Advanced DevOps Lab
Experiment:3

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

Reference: https://www.youtube.com/watch?v=Cz7hSJNq2GU

Theory:

Container-based microservices architectures have profoundly changed the way development and operations teams test and deploy modern software. Containers help companies modernize by making it easier to scale and deploy applications, but containers have also introduced new challenges and more complexity by creating an entirely new infrastructure ecosystem.

Large and small software companies alike are now deploying thousands of container instances daily, and that's a complexity of scale they have to manage. So how do they do it?

Enter the age of Kubernetes.

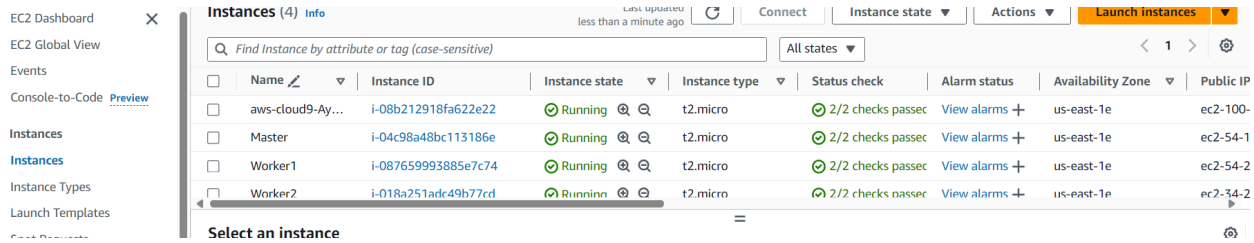
Originally developed by Google, Kubernetes is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications. In fact, Kubernetes has established itself as the defacto standard for container orchestration and is the flagship project of the Cloud Native Computing Foundation (CNCF), backed by key players like Google, AWS, Microsoft, IBM, Intel, Cisco, and Red Hat.

Kubernetes makes it easy to deploy and operate applications in a microservice architecture. It does so by creating an abstraction layer on top of a group of hosts so that development teams can deploy their applications and let Kubernetes manage the following activities:

- Controlling resource consumption by application or team
- Evenly spreading application load across a hosting infrastructure
- Automatically load balancing requests across the different instances of an application
- Monitoring resource consumption and resource limits to automatically stop applications from consuming too many resources and restarting the applications again
- Moving an application instance from one host to another if there is a shortage of resources in a host, or if the host dies
- Automatically leveraging additional resources made available when a new host is added to the cluster
- Easily performing canary deployments and rollbacks

Steps:

1. Create 3 EC2 Ubuntu Instances on AWS.
(Name 1 as Master, the other 2 as w_1 and w_2)



3. SSH into all 3 machines

ssh -i <keyname>.pem ubuntu@<public_ip_address>
(While writing use same public ip as given)

```
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


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-52-126:~$ ssh -i adv_devops_exp3.pem ubuntu@54.157.150.221
Warning: Identity file adv_devops_exp3.pem not accessible: No such file or directory.
The authenticity of host '54.157.150.221 (54.157.150.221)' can't be established.
ED25519 key fingerprint is SHA256:zqB5bOKb3rhGMg0SHg/38ph1Wr8UQnHCktN2cbmNsrU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '54.157.150.221' (ED25519) to the list of known hosts.
Connection closed by 54.157.150.221 port 22
ubuntu@ip-172-31-52-126:~$
```

4. From now on, until mentioned, perform these steps on all 3 machines.

Install Docker

→ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
→ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
→ sudo apt-get update
→ sudo apt-get install -y docker-ce

```
ubuntu@ip-172-31-52-126:~$ 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-172-31-52-126:~$
```

i-095c8528e3d57c5ca (master)
PublicIPs: 54.157.150.221 PrivateIPs: 172.31.52.126

```
ubuntu@ip-172-31-52-126:~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
Repository: 'deb [arch=amd64] https://download.docker.com/linux/ubuntu noble stable'
Description:
Archive for codename: noble components: stable
More info: https://download.docker.com/linux/ubuntu
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
Adding deb entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:5 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
Get:6 https://download.docker.com/linux/ubuntu noble/stable amd64 Packages [12.4 kB]
Fetched 61.2 kB in 1s (76.4 kB/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION sectio
n in apt-key(8) for details.
ubuntu@ip-172-31-52-126:~$
```

```
n in apt-key(8) for details.
ubuntu@ip-172-31-52-126:~$ 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: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION sectio
n in apt-key(8) for details.
```

```
ubuntu@ip-172-31-52-126:~$ 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 83 not upgraded.
Need to get 122 MB of archives.
After this operation, 437 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 1.7.20-1 [30.5 MB]
Get:6 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-buildx-plugin amd64 0.16.2-1~ubuntu.24.04~noble [29.9 MB]
Get:7 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-ce-cli amd64 5:27.1.2-1~ubuntu.24.04~noble [14.6 MB]
Get:8 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-ce amd64 5:27.1.2-1~ubuntu.24.04~noble [25.2 MB]
Get:9 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-ce-rootless-extras amd64 5:27.1.2-1~ubuntu.24.04~noble [9318 kB]
Get:10 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-compose-plugin amd64 2.29.1-1~ubuntu.24.04~noble [12.5 MB]
Fetched 122 MB in 2s (59.1 MB/s)
Selecting previously unselected package pigz.
(Reading database ... 67741 files and directories currently installed.)
```

```
Setting up docker-ce-rootless-extras (5:27.1.2-1~ubuntu.24.04~noble) ...
Setting up slirp4netns (1.2.1-1build2) ...
Setting up docker-ce (5:27.1.2-1~ubuntu.24.04~noble) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
```

```
ubuntu@ip-172-31-52-126:~$ docker --version
Docker version 27.1.2, build d01f264
ubuntu@ip-172-31-52-126:~$
```

Then, configure cgroup in a daemon.json file.

→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
→ sudo systemctl enable docker
→ sudo systemctl daemon-reload
→ sudo systemctl restart docker

```
ubuntu@ip-172-31-85-18:~$ cd /etc/docker
ubuntu@ip-172-31-85-18:/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
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
ubuntu@ip-172-31-85-18:/etc/docker$
```

```
ubuntu@ip-172-31-85-76:/etc/docker$ 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-172-31-85-76:/etc/docker$ sudo systemctl daemon-reload
sudo systemctl restart docker
docker -v
Docker version 27.2.1, build 9e34c9b
ubuntu@ip-172-31-85-76:/etc/docker$
```

Install Kubernetes on all 3 machines

→ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/k8s.gpg
→ echo 'deb [signed-by=/etc/apt/keyrings/k8s.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /' | sudo tee /etc/apt/sources.list.d/k8s.list
→ sudo apt-get update
→ sudo apt-get install kubelet kubeadm kubectl -y

```
ubuntu@ip-172-31-85-76:~$ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/k8s.gpg
File '/etc/apt/keyrings/k8s.gpg' exists. Overwrite? (y/N) y
ubuntu@ip-172-31-85-76:~$ echo 'deb [signed-by=/etc/apt/keyrings/k8s.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /' | sudo tee /etc/apt/sou
list
deb [signed-by=/etc/apt/keyrings/k8s.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /
ubuntu@ip-172-31-85-76:~$ sudo apt update
sudo apt install kubelet kubeadm kubectl -y
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
Ign:7 https://packages.cloud.google.com/apt kubernetes-ubuntu-24.04 InRelease
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb  InRelease [1186 B]
Err:8 https://packages.cloud.google.com/apt kubernetes-ubuntu-24.04 Release
  404  Not Found [IP: 172.253.62.102 443]
Get:9 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb  Packages [9318 B]
Reading package lists... Done
E: The repository 'https://packages.cloud.google.com/apt kubernetes-ubuntu-24.04 Release' does not have a Release file.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
```

(shows installed kubelet kubeadm kubectl)

```
ubuntu@ip-172-31-85-76:~$ apt-cache search kubelet
apt-cache search kubeadm
apt-cache search kubectl
kubelet - Node agent for Kubernetes clusters
kubeadm - Command-line utility for administering a Kubernetes cluster
golang-k8s-system-validators-dev - System-oriented validators for kubeadm preflight checks (library)
golang-k8s-utils-dev - Non-Kubernetes-specific utility libraries (library)
kubectl - Command-line utility for interacting with a Kubernetes cluster
golang-k8s-utils-dev - Non-Kubernetes-specific utility libraries (library)
kubecolor - colorizes kubectl output
kubectx - Fast way to switch between clusters and namespaces in kubectl
kubetail - Aggregate logs from multiple Kubernetes pods into one stream
ubuntu@ip-172-31-85-76:~$
```

After installing Kubernetes, we need to configure internet options to allow bridging.

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

```
ubuntu@ip-172-31-85-76:~$ 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
ubuntu@ip-172-31-85-76:~$
```

5. Perform this ONLY on the Master machine
Initialize the Kubecluster
sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=all
(before this perform necessary changes ,and check whether kubeadm or containerd are running)

```
ubuntu@ip-172-31-85-76:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errorssudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ig
-preflight-errors=all
I0914 10:54:54.900697    7853 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
        [WARNING NumCPU]: the number of available CPUs 1 is less than the required 2
        [WARNING Mem]: the system RAM (957 MB) is less than the minimum 1700 MB
```

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

```
        --discovery-token-ca-cert-hash sha256:88de331933950cda08c392a0a3b0136da1eccc5841819e8265028060a48ab3cb11
ubuntu@ip-172-31-85-76:~$  mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/
k ube-flannel.yml

```
ubuntu@ip-172-31-85-76:~$ sudo nano /etc/kubernetes/manifests/kube-apiserver.yaml                    sudo crictl ps^C
ubuntu@ip-172-31-85-76:~$ 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
```

Check the created pod using this command

Now, keep a watch on all nodes using the following command watch kubectl get nodes

Error:

```
ubuntu@ip-172-31-85-76:~$ sudo kubeadm join 172.31.85.76:6443 --token bugy6v.pkyb3qnm4nuwo8gt --discovery-token-ca-cert-hash sha256:88de33f93550cda08c392a0a3b013
6da1eccc5841819e826502806a48ab3cbf1
[preflight] Running pre-flight checks
error execution phase preflight: couldn't validate the identity of the API Server: failed to request the cluster-info ConfigMap: client rate limiter Wait returne
d an error: rate: Wait(n=1) would exceed context deadline
To see the stack trace of this error execute with --v=5 or higher
ubuntu@ip-172-31-85-76:~$
```

→The error indicates that kubeadm failed during pre-flight checks because it couldn't validate the identity of the API server due to a rate limiter exceeding the context deadline while requesting the cluster-info ConfigMap.
→The reason for this error is likely due to API server throttling caused by the rate limiting of requests. This can happen if there are too many requests being made to the Kubernetes API server, or network issues delaying the response, leading to a timeout before the operation completes.

Conclusion:

In this Advanced DevOps Lab experiment, we began by setting up three EC2 Ubuntu instances on AWS, designating one as the Master node and the others as Worker nodes. We then installed Docker and Kubernetes on all instances, ensuring Docker was properly configured. The Kubernetes cluster was initialized on the Master node, and the Flannel networking plugin was applied to facilitate communication between nodes. Finally, we joined the Worker nodes to the cluster using the provided token and hash, resulting in a fully operational Kubernetes cluster ready for managing and scaling containerized applications.