES    AGE    VERSION
ip-172-31-81-27.ec2.internal        Ready     control-plane 15m    v1.31.1
[ec2-user@ip-172-31-81-27 bin]$

[ec2-user@ip-172-31-81-27 bin]$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
[ec2-user@ip-172-31-81-27 bin]$

[ec2-user@ip-172-31-81-27 bin]$ kubectl get pods
NAME                                READY    STATUS    RESTARTS    AGE
nginx-deployment-d556bf558-4prm9    0/1      Pending   0            94s
nginx-deployment-d556bf558-d6dlb    0/1      Pending   0            94s
[ec2-user@ip-172-31-81-27 bin]$
```

Conclusion:

In this experiment, we successfully deployed an Nginx server to a Kubernetes cluster and addressed common challenges such as node taints that can hinder the deployment process. Additionally, we configured the server to run on the desired port, ensuring proper traffic routing. This exercise not only enhanced our understanding of Kubernetes deployment techniques but also provided valuable insights into managing cluster configurations for smooth and efficient operations. Overall, it has strengthened our skills in setting up and maintaining applications in containerized environments.