

Name : Devansh Wadhwani

Div : D15A

Roll No. : 64

Advanced DevOps Lab

Experiment:3

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

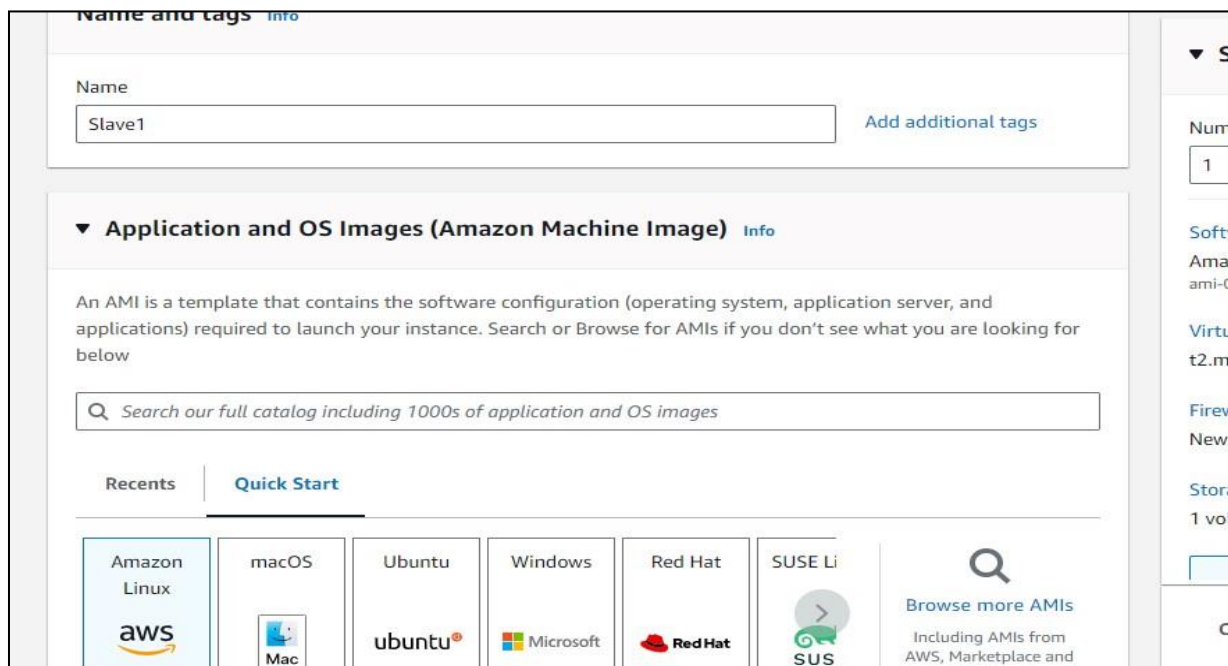
Theory:

To understand Kubernetes Cluster Architecture and how to install and spin up a Kubernetes cluster on Linux machines or cloud platforms, it's essential to grasp the fundamental components and design principles of Kubernetes.

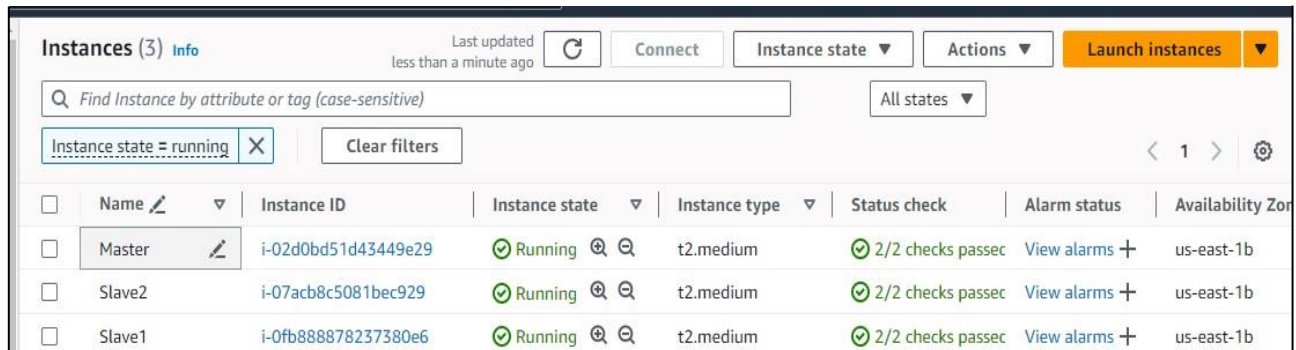
Steps:

1. Create 3 EC2 Ubuntu Instances on AWS.

(Name 1 as Master, the other 2 as Slave1 and Slave2)



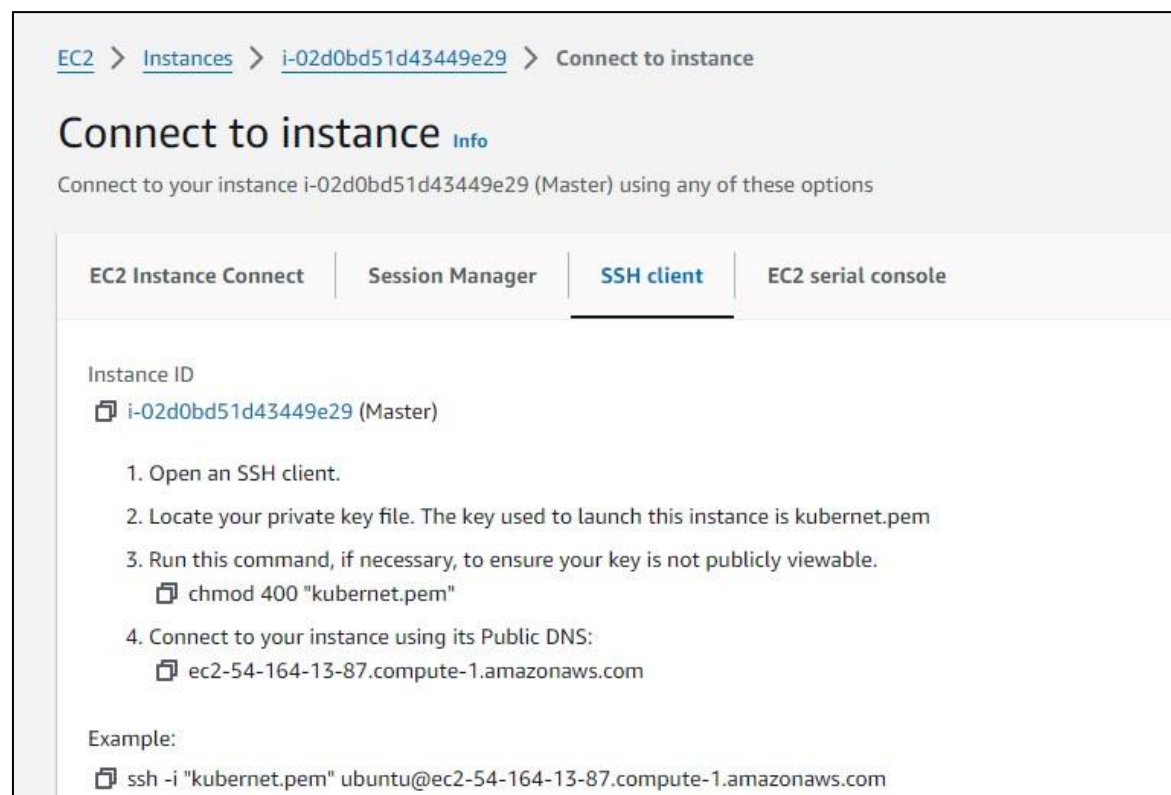
1. Created a master and 2 slaves:



	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
<input type="checkbox"/>	Master	i-02d0bd51d43449e29	Running	t2.medium	2/2 checks passed	View alarms +	us-east-1b
<input type="checkbox"/>	Slave2	i-07acb8c5081bec929	Running	t2.medium	2/2 checks passed	View alarms +	us-east-1b
<input type="checkbox"/>	Slave1	i-0fb888878237380e6	Running	t2.medium	2/2 checks passed	View alarms +	us-east-1b

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

3. Now copy the ssh from the example and paste it on command prompt.(I used gitbash)



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:

4. Now since you are on GitBash, first type `sudo su` to perform the command as a root user.

5. 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
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	1.1.11-1.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      : libcgroup-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    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-1.1.11-1.amzn2023.0.1.x86_64

Complete!
```

6. To start the docker on master and slave perform this command: `Systemctl start docker`

Extra

7. To check if docker is Installed successfully:

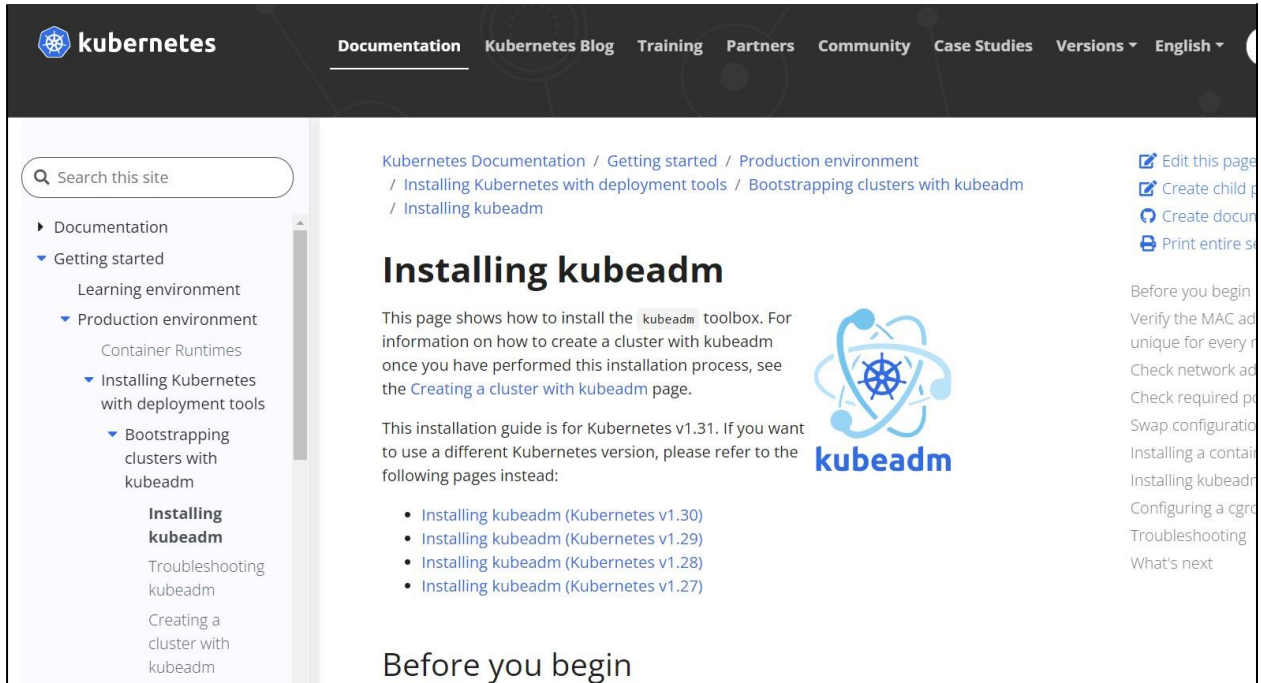
`Docker -v` or `Docker --version`

```
[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
```

8. Now to install kubeadm on master and slaves :

Installing kubeadm:

Go the official documentation off kubeadm.



The screenshot shows the Kubernetes documentation website. The top navigation bar includes links for Documentation, Kubernetes Blog, Training, Partners, Community, Case Studies, Versions, and English. A search bar is located on the left. The left sidebar shows a navigation menu with categories like Documentation, Getting started, Learning environment, Production environment, Container Runtimes, Installing Kubernetes with deployment tools, and Bootstrapping clusters with kubeadm. The main content area is titled "Installing kubeadm" and includes a breadcrumb trail: "Kubernetes Documentation / Getting started / Production environment / Installing Kubernetes with deployment tools / Bootstrapping clusters with kubeadm / Installing kubeadm". The text explains that the page shows how to install the kubeadm toolbox and provides a list of links for different Kubernetes versions (v1.30, v1.29, v1.28, v1.27). A "Before you begin" section is also visible.

Kubernetes Documentation / Getting started / Production environment / Installing Kubernetes with deployment tools / Bootstrapping clusters with kubeadm / Installing kubeadm

Installing kubeadm

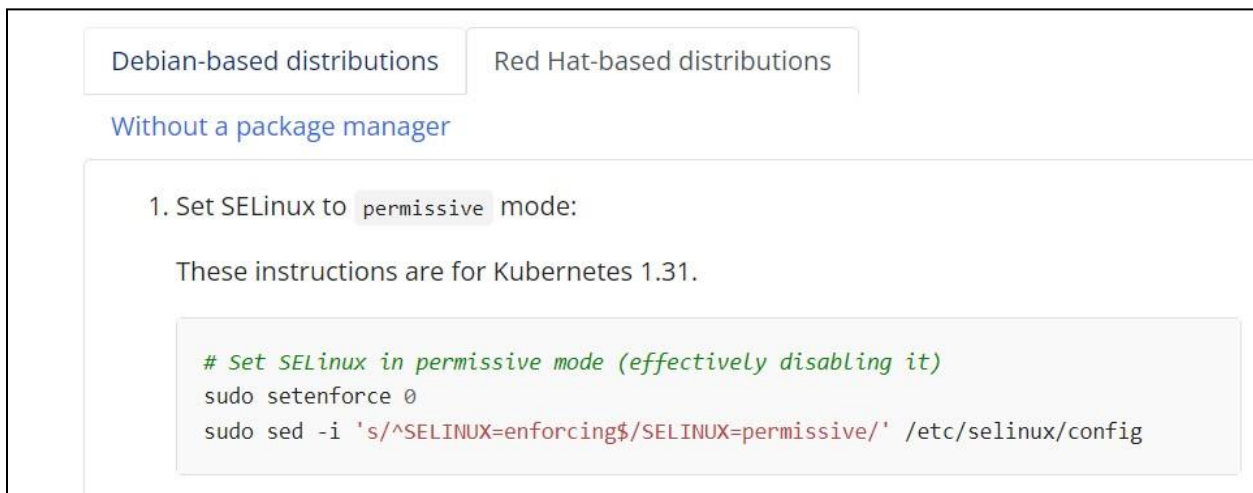
This page shows how to install the `kubeadm` toolbox. For information on how to create a cluster with kubeadm once you have performed this installation process, see the [Creating a cluster with kubeadm](#) page.

This installation guide is for Kubernetes v1.31. If you want to use a different Kubernetes version, please refer to the following pages instead:

- [Installing kubeadm \(Kubernetes v1.30\)](#)
- [Installing kubeadm \(Kubernetes v1.29\)](#)
- [Installing kubeadm \(Kubernetes v1.28\)](#)
- [Installing kubeadm \(Kubernetes v1.27\)](#)

Before you begin

9. Scroll down and select Red Hat based distributions:



The screenshot shows the "Red Hat-based distributions" section of the kubeadm documentation. It includes a tab for "Without a package manager" and a list of instructions for setting SELinux to permissive mode. The instructions are for Kubernetes 1.31 and include a code block with the following commands:

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

10. Now copy the command on all 3 machines:

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
```

11. Now copy all the commands on the GitBash on all the 3 machines:

```
# This overwrites any existing configuration in /etc/yum.repos.d/kubernetes.repo 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/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
```

```
#Install kubelet, kubeadm and kubectl: sudo yum install -y kubelet kubeadm
```

```
kubectl --disableexcludes=kubernetes #(Optional) Enable the kubelet service
```

before running kubeadm:

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

```
Installing      : kubeadm-1.31.0-150500.1.1.x86_64
Installing      : kubectl-1.31.0-150500.1.1.x86_64
Running scriptlet: kubectl-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       : kubectl-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!
[root@ip-172-31-84-37 ec2-user]# sudo systemctl enable --now kubelet
```

12. Type yum repolist to check the repository of kubernetes

```
[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
```

EXTRA

Got an error in initialization kubeadm

```
[root@ip-172-31-31-240 ec2-user]# kubeadm init
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
W0908 11:25:45.820964    2320 checks.go:1080] [preflight] WARNING: Couldn't create the interface used for talking to
CRI runtime service: validate service connection: validate CRI v1 runtime API for endpoint "unix:///var/run/contain
ple desc = connection error: desc = "transport: Error while dialing: dial unix /var/run/containerd/containerd.sock:
[WARNING FileExisting-tc]: tc not found in system path
error execution phase preflight: [preflight] Some fatal errors occurred:
[ERROR FileContent--proc-sys-net-ipv4-ip_forward]: /proc/sys/net/ipv4/ip_forward contents are not set to 1
[preflight] 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
```

Error was resolved:

(after again starting from scratch)

13. Initialize the kubeadm by the command kubeadm init only on master:

Kubeadm initialized successfully:

```
[root@ip-172-31-26-66 ec2-user]# kubeadm init
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
[WARNING FileExisting-socat]: socat not found in system path
[WARNING FileExisting-tc]: tc 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
[preflight] You can also perform this action beforehand using 'kubeadm config images pull'
W0912 06:07:49.475553    28037 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.10" as the
that used by kubeadm. It is recommended to use "registry.k8s.io/pause:3.10" as the
[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-172-31-26-66.ec2.internal
efault.svc.cluster.local] and IPs [10.96.0.1 172.31.26.66]
[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
```

14. After this we will get 3 things:

- The directory
- Some export Statement
- The most important thing - the token to connect the slaves with the master.

15. Copy them one by one and paste it on the 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 grw4r4.gb3kkhb7392dnvjp \
--discovery-token-ca-cert-hash sha256:b61f1de7eedb2c0dc0cc237d4629e9631920b63dd6634c3e22e76aaa36d01920
```

16. After pasting type kubectl get nodes:

The nodes are connected successfully:

```
ubuntu@ip-172-31-17-23:~$ kubectl get nodes
NAME                 STATUS    ROLES    AGE   VERSION
ip-172-31-17-23      Ready     control-plane   3m56s   v1.29.0
ip-172-31-18-12      Ready     <none>         37s     v1.29.0
ip-172-31-26-153     Ready     <none>         24s     v1.29.0
ubuntu@ip-172-31-17-23:~$ kubectl get nodes
NAME                 STATUS    ROLES    AGE   VERSION
ip-172-31-17-23      Ready     control-plane   9m34s   v1.29.0
ip-172-31-18-12      Ready     <none>         6m15s   v1.29.0
ip-172-31-26-153     Ready     <none>         6m2s    v1.29.0
ubuntu@ip-172-31-17-23:~$ |
```