

EXPERIMENT. 3

Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.

Steps:

Create 3 EC2 Ubuntu Instances on AWS.

Launch an instance [Info](#)

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.

Name and tags [Info](#)

Name: [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.

Quick Start

Amazon Linux macOS Ubuntu Windows Red Hat SUSE L

Summary

Number of instances: [Info](#)

Software Image (AMI)
Canonical, Ubuntu, 22.04 LTS, ...[read more](#)
ami-0c2af51e265bd5e0e

Virtual server type (instance type)
t2.medium

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month; 750 hours of public IPv4 address usage per month.

[Cancel](#) [Launch instance](#) [Review commands](#)

Created a master and 2 slaves

Now click on connect to instance, then click on SSH client

Instances (1/3) [Info](#)

Last updated less than a minute ago

[All states](#)

[Instance state = running](#) [Clear filters](#)

| | Name | Instance ID | Instance state | Instance type | Status check | Alarm status | Availability Zone | Public IPv4 DNS | Public IP |
|-------------------------------------|----------|---------------------|----------------|---------------|-------------------|-----------------------------|-------------------|--------------------------|-----------|
| <input type="checkbox"/> | Worker-2 | i-0e3930ceb2d892d01 | Running | t2.medium | 2/2 checks passed | View alarms | ap-south-1a | ec2-13-234-226-219.ap... | 13.234.2 |
| <input type="checkbox"/> | Worker-1 | i-0d16e01d1824e0e3a | Running | t2.medium | 2/2 checks passed | View alarms | ap-south-1a | ec2-65-0-104-95.ap-so... | 65.0.104 |
| <input checked="" type="checkbox"/> | Master | i-01ae3d58db90ad73 | Running | t2.medium | 2/2 checks passed | View alarms | ap-south-1a | ec2-13-252-36-34.ap-s... | 13.252.3 |

Shreya Sawant D15A -54

Now copy the ssh from the example and paste it on command prompt.

[EC2](#) > [Instances](#) > [i-02d0bd51d43449e29](#) > [Connect to instance](#)

Connect to instance Info

Connect to your instance i-02d0bd51d43449e29 (Master) using any of these options

EC2 Instance Connect

Session Manager

SSH client

EC2 serial console

Instance ID
i-02d0bd51d43449e29 (Master)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is `kubernetes.pem`
3. Run this command, if necessary, to ensure your key is not publicly viewable.
`chmod 400 "kubernetes.pem"`
4. Connect to your instance using its Public DNS:
`ec2-54-164-13-87.compute-1.amazonaws.com`

Example:

```
ssh -i "kubernetes.pem" ubuntu@ec2-54-164-13-87.compute-1.amazonaws.com
```

Commands:

After this type on all 3 machines

Yum install docker -y

```
ec2-user@ip-172-31-84-37 ~]$ sudo su
[root@ip-172-31-84-37 ec2-user]# yum install docker -y
Last metadata expiration check: 0:18:22 ago on Thu Aug 29 08:52:52 2024.
Dependencies resolved.
```

| Package | Architecture | Version | Repository | Size |
|--------------------------|--------------|-----------------------|-------------|-------|
| Installing: | | | | |
| docker | x86_64 | 25.0.6-1.amzn2023.0.1 | 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 |
| libnftnl | 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 |
| pkgconf | x86_64 | 2.5-1.amzn2023.0.3 | amazonlinux | 83 k |
| runo | x86_64 | 1.1.11-1.amzn2023.0.1 | amazonlinux | 3.0 M |

```
Running scriptlet: docker-25.0.6-1.amzn2023.0.1.x86_64
Installing      : docker-25.0.6-1.amzn2023.0.1.x86_64
Running scriptlet: docker-25.0.6-1.amzn2023.0.1.x86_64
Created 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
Verifying      : docker-25.0.6-1.amzn2023.0.1.x86_64
Verifying      : iptables-libs-1.8.8-3.amzn2023.0.2.x86_64
Verifying      : iptables-nft-1.8.8-3.amzn2023.0.2.x86_64
Verifying      : libcgroupp-3.0-1.amzn2023.0.1.x86_64
Verifying      : libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64
Verifying      : libnftnl-1.0.1-19.amzn2023.0.2.x86_64
Verifying      : libnftnl-1.2.2-2.amzn2023.0.2.x86_64
Verifying      : pigz-2.5-1.amzn2023.0.3.x86_64
Verifying      : runc-1.1.11-1.amzn2023.0.1.x86_64

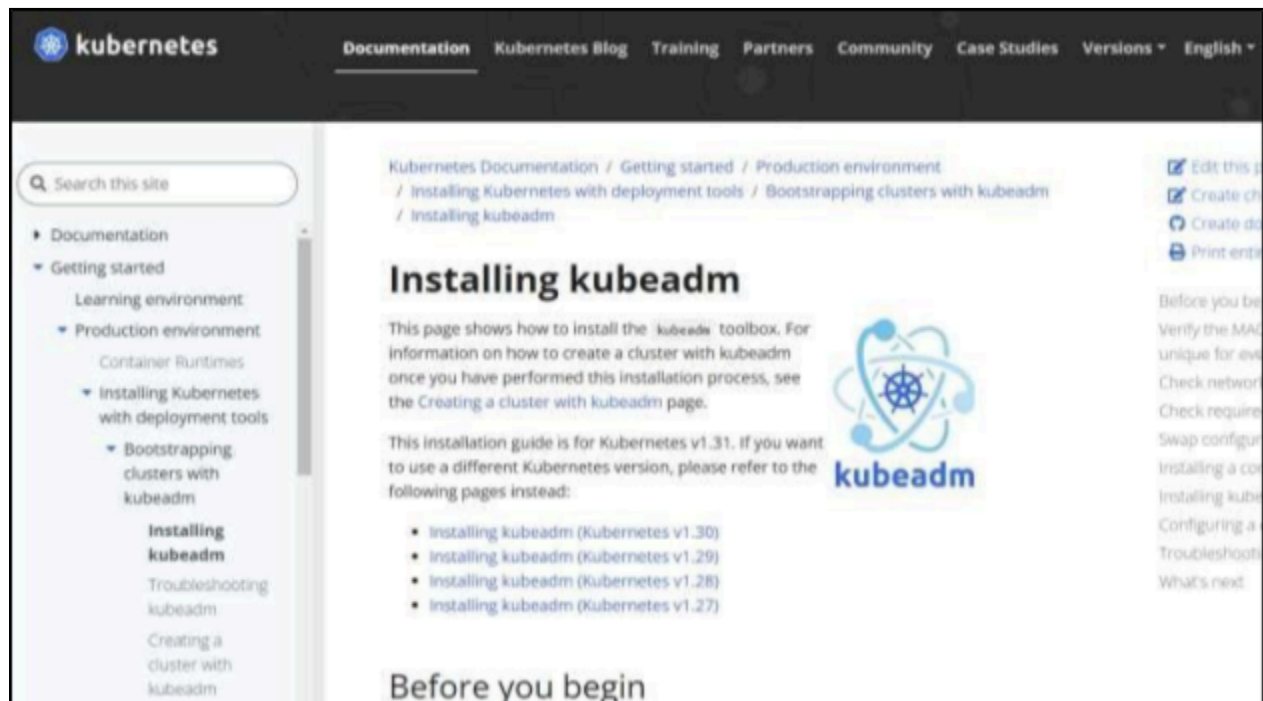
Installed:
  containerd-1.7.20-1.amzn2023.0.1.x86_64    docker-25.0.6-1.amzn2023.0.1.x86_64    iptables-libs-1.8.8-3.amzn2023.0.2.x86_64
  iptables-nft-1.8.8-3.amzn2023.0.2.x86_64    libcgroupp-3.0-1.amzn2023.0.1.x86_64    libnetfilter_conntrack-1.0.8-2.amzn2023
  libnftnl-1.0.1-19.amzn2023.0.2.x86_64    libnftnl-1.2.2-2.amzn2023.0.2.x86_64    pigz-2.5-1.amzn2023.0.3.x86_64
  runc-1.1.11-1.amzn2023.0.1.x86_64

Complete!
```

To start the docker on master and slave perform this command :Systemctl start docker
To check if docker is installed successfully: Docker -v or Docker --version

```
[root@ip-172-31-84-37 ec2-user]# systemctl start docker
[root@ip-172-31-84-37 ec2-user]# sudo su
[root@ip-172-31-84-37 ec2-user]# yum repolist
repo id                                repo name
amazonlinux                            Amazon Linux 2023 repository
kernel-livepatch                       Amazon Linux 2023 Kernel Livepatch repository
[root@ip-172-31-84-37 ec2-user]# docker --version
Docker version 25.0.5, build 5dc9bcc
```

Now install kubeadm on master and slaves:



Scroll down and select Red Hat based distributions

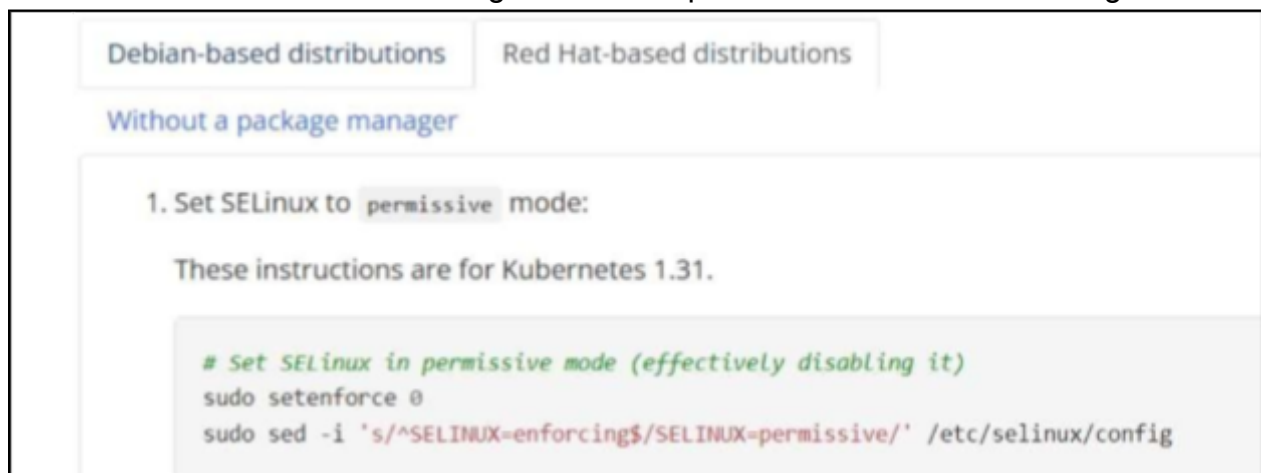
Now copy the command on all 3 machines:

Set SELinux to permissive mode: These instructions are for Kubernetes

Set SELinux in permissive mode (effectively disabling it)

```
sudo setenforce 0
```

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



Now copy all the commands on the GitBash on all 3 machines

```
Installing      : kubect1-1.31.0-150500.1.1.x86_64
Running scriptlet: kubect1-1.31.0-150500.1.1.x86_64
Verifying       : conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64
Verifying       : libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64
Verifying       : libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64
Verifying       : libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64
Verifying       : socat-1.7.4.2-1.amzn2023.0.2.x86_64
Verifying       : cri-tools-1.31.1-150500.1.1.x86_64
Verifying       : kubeadm-1.31.0-150500.1.1.x86_64
Verifying       : kubect1-1.31.0-150500.1.1.x86_64
Verifying       : kubelet-1.31.0-150500.1.1.x86_64
Verifying       : kubernetes-cni-1.5.0-150500.2.1.x86_64

Installed:
conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64
kubeadm-1.31.0-150500.1.1.x86_64
kubelet-1.31.0-150500.1.1.x86_64
libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64
libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64

Complete!
```

Type yum repolist to check the repository of kubernetes

Shreya Sawant D15A -54

```
[root@ip-172-31-84-143 ec2-user]# yum repolist
```

| repo id | repo name |
|------------------|---|
| amazonlinux | Amazon Linux 2023 repository |
| kernel-livepatch | Amazon Linux 2023 Kernel Livepatch repository |
| kubernetes | Kubernetes |

Copy them one by one and
paste it on slaves

```
To start using your cluster, you need to run the following as a regular user:
```

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

Alternatively, if you are the root user, you can run:

```
export KUBECONFIG=/etc/kubernetes/admin.conf
```

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
<https://kubernetes.io/docs/concepts/cluster-administration/addons/>

Then you can join any number of worker nodes by running the following on each as root:

```
kubeadm join 172.31.26.66:6443 --token qrw4r4.qb3kkhb7392dnvjp \
```

After pasting type kubectl get nodes:

The nodes are connected successfully

```
ubuntu@ip-172-31-45-227:~$ kubectl get nodes
```

| NAME | STATUS | ROLES | AGE | VERSION |
|------------------|--------|---------------|-------|---------|
| ip-172-31-43-211 | Ready | <none> | 50s | v1.29.0 |
| ip-172-31-45-13 | Ready | <none> | 34s | v1.29.0 |
| ip-172-31-45-227 | Ready | control-plane | 5m17s | v1.29.0 |