

Case Study: Topic No 2

Topic Name: Kubernetes Application Deployment

Main Problem Statement:

- Concepts Used: Kubernetes, AWS Cloud9 IDE, and Kubectl.
- Problem Statement: "Set up a Kubernetes cluster on AWS using the Cloud9 IDE.

Deploy a sample application using kubectl and ensure it runs successfully."

- Tasks:

Install and configure kubectl using AWS Cloud9 IDE.

Deploy a sample application (like a simple Nginx server) on the Kubernetes clusters.

Verify the application deployment by accessing it through a NodePort or LoadBalancer.

1. Introduction

Case Study Overview:

A mid-sized e-commerce organization faced challenges in managing its applications, including slow deployment cycles, difficulty scaling, and inconsistent environments across development and production. To address these issues, the organization decided to adopt Kubernetes as its container orchestration platform.

Key Features

Kubernetes provides automated container orchestration, enabling efficient deployment and management of applications across clusters. Its self-healing capability ensures that failed containers are automatically restarted or replaced. The platform includes service discovery for seamless communication between microservices, along with load balancing to distribute traffic effectively. Rolling updates allow for zero-downtime deployments, facilitating smooth transitions between application versions. Additionally, Kubernetes supports horizontal scaling, automatically adjusting the number of active containers based on demand.

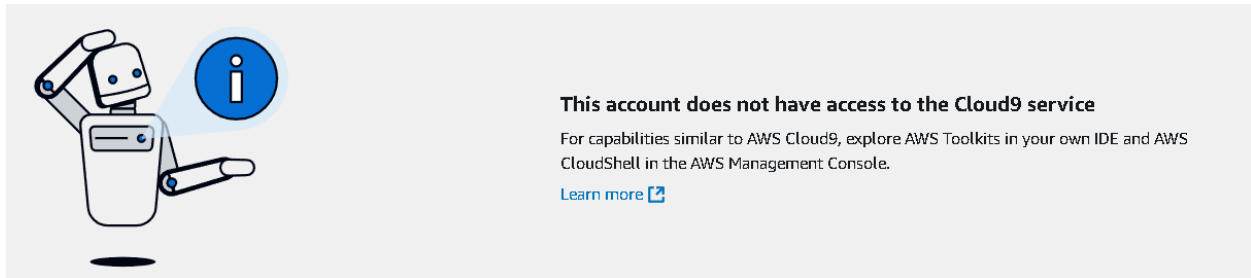
Applications

Kubernetes is widely used for **microservices architectures**, allowing teams to develop, deploy, and manage services independently. It is ideal for **cloud-native applications**, leveraging its scalability and resilience in dynamic environments. Organizations also use Kubernetes for **continuous integration and continuous deployment (CI/CD)**

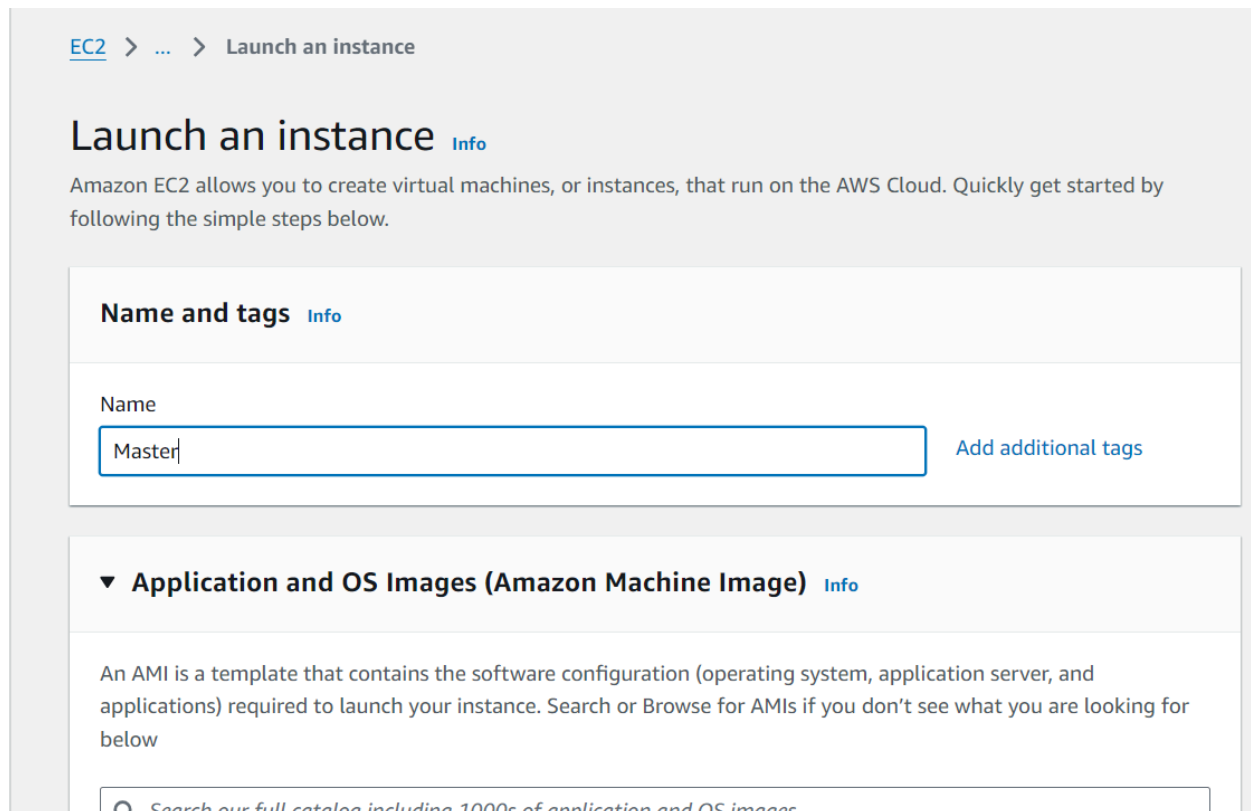
pipelines, streamlining the software development lifecycle. It supports **big data processing**, enabling efficient management of data-intensive applications. Additionally, Kubernetes is increasingly adopted for **multi-cloud strategies**, allowing businesses to deploy applications across various cloud providers seamlessly.

Note**

AWS **Cloud9** has been **discontinued**, so we will now use **EC2** for our development environment.

**Steps:****Step 1: Create an EC2 Ubuntu Instance on AWS**

Launch an Ubuntu instance on AWS EC2 to serve as the host for Kubernetes and Docker. Choose the appropriate instance type and configure network settings as needed.

A screenshot of the AWS Management Console 'Launch an instance' page. The breadcrumb trail at the top is 'EC2 > ... > Launch an instance'. The main heading is 'Launch an instance' with an 'Info' link. Below the heading is a descriptive paragraph: 'Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.' The first section is 'Name and tags' with an 'Info' link. It contains a 'Name' label and a text input field with the value 'Master'. To the right of the input field is a link 'Add additional tags'. The second section is 'Application and OS Images (Amazon Machine Image)' with a dropdown arrow and an 'Info' link. It contains a paragraph: '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'. At the bottom is a search bar with a magnifying glass icon and the placeholder text 'Search our full catalog including 1000s of application and OS images'.

▼ 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

Recents

Quick Start

Amazon Linux

aws

macOS

Mac

Ubuntu

ubuntu

Windows

Microsoft

Red Hat

Red Hat

SUSE Li

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-0866a3c8686eae6ba (64-bit (x86)) / ami-0325498274077fac5 (64-bit (Arm))

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

Free tier eligible

▼ Summary

Number of instances

1

Software Image (AMI)

Amazon Linux 2023 AMI 2023.6.2...[read more](#)

ami-06b21ccea88cd686

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 8 GiB

Cancel

Launch instance

Preview code

Step 2: Edit the Security Group Inbound Rules to Allow SSH

Modify the security group settings to allow SSH access (port 22) from your IP address. This enables secure remote access to the instance.

Inbound rules [Info](#)

Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info	
SSH	TCP	22	Custom <input type="text"/> 0.0.0.0/0		Delete
HTTP	TCP	80	Custom <input type="text"/> 0.0.0.0/0		Delete
HTTPS	TCP	443	Custom <input type="text"/> 0.0.0.0/0		Delete
Add rule					

Step 3:SSH into the machine `ssh -i <keyname>.pem ubuntu@<public_ip_address>`

Use this command to connect to your EC2 instance via SSH, replacing <keyname> with your private key file and <public_ip_address> with your instance's public IP. This gives you command-line access to the server.

```
C:\Users\Nandita\Desktop\keypair>ssh -i "kuberkey.pem" ubuntu@ec2-65-2-178-154.ap-south-1.compute.amazonaws.com
The authenticity of host 'ec2-65-2-178-154.ap-south-1.compute.amazonaws.com (65.2.178.154)' can't be established.
ED25519 key fingerprint is SHA256:6k4jKtELKrpKrkhiAXaci/SSFgdfzR/iJALAp9guhg.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-65-2-178-154.ap-south-1.compute.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

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

System information as of Thu Sep 26 15:11:47 UTC 2024

System load:  0.08          Processes:      118
Usage of /:   22.9% of 6.71GB Users logged in: 0
Memory usage: 5%           IPv4 address for enx0: 172.31.42.23
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

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Thu Sep 26 15:11:48 2024 from 13.233.177.4
```

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

- This command adds Docker's GPG key to your system. It's an essential step for secure installations

`curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -`

- This command allows the package manager to verify Docker package signatures during installation.

`curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null`

- This adds Docker's stable repository to your package sources, enabling you to install Docker and its components using the **apt** package manager.

`sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"`

```

ubuntu@ip-172-31-80-240:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
ubuntu@ip-172-31-80-240:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null
ubuntu@ip-172-31-80-240:~$ 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]
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 https://download.docker.com/linux/ubuntu noble/stable amd64 Packages [15.3 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:15 https://download.docker.com/linux/ubuntu noble/stable/main amd64 Packages [1507 kB]

```

- This command updates the local package index to include packages from the newly added Docker repository, ensuring you have the latest information.

sudo apt-get update

```

ubuntu@ip-172-31-80-240:~$ 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
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.
ubuntu@ip-172-31-80-240:~$ sudo apt-get install -y docker-ce
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
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz slirp4netns
0 upgraded, 10 newly installed, 0 to remove and 25 not upgraded.
Need to get 123 MB of archives.
After this operation, 442 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libltdl7 amd64 2.4.7-7build1 [40.3 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libslirp0 amd64 4.7.0-1ubuntu3 [63.8 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 slirp4netns amd64 1.2.1-1build2 [34.9 kB]
Get:5 https://download.docker.com/linux/ubuntu noble/stable amd64 containerd.io amd64 1.7.22-1 [29.5 MB]
Get:6 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-buildx-plugin amd64 0.17.1-1-ubuntu.24.04-noble [30.3 MB]

```

- Install the Docker Community Edition on your Ubuntu instance, enabling containerization capabilities. The **-y** flag automatically confirms the installation.

sudo apt-get install -y docker-ce

```

ubuntu@ip-172-31-80-240:~$ sudo apt-get install -y docker-ce
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
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz slirp4netns
0 upgraded, 10 newly installed, 0 to remove and 25 not upgraded.
Need to get 123 MB of archives.
After this operation, 442 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libltdl7 amd64 2.4.7-7build1 [40.3 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libslirp0 amd64 4.7.0-1ubuntu3 [63.8 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 slirp4netns amd64 1.2.1-1build2 [34.9 kB]
Get:5 https://download.docker.com/linux/ubuntu noble/stable amd64 containerd.io amd64 1.7.22-1 [29.5 MB]
Get:6 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-buildx-plugin amd64 0.17.1-1~ubuntu.24.04~noble [30.3 MB]
Get:7 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-ce-cli amd64 5:27.3.1-1~ubuntu.24.04~noble [15.0 MB]
Get:8 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-ce amd64 5:27.3.1-1~ubuntu.24.04~noble [25.6 MB]
Get:9 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-ce-rootless-extras amd64 5:27.3.1-1~ubuntu.24.04~noble [9588 kB]
Get:10 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-compose-plugin amd64 2.29.7-1~ubuntu.24.04~noble [12.7 MB]
Fetched 123 MB in 2s (74.2 MB/s)
Selecting previously unselected package pigz.
(Reading database ... 67836 files and directories currently installed.)
Preparing to unpack .../0-pigz_2.8-1_amd64.deb ...
Unpacking pigz (2.8-1) ...
Selecting previously unselected package containerd.io.
Preparing to unpack .../1-containerd.io_1.7.22-1_amd64.deb ...

```

- Creates a directory for Docker configuration files, ensuring that the necessary structure exists for Docker to operate correctly.

sudo mkdir -p /etc/docker

cat <<EOF | sudo tee /etc/docker/daemon.json

```

{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF

```

```

ubuntu@ip-172-31-80-240:~$ sudo mkdir -p /etc/docker
ubuntu@ip-172-31-80-240:~$ cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
"exec-opts": ["native.cgroupdriver=systemd"]
}

```

- Enables the Docker service to start automatically on boot.
sudo systemctl enable docker
- Reloads the systemd manager configuration.
sudo systemctl daemon-reload
- Restarts the Docker service to apply any changes made to its configuration files, ensuring that it runs with the new settings.
sudo systemctl restart docker

```

ubuntu@ip-172-31-80-240:~$ 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-172-31-80-240:~$ sudo systemctl daemon-reload
ubuntu@ip-172-31-80-240:~$ sudo systemctl restart docker

```

Step 5: Run the below command to install Kubernetes.

- This command adds the GPG key for the Kubernetes repository to ensure package authenticity during 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
```

- This adds the Kubernetes package repository to your system's sources list, enabling you to install Kubernetes components.

```
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-172-31-80-240:~$ 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-172-31-80-240:~$
```

- Updates the package index again to include packages from the Kubernetes repository, ensuring access to the latest versions.

```
sudo apt-get update
```

- Installs the essential Kubernetes components: kubelet (the agent), kubeadm (for cluster management), and kubectl (the command-line interface).

```
sudo apt-get install -y kubelet kubeadm kubectl
```

- Marks these packages to prevent them from being automatically updated, ensuring stability in your Kubernetes setup.

```
sudo apt-mark hold kubelet kubeadm kubectl
```

```
ubuntu@ip-172-31-80-240:~$ 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 https://download.docker.com/linux/ubuntu noble InRelease
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Hit:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:6 https://prod-cdn.packages.k8s.io/repositories/ipv:/kubernetes:/core:/stable:/v1.31/deb InRelease
Fetched 126 kB in 1s (240 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
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 25 not upgraded.
kubelet was already set on hold.
kubeadm was already set on hold.
kubectl was already set on hold.
```

- Enables and starts the kubelet service immediately, allowing it to begin managing the Kubernetes node.

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

- Initializes the Kubernetes cluster with a specified pod network CIDR, setting up the control plane for managing nodes and workloads.

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

```
ubuntu@ip-172-31-80-240:~$ sudo systemctl enable --now kubelet
sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.1
[preflight] Running pre-flight checks
W1020 10:02:22.795027 5411 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 for 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[preflight] If you know what you are doing, you can make a
heck non-fatal with '--ignore-preflight-errors=...'
To see the stack trace of this error execute with --v=5 or higher
ubuntu@ip-172-31-80-240:~$
```

- Installs containerd, a container runtime that manages the container lifecycle, which is necessary for Kubernetes to manage containers.

sudo apt-get install -y containerd

```
ubuntu@ip-172-31-80-240:~$ 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 25 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 MB]
Fetched 47.2 MB in 1s (70.4 MB/s)
(Reading database ... 68159 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 ... 68139 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) ...
```

- Creates a directory for containerd configuration, ensuring that the necessary structure exists for its operation.

sudo mkdir -p /etc/containerd

- Generates a default configuration file for containerd and saves it, ensuring that the runtime is correctly configured.

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

```
ubuntu@ip-172-31-80-240:~$ 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
  max_recv_message_size = 16777216
  max_send_message_size = 16777216
  tcp_address = ""
  tcp_tls_ca = ""
  tcp_tls_cert = ""
```


- Restarts the containerd service to apply the new configuration settings, ensuring it runs with the updated parameters.

sudo systemctl restart containerd

- Enables containerd to start automatically on boot, ensuring its availability after a system restart

sudo systemctl enable containerd

- Checks the current status of the containerd service, confirming that it is running properly.

sudo systemctl status containerd

```
ubuntu@ip-172-31-80-240:~$ sudo systemctl restart containerd
ubuntu@ip-172-31-80-240:~$ sudo systemctl enable containerd
ubuntu@ip-172-31-80-240:~$ sudo systemctl status containerd
● containerd.service - containerd container runtime
   Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: enabled)
   Active: active (running) since Sun 2024-10-20 10:05:40 UTC; 21s ago
     Docs: https://containerd.io
   Main PID: 5885 (containerd)
    Tasks: 8
   Memory: 13.1M (peak: 13.9M)
      CPU: 129ms
   CGroup: /system.slice/containerd.service
           └─5885 /usr/bin/containerd

Oct 20 10:05:40 ip-172-31-80-240 containerd[5885]: time="2024-10-20T10:05:40.905924648Z" level=error msg="failed to load cni during init, please check the cni config"
Oct 20 10:05:40 ip-172-31-80-240 containerd[5885]: time="2024-10-20T10:05:40.906093670Z" level=info msg="Start subscribing containerd event"
Oct 20 10:05:40 ip-172-31-80-240 containerd[5885]: time="2024-10-20T10:05:40.906132437Z" level=info msg="serving... address=/run/containerd/containerd.sock"
Oct 20 10:05:40 ip-172-31-80-240 containerd[5885]: time="2024-10-20T10:05:40.906135956Z" level=info msg="Start recovering state"
Oct 20 10:05:40 ip-172-31-80-240 containerd[5885]: time="2024-10-20T10:05:40.906185395Z" level=info msg="serving... address=/run/containerd/containerd.sock"
Oct 20 10:05:40 ip-172-31-80-240 containerd[5885]: time="2024-10-20T10:05:40.906205371Z" level=info msg="Start event monitor"
Oct 20 10:05:40 ip-172-31-80-240 containerd[5885]: time="2024-10-20T10:05:40.906215409Z" level=info msg="Start snapshots syncer"
Oct 20 10:05:40 ip-172-31-80-240 containerd[5885]: time="2024-10-20T10:05:40.906223526Z" level=info msg="Start cni network config syncer for default"
Oct 20 10:05:40 ip-172-31-80-240 containerd[5885]: time="2024-10-20T10:05:40.906229655Z" level=info msg="Start streaming server"
Oct 20 10:05:40 ip-172-31-80-240 containerd[5885]: time="2024-10-20T10:05:40.906282756Z" level=info msg="containerd successfully booted in 0.028768s"
since 1-21/21 (FPM)
```

- Installs socat, a utility that facilitates communication between two data streams, often used for forwarding traffic in Kubernetes setups.

sudo apt-get install -y socat

```
ubuntu@ip-172-31-80-240:~$ 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 25 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 (11.8 MB/s)
Selecting previously unselected package socat.
(Reading database ... 68203 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 processes... [=
Scanning processes... [==
Scanning processes... [===
Scanning processes... [====
Scanning processes... [=====
Scanning processes... [=====
Scanning processes... [=====
Scanning processes... [=====
```

Step 6: Initialize the Kubeccluster

- This command re-initializes the Kubernetes cluster with a specific pod network, ensuring that the networking is set up correctly for pods.

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

```
ubuntu@ip-172-31-80-240:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.1
[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'
W1020 10:07:42.067638 6134 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-172-31-80-240 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 172.31.80.240]
[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-172-31-80-240 localhost] and IPs [172.31.80.240 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-80-240 localhost] and IPs [172.31.80.240 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "super-admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
```

Copy the mkdir and chown commands from the top and execute them.

- Creates a directory for Kubernetes configuration files in the user's home directory, ensuring that the user has a place to manage Kubernetes configurations.

mkdir -p \$HOME/.kube

- Copies the Kubernetes admin configuration file to the user's home directory, allowing access to the Kubernetes cluster using kubectl.

sudo cp -i /etc/kubernetes/admin.conf \$HOME/.kube/config

- Changes the ownership of the Kubernetes config file to the current user, allowing them to manage the cluster without needing elevated privileges

sudo chown \$(id -u):\$(id -g) \$HOME/.kube/config

```
ubuntu@ip-172-31-80-240:~$ mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Step 7: Add Networking Plugin (Flannel)

- This command deploys the Flannel networking plugin, enabling efficient communication between pods across the cluster.

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-172-31-80-240:~$ 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 8: Deploy an Nginx Application

- Deploys an example Nginx application to the Kubernetes cluster, demonstrating how to manage containerized applications.

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

```
ubuntu@ip-172-31-80-240:~$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
```

- Retrieves the status of all pods in the cluster, allowing you to verify that the Nginx deployment is running as expected.

kubectl get pods

```
ubuntu@ip-172-31-80-240:~$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
nginx-deployment-d556bf558-87mfp	0/1	Pending	0	26s
nginx-deployment-d556bf558-fcfx2	0/1	Pending	0	26s

- Stores the name of the first Nginx pod in a variable for easy reference in subsequent commands.

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

```
ubuntu@ip-172-31-80-240:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8080:80
error: unable to forward port because pod is not running. Current status=Pending
```

- Lists all nodes in the Kubernetes cluster, providing an overview of the cluster's status and available resources.

kubectl get nodes

```
ubuntu@ip-172-31-80-240:~$ kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
ip-172-31-80-240	Ready	control-plane	3m46s	v1.31.1

- Forwards traffic from your local machine's port 8080 to port 80 of the specified Nginx pod, enabling access to the application

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

kubectl port-forward \$POD_NAME 8080:80

```
ubuntu@ip-172-31-80-240:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
ubuntu@ip-172-31-80-240:~$ kubectl port-forward $POD_NAME 8080:80
error: unable to forward port because pod is not running. Current status=Pending
ubuntu@ip-172-31-80-240:~$ command kubectl taint nodes --all node-role.kubernetes.io/control-plane:NoSchedule-
```

- Removes the taint from control plane nodes, allowing workloads to be scheduled on them, which is useful for testing and development.

command kubectl taint nodes--all

node-role.kubernetes.io/control-plane-node/ip-172-31-20-171 untainted

```
ubuntu@ip-172-31-80-240:~$ command kubectl taint nodes --all node-role.kubernetes.io/control-plane:NoSchedule-
node/ip-172-31-80-240 untainted
```

- Again lists all nodes to confirm that the taint has been successfully removed.

kubectl get nodes

```
ubuntu@ip-172-31-80-240:~$ kubectl get nodes
NAME                STATUS    ROLES    AGE     VERSION
ip-172-31-80-240    Ready     control-plane  9m52s   v1.31.1
```

- The command retrieves and displays a list of all pods in the current Kubernetes.

kubectl get pods

```
ubuntu@ip-172-31-80-240:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-87mfp    1/1     Running   0           8m
nginx-deployment-d556bf558-fcfx2    1/1     Running   0           8m
```

- The command forwards local port 8082 to port 80 of the specified pod

kubectl port-forward \$POD_NAME 8082:80

This my POD_NAME nginx-deployment-d556bf558-87mfp

```
ubuntu@ip-172-31-80-240:~$ kubectl port-forward nginx-deployment-d556bf558-87mfp 8082:80
Forwarding from 127.0.0.1:8082 -> 80
Forwarding from [::1]:8082 -> 80
Handling connection for 8082
□
```

Step 9: 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://127.0.0.1:8082>

```
ubuntu@ip-172-31-80-240:~$ curl --head http://127.0.0.1:8082
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Sun, 20 Oct 2024 10:20:51 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
```

- The command kubectl get services retrieves and displays a list of all services in the current Kubernetes namespace.

kubectl get services

```
ubuntu@ip-172-31-80-240:~$ kubectl get services
NAME                TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
kubernetes          ClusterIP   10.96.0.1     <none>          443/TCP    29m
```

- The command creates a new deployment named "nginx" that runs the NGINX web server container in a Kubernetes cluster.

kubectl create deployment nginx --image=nginx

```
ubuntu@ip-172-31-80-240:~$ kubectl create deployment nginx --image=nginx
deployment.apps/nginx created
```

- The command `kubectl get deployments` lists all deployments in the current Kubernetes namespace

kubectl get deployments

```
ubuntu@ip-172-31-80-240:~$ kubectl get deployments
NAME                READY    UP-TO-DATE    AVAILABLE    AGE
nginx                1/1      1              1            11s
nginx-deployment    2/2      2              2            29m
```

- The command creates a NodePort service that exposes the NGINX deployment on port 80, allowing external access to the application through any node's IP address.

kubectl expose deployment nginx --type=NodePort --port=80

```
ubuntu@ip-172-31-80-240:~$ kubectl expose deployment nginx --type=NodePort --port=80
service/nginx exposed
```

- Nginx server is running successfully on the EC2 instance, and it's accessible locally via localhost on port 31301.

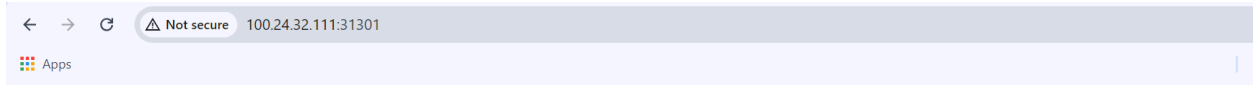
curl http://100.24.32.111:31301

```
ubuntu@ip-172-31-80-240:~$ curl http://localhost:31301
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Type this in the url **http://100.24.32.111:31301** you can see the output



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Conclusion:

This case study demonstrated the successfully installed Kubernetes and Docker on an AWS EC2 Ubuntu instance, configured essential settings, and initialized a Kubernetes cluster. We deployed an Nginx server using a Kubernetes Deployment and applied a Flannel networking plugin for pod communication. By verifying the pod status and forwarding ports, we accessed the Nginx server locally. The successful **200 OK** response from the curl command confirmed that the deployment was operational. This setup demonstrated fundamental Kubernetes operations, including cluster management, application deployment, and verification, showcasing Kubernetes' power in orchestrating containerized applications efficiently.