

### Experiment 3

**Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.**

#### 1) Launch an EC2 Instance:

Choose Amazon Linux as the operating system for your instance.

The screenshot shows the AWS Management Console interface for launching an EC2 instance. The 'Name and tags' section has a text input field with 'Master' and a link to 'Add additional tags'. The 'Application and OS Images (Amazon Machine Image)' section includes a search bar with the placeholder text 'Search our full catalog including 1000s of application and OS images'. Below the search bar are tabs for 'Recents' and 'Quick Start'. The 'Quick Start' tab is active, displaying a grid of AMI icons for Amazon Linux, macOS, Ubuntu, Windows, Red Hat, and SUSE Linux. To the right of the grid is a link 'Browse more AMIs' with a magnifying glass icon, and a note 'Including AMIs from AWS, Marketplace and the Community'.

**Note:** The AWS free tier provides a t2.micro instance (1 CPU, 1 GiB RAM), but Kubernetes requires a minimum of 2 CPUs and 2 GiB RAM. So, make sure to select **t2.medium**

The screenshot shows the 'Instance type' section of the AWS Management Console. It features a dropdown menu with 't2.medium' selected. Below the dropdown, the following details are listed: 'Family: t2', '2 vCPU', '4 GiB Memory', and 'Current generation: true'. Pricing information is provided for different operating systems: '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', and 'On-Demand SUSE base pricing: 0.1464 USD per Hour'. To the right of the dropdown is a radio button labeled 'All generations' and a link 'Compare instance types'. At the bottom, a note states 'Additional costs apply for AMIs with pre-installed software'.

**2) select a Key Pair:**


You can either use the default key pair provided by AWS or create a new one for better security. Click on **Create**.

▼ 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*

▼

 [Create new key pair](#)

**All your instances will appear here**

Instances (3) Info

Last updated less than a minute ago

Connect

Instance state ▾

Actions ▾

Launch instances ▾

Find Instance by attribute or tag (case-sensitive)

Running ▾

< 1 >

<input type="checkbox"/>	Name <div></div>	Instance ID	Instance state ▾	Instanc... ▾	Status check	Alarm status	Availabi... ▾	Public I... ▾	Pu
<input type="checkbox"/>	Master <div></div>	i-0d3f7911...	<div><div></div>Running</div> <div><div></div><div></div></div>	t2.medium	<div><div></div>2/2 checks p...</div> <div><div></div>View alarms +</div>		us-east-1d	ec2-54-24...	54
<input type="checkbox"/>	Computer2	i-09a31ba0...	<div><div></div>Running</div> <div><div></div><div></div></div>	t2.medium	<div><div></div>Initializing</div> <div><div></div>View alarms +</div>		us-east-1d	ec2-34-20...	34
<input type="checkbox"/>	Computer1	i-041bb34...	<div><div></div>Running</div> <div><div></div><div></div></div>	t2.medium	<div><div></div>2/2 checks p...</div> <div><div></div>View alarms +</div>		us-east-1d	ec2-54-17...	54

▼ Summary

Number of instances Info

Software Image (AMI)

Amazon Linux 2023 AMI 2023.5.2...[read more](#)  
ami-0182f373e66f89c85

Virtual server type (instance type)

t2.medium

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 8 GiB

Cancel

Launch instance

[Review commands](#)

### 3) Connect to the Instance:

Navigate to the instances page, locate your instance, and click on its ID. Then, select **Connect** and keep the default connection settings. Finally, click **Connect** to start your session.

EC2 > Instances > i-0d3f7911e0aabcc35

**Instance summary for i-0d3f7911e0aabcc35 (Master)** [Info](#)

Updated less than a minute ago

[Refresh](#) [Connect](#) [Instance state ▼](#) [Actions ▼](#)

Instance ID i-0d3f7911e0aabcc35 (Master)	Public IPv4 address 54.242.215.34   <a href="#">open address</a>	Private IPv4 addresses 172.31.22.81
IPv6 address -	Instance state <span>Running</span>	Public IPv4 DNS ec2-54-242-215-34.compute-1.amazonaws.com   <a href="#">open address</a>
Hostname type IP name: ip-172-31-22-81.ec2.internal	Private IP DNS name (IPv4 only) ip-172-31-22-81.ec2.internal	Elastic IP addresses -
Answer private resource DNS name IPv4 (A)	Instance type t2.medium	AWS Compute Optimizer finding <a href="#">Opt-in to AWS Compute Optimizer for recommendations.</a>   <a href="#">Learn more</a>
Auto-assigned IP address 54.242.215.34 [Public IP]	VPC ID vpc-0f7970ea32a533bcc	Auto Scaling Group name -
IAM Role -	Subnet ID subnet-09ba566295275f771	
IMDSv2 Required	Instance ARN arn:aws:ec2:us-east-1:74255988891:instance/i-0d3f7911e0aabcc35	

[Details](#) | [Status and alarms](#) | [Monitoring](#) | [Security](#) | [Networking](#) | [Storage](#) | [Tags](#)

EC2 Instance Connect | Session Manager | SSH client | EC2 serial console

**⚠ Port 22 (SSH) is open to all IPv4 addresses**  
Port 22 (SSH) is currently open to all IPv4 addresses, indicated by **0.0.0.0/0** in the inbound rule in [your security group](#). For increased security, consider restricting access to only the EC2 Instance Connect service IP addresses for your Region: 18.206.107.24/29. [Learn more](#).

Instance ID  
i-0d3f7911e0aabcc35 (Master)

Connection Type

☒ **Connect using EC2 Instance Connect**  
Connect using the EC2 Instance Connect browser-based client, with a public IPv4 address.

☐ **Connect using EC2 Instance Connect Endpoint**  
Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

Public IPv4 address  
54.242.215.34

Username  
Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ec2-user.

**Note:** In most cases, the default username, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

[Cancel](#) [Connect](#)

[EC2](#) > [Instances](#) > i-041bb34892373b3d7

**Instance summary for i-041bb34892373b3d7 (Computer1)** [info](#)

Updated less than a minute ago

[Refresh](#) [Connect](#) [Instance state ▼](#) [Actions ▼](#)

<b>Instance ID</b> i-041bb34892373b3d7 (Computer1)	<b>Public IPv4 address</b> 54.174.132.164   <a href="#">open address</a>	<b>Private IPv4 addresses</b> 172.31.28.157
<b>IPv6 address</b> –	<b>Instance state</b> Running	<b>Public IPv4 DNS</b> ec2-54-174-132-164.compute-1.amazonaws.com   <a href="#">open address</a>
<b>Hostname type</b> IP name: ip-172-31-28-157.ec2.internal	<b>Private IP DNS name (IPv4 only)</b> ip-172-31-28-157.ec2.internal	<b>Elastic IP addresses</b> –
<b>Answer private resource DNS name</b> IPv4 (A)	<b>Instance type</b> t2.medium	<b>AWS Compute Optimizer finding</b> <a href="#">Opt-in to AWS Compute Optimizer for recommendations.</a>   <a href="#">Learn more</a>
<b>Auto-assigned IP address</b> 54.174.132.164 [Public IP]	<b>VPC ID</b> vpc-0f7970ea32a533bcc	<b>Auto Scaling Group name</b> –
<b>IAM Role</b> –	<b>Subnet ID</b> subnet-09ba566295275f771	
<b>IMDSv2</b> Required	<b>Instance ARN</b> arn:aws:ec2:us-east-1:742555988891:instance/i-041bb34892373b3d7	

[Details](#) | [Status and alarms](#) | [Monitoring](#) | [Security](#) | [Networking](#) | [Storage](#) | [Tags](#)

[EC2](#) > [Instances](#) > i-041bb34892373b3d7

**Instance summary for i-041bb34892373b3d7 (Computer1)** [info](#)

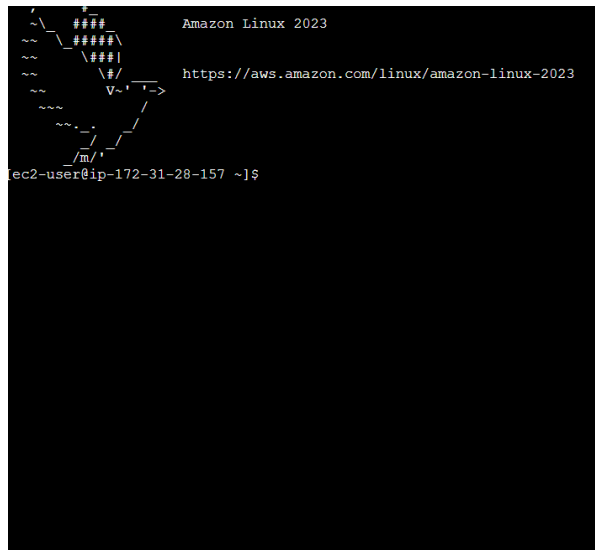
Updated less than a minute ago

[Refresh](#) [Connect](#) [Instance state ▼](#) [Actions ▼](#)

<b>Instance ID</b> i-041bb34892373b3d7 (Computer1)	<b>Public IPv4 address</b> 54.174.132.164   <a href="#">open address</a>	<b>Private IPv4 addresses</b> 172.31.28.157
<b>IPv6 address</b> –	<b>Instance state</b> Running	<b>Public IPv4 DNS</b> ec2-54-174-132-164.compute-1.amazonaws.com   <a href="#">open address</a>
<b>Hostname type</b> IP name: ip-172-31-28-157.ec2.internal	<b>Private IP DNS name (IPv4 only)</b> ip-172-31-28-157.ec2.internal	<b>Elastic IP addresses</b> –
<b>Answer private resource DNS name</b> IPv4 (A)	<b>Instance type</b> t2.medium	<b>AWS Compute Optimizer finding</b> <a href="#">Opt-in to AWS Compute Optimizer for recommendations.</a>   <a href="#">Learn more</a>
<b>Auto-assigned IP address</b> 54.174.132.164 [Public IP]	<b>VPC ID</b> vpc-0f7970ea32a533bcc	<b>Auto Scaling Group name</b> –
<b>IAM Role</b> –	<b>Subnet ID</b> subnet-09ba566295275f771	
<b>IMDSv2</b> Required	<b>Instance ARN</b> arn:aws:ec2:us-east-1:742555988891:instance/i-041bb34892373b3d7	

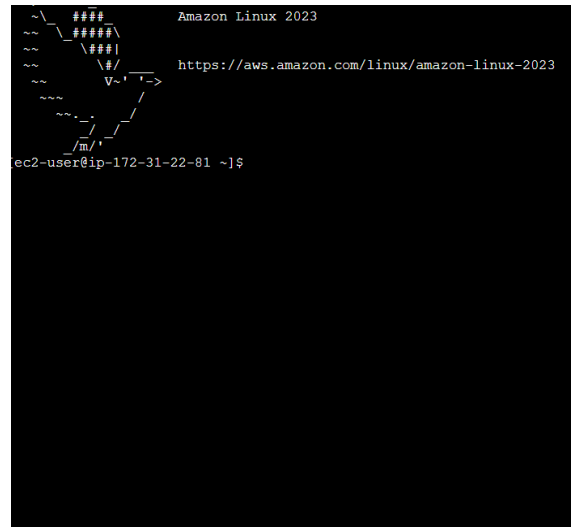
[Details](#) | [Status and alarms](#) | [Monitoring](#) | [Security](#) | [Networking](#) | [Storage](#) | [Tags](#)

All the three server



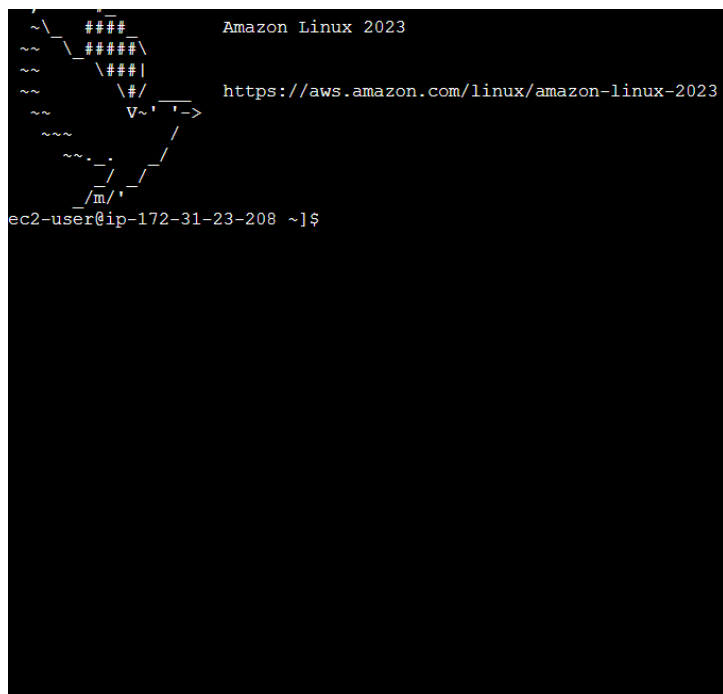
i-041bb34892373b3d7 (Computer1)

PublicIPs: 54.174.132.164 PrivateIPs: 172.31.28.157



i-0d3f7911e0aabcc35 (Master )

PublicIPs: 54.242.215.34 PrivateIPs: 172.31.22.81



i-09a31ba0572c10bdb (Computer2)

PublicIPs: 34.204.51.153 PrivateIPs: 172.31.23.208

## Step 2: Installation of Docker

### 1)Switch to Root User:

Run **sudo su** to get root-level access in the terminal.

```
~/m/'  
[ec2-user@ip-172-31-22-81 ~]$ sudo su  
[root@ip-172-31-22-81 ec2-user]#
```

### 2)Install Docker:

Use the YUM package manager to install Docker by running:

**yum install docker -y**

```
[root@ip-172-31-22-81 ec2-user]# yum install docker -y  
Last metadata expiration check: 0:11:19 ago on Sun Sep 15 06:50:06 2024.  
Dependencies resolved.  
=====
```

Package	Architecture	Version	Repository	Size
Installing: docker	x86_64	25.0.6-1.amzn2023.0.2	amazonlinux	44 M
Installing dependencies:				
containerd	x86_64	1.7.20-1.amzn2023.0.1	amazonlinux	35 M
iptables-libs	x86_64	1.8.8-3.amzn2023.0.2	amazonlinux	401 k
iptables-nft	x86_64	1.8.8-3.amzn2023.0.2	amazonlinux	183 k
libcgroup	x86_64	3.0-1.amzn2023.0.1	amazonlinux	75 k
libnetfilter_conntrack	x86_64	1.0.8-2.amzn2023.0.2	amazonlinux	58 k
libnftnl	x86_64	1.0.1-19.amzn2023.0.2	amazonlinux	30 k
libnftnl	x86_64	1.2.2-2.amzn2023.0.2	amazonlinux	84 k
pigz	x86_64	2.5-1.amzn2023.0.3	amazonlinux	83 k
runc	x86_64	1.1.13-1.amzn2023.0.1	amazonlinux	3.2 M

```
=====
```

Transaction Summary

-----

Install 10 Packages

Total download size: 84 M  
Installed size: 317 M

Downloading Packages:

(1/10): iptables-libs-1.8.8-3.amzn2023.0.2.x86_64.rpm	6.3 MB/s   401 kB	00:00
(2/10): iptables-nft-1.8.8-3.amzn2023.0.2.x86_64.rpm	6.0 MB/s   183 kB	00:00
(3/10): libcgroup-3.0-1.amzn2023.0.1.x86_64.rpm	2.8 MB/s   75 kB	00:00

```
-----
```

Install 10 Packages

Total download size: 84 M  
Installed size: 317 M

Downloading Packages:

(1/10): iptables-libs-1.8.8-3.amzn2023.0.2.x86_64.rpm	6.3 MB/s   401 kB	00:00
(2/10): iptables-nft-1.8.8-3.amzn2023.0.2.x86_64.rpm	6.0 MB/s   183 kB	00:00
(3/10): libcgroup-3.0-1.amzn2023.0.1.x86_64.rpm	2.8 MB/s   75 kB	00:00
(4/10): libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64.rpm	2.6 MB/s   58 kB	00:00
(5/10): libnftnl-1.0.1-19.amzn2023.0.2.x86_64.rpm	1.4 MB/s   30 kB	00:00
(6/10): libnftnl-1.2.2-2.amzn2023.0.2.x86_64.rpm	1.9 MB/s   84 kB	00:00
(7/10): pigz-2.5-1.amzn2023.0.3.x86_64.rpm	1.1 MB/s   83 kB	00:00
(8/10): runc-1.1.13-1.amzn2023.0.1.x86_64.rpm	22 MB/s   3.2 MB	00:00
(9/10): containerd-1.7.20-1.amzn2023.0.1.x86_64.rpm	41 MB/s   35 MB	00:00
(10/10): docker-25.0.6-1.amzn2023.0.2.x86_64.rpm	38 MB/s   44 MB	00:01

Total

70 MB/s | 84 MB 00:01

Running transaction check  
Transaction check succeeded.  
Running transaction test  
Transaction test succeeded.  
Running transaction

Preparing	:	1/1
Installing	: runc-1.1.13-1.amzn2023.0.1.x86_64	1/10
Installing	: containerd-1.7.20-1.amzn2023.0.1.x86_64	2/10
Running scriptlet:	containerd-1.7.20-1.amzn2023.0.1.x86_64	2/10
Installing	: pigz-2.5-1.amzn2023.0.3.x86_64	3/10
Installing	: libnftnl-1.2.2-2.amzn2023.0.2.x86_64	4/10
Installing	: libnftnl-1.0.1-19.amzn2023.0.2.x86_64	5/10
Installing	: libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64	6/10
Installing	: iptables-libs-1.8.8-3.amzn2023.0.2.x86_64	7/10
Installing	: iptables-nft-1.8.8-3.amzn2023.0.2.x86_64	8/10
Running scriptlet:	iptables-nft-1.8.8-3.amzn2023.0.2.x86_64	8/10
Installing	: libcgroup-3.0-1.amzn2023.0.1.x86_64	9/10
Running scriptlet:	docker-25.0.6-1.amzn2023.0.2.x86_64	10/10
Installing	: docker-25.0.6-1.amzn2023.0.2.x86_64	10/10
Running scriptlet:	docker-25.0.6-1.amzn2023.0.2.x86_64	10/10

Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.

```

reated symlink /etc/systemd/system/sockets.target.wants/docker.socket -> /usr/lib/systemd/system/docker.socket.

Verifying      : containerd-1.7.20-1.amzn2023.0.1.x86_64      1/10
Verifying      : docker-25.0.6-1.amzn2023.0.2.x86_64        2/10
Verifying      : iptables-libs-1.8.8-3.amzn2023.0.2.x86_64   3/10
Verifying      : iptables-nft-1.8.8-3.amzn2023.0.2.x86_64   4/10
Verifying      : libcgroup-3.0-1.amzn2023.0.1.x86_64         5/10
Verifying      : libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64 6/10
Verifying      : libnftnl-1.0.1-19.amzn2023.0.2.x86_64      7/10
Verifying      : libnftnl-1.2.2-2.amzn2023.0.2.x86_64       8/10
Verifying      : pigz-2.5-1.amzn2023.0.3.x86_64              9/10
Verifying      : runc-1.1.13-1.amzn2023.0.1.x86_64           10/10

Installed:
containerd-1.7.20-1.amzn2023.0.1.x86_64  docker-25.0.6-1.amzn2023.0.2.x86_64  iptables-libs-1.8.8-3.amzn2023.0.2.x86_64  iptables-nft-1.8.8-3.amzn2023.0.2.x86_64
libcgroup-3.0-1.amzn2023.0.1.x86_64      libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64  libnftnl-1.0.1-19.amzn2023.0.2.x86_64  libnftnl-1.2.2-2.amzn2023.0.2.x86_64
pigz-2.5-1.amzn2023.0.3.x86_64          runc-1.1.13-1.amzn2023.0.1.x86_64

Complete!
[root@ip-172-31-22-81 ec2-user]#

```

### 3) Install in all devices

#### Computer 1

```

Installed:
containerd-1.7.20-1.amzn2023.0.1.x86_64  docker-25.0.6-1.amzn2023.0.2.x86_64  iptables-libs-1.8.8-3.amzn2023.0.2.x86_64  iptables-nft-1.8.8-3.amzn2023.0.2.x86_64
libcgroup-3.0-1.amzn2023.0.1.x86_64      libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64  libnftnl-1.0.1-19.amzn2023.0.2.x86_64  libnftnl-1.2.2-2.amzn2023.0.2.x86_64
pigz-2.5-1.amzn2023.0.3.x86_64          runc-1.1.13-1.amzn2023.0.1.x86_64

Complete!
ec2-user@ip-172-31-28-157 ~]$

```

#### Computer2

```

Installed:
containerd-1.7.20-1.amzn2023.0.1.x86_64  docker-25.0.6-1.amzn2023.0.2.x86_64  iptables-libs-1.8.8-3.amzn2023.0.2.x86_64  iptables-nft-1.8.8-3.amzn2023.0.2.x86_64
libcgroup-3.0-1.amzn2023.0.1.x86_64      libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64  libnftnl-1.0.1-19.amzn2023.0.2.x86_64  libnftnl-1.2.2-2.amzn2023.0.2.x86_64
pigz-2.5-1.amzn2023.0.3.x86_64          runc-1.1.13-1.amzn2023.0.1.x86_64

Complete!
[ec2-user@ip-172-31-23-208 ~]$

```

i-09a31ba0572c10bdb (Computer2)  
PublicIPs: 34.204.51.153 PrivateIPs: 172.31.23.208

### 4) Configure Docker Daemon:

You need to configure Docker to use the systemd cgroup driver.

Run the following commands: `cd /etc/docker`

```

cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF

```

```
[root@ip-172-31-22-81 docker]# cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
EOF
{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
```

### 5)Enable and Start Docker:

Start Docker by running:

bash

Copy code

```
sudo systemctl enable docker
```

```
sudo systemctl daemon-reload
```

```
sudo systemctl restart docker
```

```
[root@ip-172-31-22-81 docker]# sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
docker -v
Docker version 25.0.5, build 5dc9bcc
[root@ip-172-31-22-81 docker]#
```

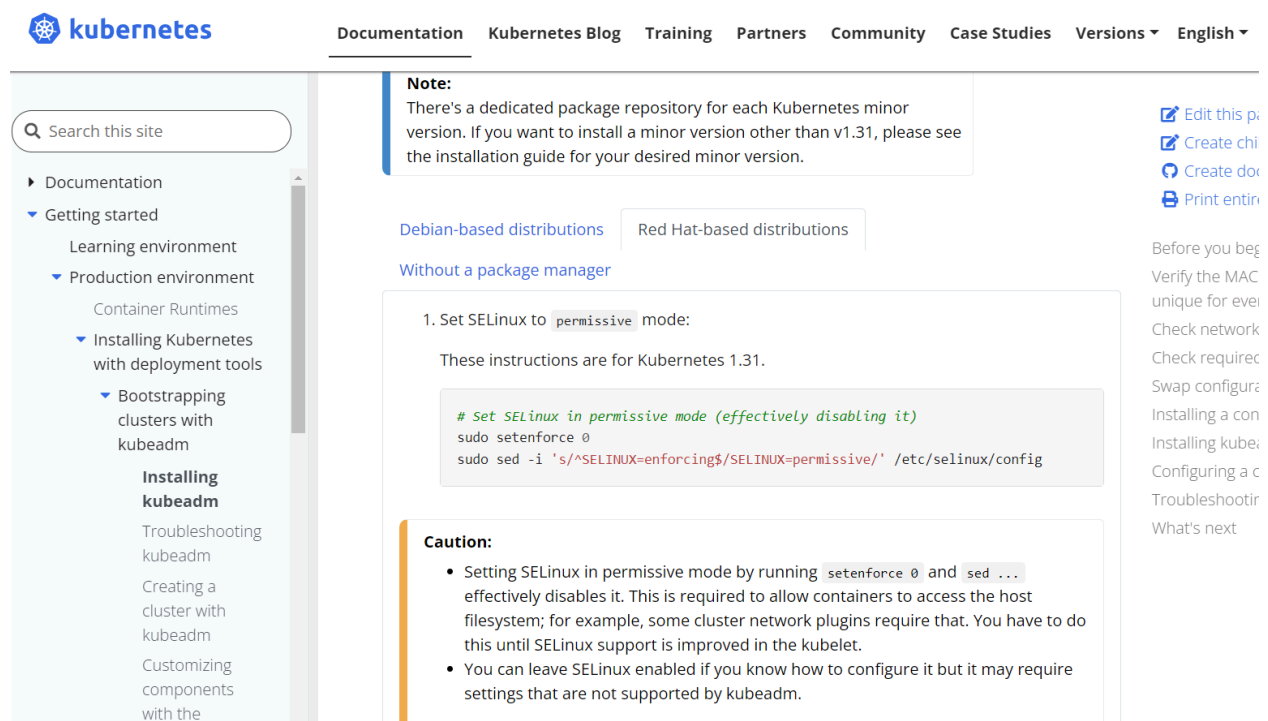


## Step 3: Installing Kubernetes

For installing kubernetes, we will be using kubeadm, a framework used for creating kubernetes clusters using command line.

<https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm>

The following will be visible when you visit the website.



The screenshot shows the Kubernetes documentation website. The left sidebar contains a search bar and a navigation menu with categories like Documentation, Getting started, Learning environment, Production environment, Container Runtimes, Installing Kubernetes with deployment tools, Bootstrapping clusters with kubeadm, and Troubleshooting kubeadm. The main content area is titled 'Installing kubeadm' and includes a 'Note' about package repositories, tabs for 'Debian-based distributions' and 'Red Hat-based distributions', and a section 'Without a package manager'. This section contains a numbered list starting with '1. Set SELinux to permissive mode:' and a code block showing the commands to set SELinux to permissive mode. A 'Caution' box follows, explaining the implications of setting SELinux to permissive mode.

**Note:**  
There's a dedicated package repository for each Kubernetes minor version. If you want to install a minor version other than v1.31, please see the installation guide for your desired minor version.

Debian-based distributions | Red Hat-based distributions

Without a package manager

1. Set SELinux to `permissive` mode:

These instructions are for Kubernetes 1.31.

```
# Set SELinux in permissive mode (effectively disabling it)
sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
```

**Caution:**

- Setting SELinux in permissive mode by running `setenforce 0` and `sed ...` effectively disables it. This is required to allow containers to access the host filesystem; for example, some cluster network plugins require that. You have to do this until SELinux support is improved in the kubelet.
- You can leave SELinux enabled if you know how to configure it but it may require settings that are not supported by kubeadm.

### 1) Prepare the System:

Configure SELinux to run in permissive mode to avoid permission issues during the Kubernetes setup:

bash

Copy code

```
sudo setenforce 0
```

```
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/'  
/etc/selinux/config
```

```
[root@ip-172-31-22-81 ec2-user]# sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[root@ip-172-31-22-81 ec2-user]#
```

### Computer 1

```
[ec2-user@ip-172-31-28-157 ~]$ sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[ec2-user@ip-172-31-28-157 ~]$
```

### Computer 2

```
[ec2-user@ip-172-31-23-208 ~]$ sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[ec2-user@ip-172-31-23-208 ~]$
```

## 2)Add the Kubernetes Repo:

Create a Kubernetes repository by running:

bash

Copy code

```
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repo
data/repo
md.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
```

```
[root@ip-172-31-22-81 ec2-user]# cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
[root@ip-172-31-22-81 ec2-user]#
```

i-Od3f7911e0aabcc35 (Master )

PublicIPs: 54.242.215.34 PrivateIPs: 172.31.22.81

```
[root@ip-172-31-22-81 ec2-user]# sudo yum update
Kubernetes
Kubernetes
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-172-31-22-81 ec2-user]#
```

## yum repolist

This command shows the repositories created on the machine.

```
[root@ip-172-31-22-81 ec2-user]# yum repolist
repo id                                repo name
amazonlinux                            Amazon Linux 2023 repository
kernel-livepatch                       Amazon Linux 2023 Kernel Livepatch repository
kubernetes                             Kubernetes
[root@ip-172-31-22-81 ec2-user]#
```

## 3)Install Kubernetes Components:

Now, install the required Kubernetes tools by executing:

bash

Copy code

```
sudo yum install -y kubelet kubeadm kubectl
--disableexcludes=kubernetes
```

```
[root@ip-172-31-22-81 ec2-user]# sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Last metadata expiration check: 0:03:06 ago on Sun Sep 15 07:47:07 2024.
Dependencies resolved.

```

Package	Architecture	Version	Repository	Size
Installing:				
kubeadm	x86_64	1.30.5-150500.1.1	kubernetes	10 M
kubect1	x86_64	1.30.5-150500.1.1	kubernetes	10 M
kubelet	x86_64	1.30.5-150500.1.1	kubernetes	17 M
Installing dependencies:				
conntrack-tools	x86_64	1.4.6-2.amzn2023.0.2	amazonlinux	208 k
cri-tools	x86_64	1.30.1-150500.1.1	kubernetes	8.6 M
kubernetes-cni	x86_64	1.4.0-150500.1.1	kubernetes	6.7 M
libnetfilter_cthelper	x86_64	1.0.0-21.amzn2023.0.2	amazonlinux	24 k
libnetfilter_cttimeout	x86_64	1.0.0-19.amzn2023.0.2	amazonlinux	24 k
libnetfilter_queue	x86_64	1.0.5-2.amzn2023.0.2	amazonlinux	30 k

```
Transaction Summary
Install 9 Packages

Total download size: 53 M
Installed size: 292 M
Downloading Packages:
(1/9): libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64.rpm 432 kB/s | 24 kB 00:00
(2/9): libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64.rpm 374 kB/s | 24 kB 00:00
```

```
[ec2-user@ip-172-31-28-157 ~]$ sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Last metadata expiration check: 0:03:08 ago on Sun Sep 15 07:47:52 2024.
Dependencies resolved.

```

Package	Architecture	Version	Repository	Size
Installing:				
kubeadm	x86_64	1.30.5-150500.1.1	kubernetes	10 M
kubect1	x86_64	1.30.5-150500.1.1	kubernetes	10 M
kubelet	x86_64	1.30.5-150500.1.1	kubernetes	17 M
Installing dependencies:				
conntrack-tools	x86_64	1.4.6-2.amzn2023.0.2	amazonlinux	208 k
cri-tools	x86_64	1.30.1-150500.1.1	kubernetes	8.6 M
kubernetes-cni	x86_64	1.4.0-150500.1.1	kubernetes	6.7 M
libnetfilter_cthelper	x86_64	1.0.0-21.amzn2023.0.2	amazonlinux	24 k
libnetfilter_cttimeout	x86_64	1.0.0-19.amzn2023.0.2	amazonlinux	24 k
libnetfilter_queue	x86_64	1.0.5-2.amzn2023.0.2	amazonlinux	30 k

```
Transaction Summary
Install 9 Packages

Total download size: 53 M
Installed size: 292 M
Downloading Packages:
(1/9): libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64.rpm 426 kB/s | 24 kB 00:00
```

```
[ec2-user@ip-172-31-23-208 ~]$ sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Last metadata expiration check: 0:03:07 ago on Sun Sep 15 07:47:58 2024.
Dependencies resolved.

```

Package	Architecture	Version	Repository	Size
Installing:				
kubeadm	x86_64	1.30.5-150500.1.1	kubernetes	10 M
kubect1	x86_64	1.30.5-150500.1.1	kubernetes	10 M
kubelet	x86_64	1.30.5-150500.1.1	kubernetes	17 M
Installing dependencies:				
conntrack-tools	x86_64	1.4.6-2.amzn2023.0.2	amazonlinux	208 k
cri-tools	x86_64	1.30.1-150500.1.1	kubernetes	8.6 M
kubernetes-cni	x86_64	1.4.0-150500.1.1	kubernetes	6.7 M
libnetfilter_cthelper	x86_64	1.0.0-21.amzn2023.0.2	amazonlinux	24 k
libnetfilter_cttimeout	x86_64	1.0.0-19.amzn2023.0.2	amazonlinux	24 k
libnetfilter_queue	x86_64	1.0.5-2.amzn2023.0.2	amazonlinux	30 k

```
Transaction Summary
Install 9 Packages

Total download size: 53 M
Installed size: 292 M
Downloading Packages:
(1/9): libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64.rpm 455 kB/s | 24 kB 00:00
```

Configure System Settings:  
Run the following to configure a network bridge:

```
bash
Copy code
sudo swapoff -a
echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p
```

```
[root@ip-172-31-22-81 ec2-user]# sudo swapoff -a
echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p
net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-iptables = 1
[root@ip-172-31-22-81 ec2-user]#
```

i-0d3f7911e0aabcc35 (Master )

PublicIPs: 54.242.215.34 PrivateIPs: 172.31.22.81

## PERFORM THE FOLLOWING ON ONLY THE MASTER MACHINE

### Initialize Kubernetes:

Initialize your Kubernetes cluster with the following command:

bash

Copy code

```
sudo kubeadm init --pod-network-cidr=10.244.0.0/16
--ignore-preflight-errors=NumCPU,Mem
```

```
[root@ip-172-31-22-81 ec2-user]# sudo kubeadm init --pod-network-cidr=10.244.0.0/16
--ignore-preflight-errors=all
I0915 07:56:11.461747 30220 version.go:256] remote version is much newer: v1.31.0; falling back to: stable-1.30
[init] Using Kubernetes version: v1.30.4
[preflight] Running pre-flight checks
[W0915 07:56:11.697643 30220 checks.go:844] detected that the sandbox image "registry.k8s.io/pause:3.9" of the containerd
ended to use "registry.k8s.io/pause:3.9" 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-172-31-22-81.ec2.internal kubernetes kubernetes.default kub
[10.96.0.1 172.31.22.81]
[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-172-31-22-81.ec2.internal localhost] and IPs [172.31.22.81
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-22-81.ec2.internal localhost] and IPs [172.31.22.81
[certs] Generating "etcd/healthcheck-client" certificate and key
```

**Configure `kubectl` Access:**

To set up `kubectl` for your non-root user, run:

bash

Copy code

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

```
[root@ip-172-31-22-81 ec2-user]# mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
[root@ip-172-31-22-81 ec2-user]#
```

To check whether nodes are connected, run the command

- `kubectl get nodes`

This output shows only master is connected right now.

**Kubectl get nodes**

```
[root@ip-172-31-22-81 ec2-user]# kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
ip-172-31-22-81.ec2.internal	NotReady	control-plane	4m52s	v1.30.5

## Perform this Only on node Machines

```
[root@ip-172-31-22-81 ec2-user]# sudo yum install socat -y
Last metadata expiration check: 0:19:33 ago on Sun Sep 15 07:47:07 2024.
Dependencies resolved.
=====
Package                Architecture      Version           Repository        Size
-----
Installing:
socat                  x86_64            1.7.4.2-1.amzn2023.0.2  amazonlinux        303 k
Transaction Summary
-----
Install 1 Package

Total download size: 303 k
Installed size: 1.1 M
Downloading Packages:
socat-1.7.4.2-1.amzn2023.0.2.x86_64.rpm                2.4 MB/s | 303 kB    00:00
=====
```

This is the token which we got

```
172.31.22.81:6443 --token 9wf1pg.1xgp88wof0wpslc9 \
--discovery-token-ca-cert-hash
```

```
sha256:b4efc86172e4999d3d1e530147cc26705f6543cd48689416874811c98b2
a325f
```

Put this command on the node machine to connect

Now go Back to the master machine and write 'kubectl get nodes'

```
[root@ip-172-31-22-81 docker]# kubectl get nodes
NAME                                STATUS    ROLES    AGE     VERSION
ip-172-31-22-81.ec2.internal        Ready     control-plane  4m13s   v1.30.5
ip-172-31-28-157.ec2.internal        Not Ready  computer 1    2m24s   v1.30.5
ip-172-31-23-208.ec2.internal        Not Ready  computer 2    2m17s   v1.30.5
```

We will see nodes are <NOT READY> to change it install a CNI plugin on Master Machine

kubectl apply -f

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

Now 'Run kubectl get nodes'

```
[root@ip-172-31-22-81 docker]# kubectl get nodes
NAME                                STATUS    ROLES    AGE     VERSION
ip-172-31-22-81.ec2.internal        Ready     control-plane  9m33s   v1.30.5
ip-172-31-28-157.ec2.internal        Ready     computer 1    7m24s   v1.30.5
ip-172-31-23-208.ec2.internal        Ready     computer 2    7m17s   v1.30.5
```

## **Conclusion**

In this experiment, we explored the architecture of a Kubernetes cluster and successfully deployed it on AWS EC2 instances, consisting of a master node and two worker nodes. After setting up Docker, Kubernetes components (kubelet, kubeadm, kubectl), and containerd across all nodes, the cluster was initialized by configuring the master node and joining the worker nodes. Initially, the nodes showed a "NotReady" status, which was corrected by implementing the Calico networking solution. Additionally, nodes were labeled appropriately based on their roles. By the end of the process, all nodes were fully operational, confirming that the Kubernetes cluster was configured correctly .