

ADVANCE DEVOPS EXP 3

Name :- Swaraj Patil

Roll no :- 40

Aim:-

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

Step 1: Pre-requisites

1.1 Create 3 EC2 instances, one for the master node and two for the worker nodes.

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 Linux

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](#)

Create key pair [X](#)

Key pair name

Key pairs allow you to connect to your instance securely.

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

☒ RSA
RSA encrypted private and public key pair

☐ ED25519
ED25519 encrypted private and public key pair

Private key file format

☒ .pem
For use with OpenSSH

☐ .ppk
For use with PuTTY

When prompted, store the private key in a secure and accessible location on

[Cancel](#) [Create key pair](#)

Create 3 EC2 Ubuntu Instances of Ubuntu version 20.04 and keep all the instances in the same security group on AWS. (Name 1 as Master, the other 2 as worker-1 and worker-2)

Instances (3) info							
<input type="text" value="Find instance by attribute or tag (case-sensitive)"/>							
<input type="button" value="Instance state = running"/> <input type="button" value="Clear filters"/>							
<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
<input type="checkbox"/>	Worker2	i-0454af7c775a97b9b	Running	t2.micro	2/2 checks passed	No alarms	us-east-1c
<input type="checkbox"/>	Worker-1	i-031892fec4e1152a7	Running	t2.micro	2/2 checks passed	No alarms	us-east-1c
<input type="checkbox"/>	Master	i-02750ab2198a11dca	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b

Now the ssh created, copy the text given in the example

EC2 Instance Connect

Session Manager

SSH client

EC2 serial console

Instance ID

i-Oe3930ceb2d892d01 (Worker-2)

1. Open an SSH client.

2. Locate your private key file. The key used to launch this instance is two-tier-app-k8s.pem

3. Run this command, if necessary, to ensure your key is not publicly viewable.

chmod 400 "two-tier-app-k8s.pem"

4. Connect to your instance using its Public DNS:

ec2-13-234-226-219.ap-south-1.compute.amazonaws.com

Example:

ssh -i "two-tier-app-k8s.pem" ubuntu@ec2-13-234-226-219.ap-south-1.compute.amazonaws.com

acer@TMP214-53 MINGW64 ~/Downloads

\$ ssh -i "two-tier-app-k8s.pem" ubuntu@ec2-13-232-36-34.ap-south-1.compute.amazonaws.com

The authenticity of host 'ec2-13-232-36-34.ap-south-1.compute.amazonaws.com (13.232.36.34)' can't be established.

ED25519 key fingerprint is SHA256:uVGEO+FWYefj60j0ft70Sralv8NrzEi/IwxAtBY+EPE.

This key is not known by any other names.

Are you sure you want to continue connecting (yes/no/[fingerprint])? yes

Warning: Permanently added 'ec2-13-232-36-34.ap-south-1.compute.amazonaws.com' (ED25519) to the list of known hosts.

Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-1022-aws x86_64)

* Documentation: <https://help.ubuntu.com>

* Management: <https://landscape.canonical.com>

* Support: <https://ubuntu.com/pro>

System information as of Wed Sep 11 14:07:10 UTC 2024

System load: 0.0

Usage of /: 20.7% of 7.57GB

Memory usage: 5%

Swap usage: 0%

Processes: 106

Users logged in: 0

IPv4 address for eth0: 172.31.45.227

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`

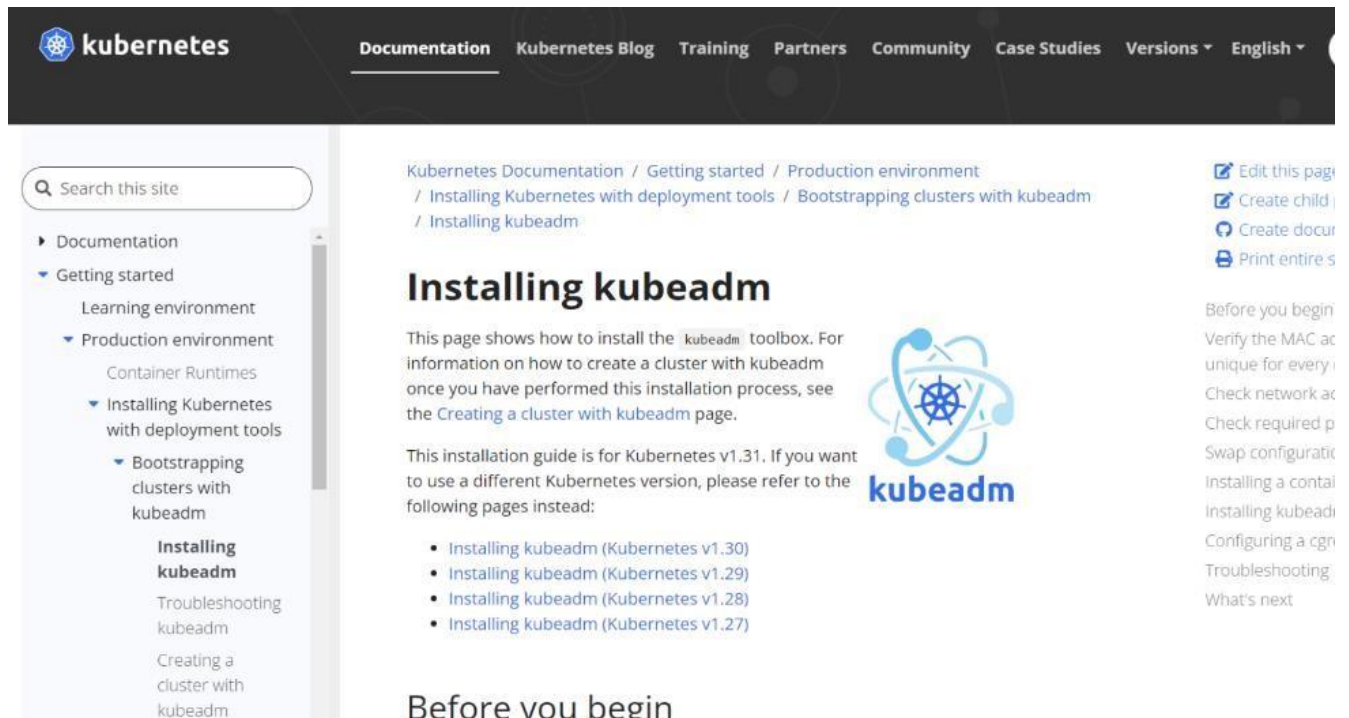
```
ubuntu@ip-172-31-81-188:~$ docker --version
Docker version 20.10.12, build 20.10.12-0ubuntu2~20.04.1
ubuntu@ip-172-31-81-188:~$
```

```
aws Services Search for services, features, blogs, docs, and more
ubuntu@ip-172-31-23-53:~$ docker --version
Docker version 20.10.12, build 20.10.12-0ubuntu2~20.04.1
ubuntu@ip-172-31-23-53:~$
```

```
ubuntu@ip-172-31-21-143:~$ docker --version
Docker version 20.10.12, build 20.10.12-0ubuntu2~20.04.1
ubuntu@ip-172-31-21-143:~$
```

Kubernetes Installation

Go to official documentation off kubedam



The screenshot shows the Kubernetes official documentation website. The top navigation bar includes the Kubernetes logo, "Documentation", "Kubernetes Blog", "Training", "Partners", "Community", "Case Studies", "Versions", and "English". A search bar is on the left. 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 installing kubeadm on different Kubernetes versions (v1.30, v1.29, v1.28, v1.27). A sidebar on the left lists the navigation menu, and a right sidebar contains links to edit the page, create child pages, create documentation, and print the entire site. A "Before you begin" section is also visible on the right.

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

1. \$sudo apt-get install kubeadm kubelet kubectl -y

```
ubuntu@ip-172-31-81-188:~$ sudo apt-get install kubeadm kubelet kubectl -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools ebtables kubernetes-cni socat
Suggested packages:
  nftables
The following NEW packages will be installed:
  conntrack cri-tools ebtables kubeadm kubectl kubelet kubernetes-cni socat
0 upgraded, 8 newly installed, 0 to remove and 62 not upgraded.
Need to get 75.9 MB of archives.
After this operation, 310 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal/main amd64 conntrack amd64 1:1.4.5-2
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal/main amd64 ebtables amd64 2.0.11-3bui
```

2. Verify the installation with

```
ubuntu@ip-172-31-23-53:~$ kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"25", GitVersion:"v1.25.0", GitCommit:"a866cbe2e5bba01cfd5e969aa3e033f3282a8a2", GitTreeState:"clean", BuildDate:"2022-08-23T17:43:25Z", GoVersion:"go1.19", Compiler:"gc", Platform:"linux/amd64"}
ubuntu@ip-172-31-23-53:~$
```

3. \$sudo swapoff -a

```
ubuntu@ip-172-31-22-29:~$ sudo swapoff -a
sudo sed -i '/ swap / s/^/#/' /etc/fstab
```

4. \$sudo hostnamectl set-hostname

```
ubuntu@ip-172-31-23-53:~$ sudo hostnamectl set-hostname worker1
ubuntu@ip-172-31-23-53:~$
```

```
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
sudo sysctl—system
```



```
ubuntu@ip-172-31-81-188:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=all
[init] Using Kubernetes version: v1.25.0
[preflight] Running pre-flight checks
        [WARNING NumCPU]: the number of available CPUs 1 is less than the required 2
        [WARNING Mem]: the system RAM (967 MB) is less than the minimum 1700 MB
[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 in beforehand using 'kubeadm config images pull'
[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 [kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local ip-172-31-81-188]
```

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 172.31.81.188:6443 --token n46tzy.ocnrf7wkiyk0t0xu \
--discovery-token-ca-cert-hash sha256:59c2fec9fc69aa85d306f8bfcadac2d827699b0db3d87e13192873a1044f86e2
ubuntu@ip-172-31-81-188:~$
```

Deploy Pod Network to Cluster A Pod Network is a way to allow communication between different nodes in the cluster. This tutorial uses the flannel virtual network.

```
ubuntu@ip-172-31-81-188:~$ sudo kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
The connection to the server localhost:8080 was refused - did you specify the right host or port?
ubuntu@ip-172-31-81-188:~$ 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
ubuntu@ip-172-31-81-188:~$
```

Join Worker Nodes to the Cluster On the worker nodes, run the command provided by the master node during initialization . It looks something like this: sudo kubeadm join :6443--token --discovery-token-ca-cert-hash sha256:

```
oot@worker1:~$ kubeadm join 172.31.81.188:6443 --token n46tzy.ocnrf7wkiyk0t0xu --discovery-token-ca-cert-hash sha256:59c2fec9fc69aa85d306f8bfcadac2d827699b0db3d87e13192873a1044f86e2 --ignore-preflight-errors=all
[preflight] Running pre-flight checks
error execution phase preflight: couldn't validate the identity of the API Server: Get "https://172.31.81.188:6443/api/v1/namespaces/kube-public/configmaps/cluster-info?timeout=10s": net/http: request canceled while waiting for connection (Client.Timeout exceeded while awaiting headers)
to see the stack trace of this error execute with --v=5 or higher
oot@worker1:~$ kubeadm join 172.31.81.188:6443 --token n46tzy.ocnrf7wkiyk0t0xu --discovery-token-ca-cert-hash sha256:59c2fec9fc69aa85d306f8bfcadac2d827699b0db3d87e13192873a1044f86e2 --ignore-preflight-errors=all
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
Certificate signing request was sent to apiserver and a response was received.
The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

Verify the Cluster

```
Using cluster from kubectl context: workshop.k8s.local
```

```
Validating cluster workshop.k8s.local
```

INSTANCE GROUPS

NAME	ROLE	MACHINETYPE	MIN	MAX	SUBNETS
master-us-west-2a	Master	t3.medium	1	1	us-west-2a
nodes-us-west-2a	Node	t3.medium	1	1	us-west-2a

NODE STATUS

NAME	ROLE	READY
ip-172-20-40-55.us-west-2.compute.internal	master	True
ip-172-20-58-174.us-west-2.compute.internal	node	True

```
Your cluster workshop.k8s.local is ready
```