Name: **Dev Gaonkar** Div/Roll no.: **12**

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:

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.

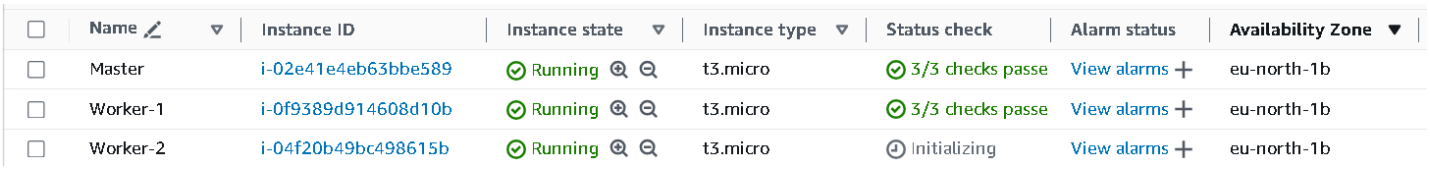
Kubernetes makes it easy to deploy and operate applications in a microservice architecture. It does so by creating an abstraction layer on top of a group of hosts so that development teams can deploy their applications and let Kubernetes manage the following activities:

- 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:

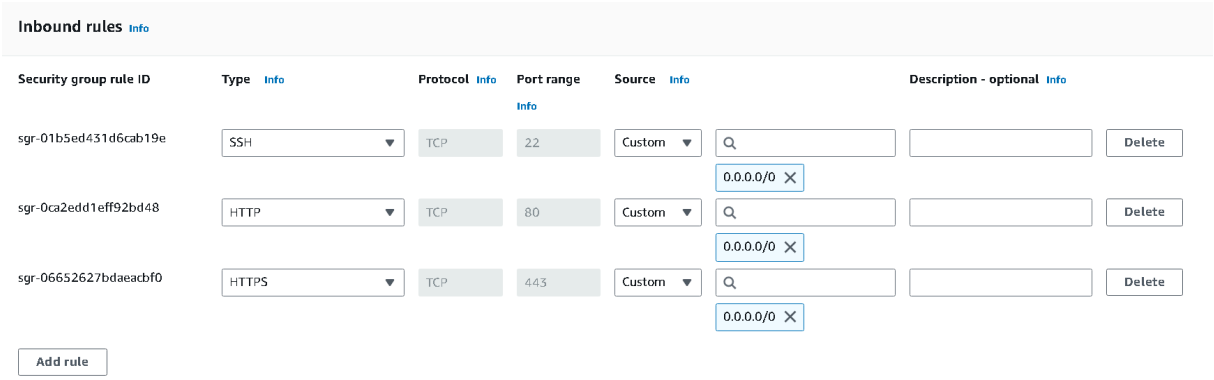
1. Create 3 EC2 Ubuntu Instances on AWS.

(Name 1 as Master, the other 2 as worker-1 and worker-2)



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
Master	i-02e41e4eb63bbe589	Running	t3.micro	3/3 checks passe	View alarms	eu-north-1b
Worker-1	i-0f9389d914608d10b	Running	t3.micro	3/3 checks passe	View alarms	eu-north-1b
Worker-2	i-04f20b49bc498615b	Running	t3.micro	Initializing	View alarms	eu-north-1b

2. Edit the Security Group Inbound Rules to allow SSH



Inbound rules Info

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info		Description - optional Info	
sgr-01b5ed431d6cab19e	SSH	TCP	22	Custom	0.0.0.0/0		Delete
sgr-0ca2edd1eff92bd48	HTTP	TCP	80	Custom	0.0.0.0/0		Delete
sgr-06652627bdaeacbf0	HTTPS	TCP	443	Custom	0.0.0.0/0		Delete

Add rule

3. SSH into all 3 machines

```
kagoran@LAPTOP-7NM7ITJ2:~$ chmod 400 devkeypair.pem
kagoran@LAPTOP-7NM7ITJ2:~$ ls -l devkeypair.pem
-r-------- 1 kagoran kagoran 1678 Sep 14 10:27 devkeypair.pem
```

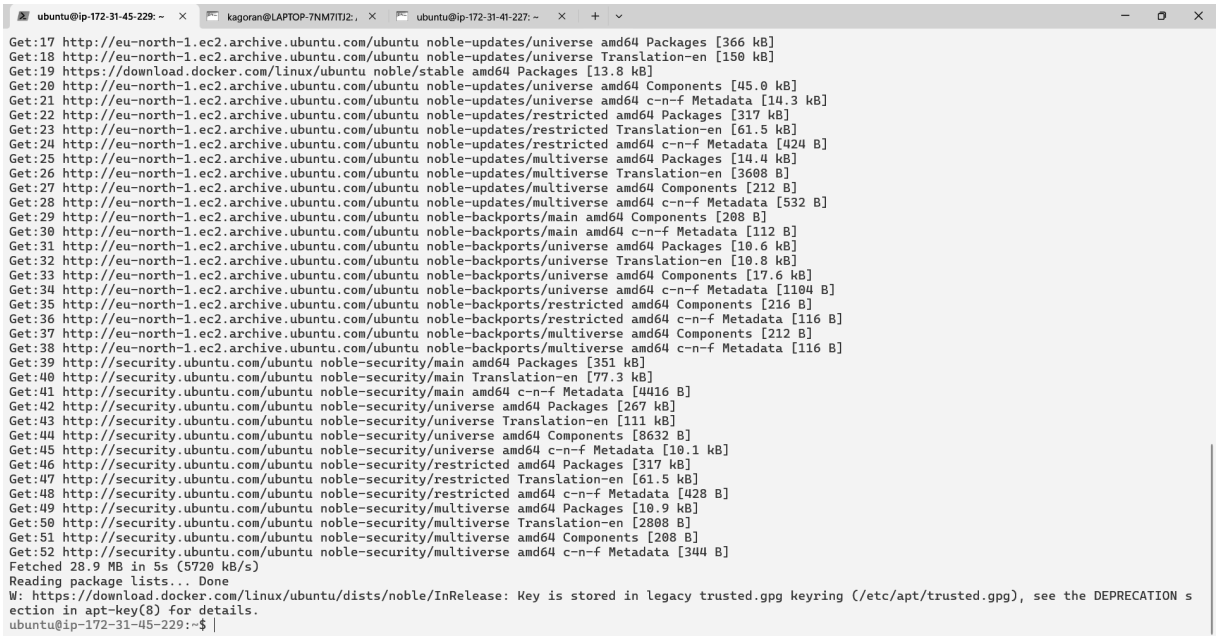
ssh -i <keyname>.pem ubuntu@<public_ip_address>

```
kagoran@LAPTOP-7NM7ITJ2:~$ ssh -i devkeypair.pem ubuntu@13.60.197.8
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sat Sep 14 04:58:19 UTC 2024

  System load:  0.0               Temperature:           -273.1 C
  Usage of /:   22.9% of 6.71GB   Processes:             108
  Memory usage: 21%               Users logged in:       0
  Swap usage:   0%                IPv4 address for ens5: 172.31.45.229


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


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Sat Sep 14 04:46:00 2024 from 13.48.4.203
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-45-229:~$
```

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
```



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
```

Install Kubernetes on all 3 machines

```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg |
sudo apt-key add -
cat << EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb https://apt.kubernetes.io/ kubernetes-xenial main EOF
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
```

```
ubuntu@ip-172-31-40-255:~$ # Add Kubernetes GPG key
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

# Add Kubernetes repository
sudo tee /etc/apt/sources.list.d/kubernetes.list <<EOF
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF

# Update package list
sudo apt-get update

# Install kubelet, kubeadm, and kubectl
sudo apt-get install -y kubelet kubeadm kubectl

# Hold the versions of Kubernetes components
sudo apt-mark hold kubelet kubeadm kubectl
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
deb https://apt.kubernetes.io/ kubernetes-xenial main
Hit:1 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://eu-north-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:6 https://packages.cloud.google.com/apt kubernetes-xenial InRelease
Err:7 https://packages.cloud.google.com/apt kubernetes-xenial Release
```

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-172-31-45-229:~$ # Disable swap
sudo swapoff -a

# Allow bridging for iptables
echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf

# Apply sysctl changes
sudo sysctl -p
net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-iptables = 1
ubuntu@ip-172-31-45-229:~$
```

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
```

```
Your Kubernetes control-plane has initialized successfully!

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 172.31.45.229:6443 --token s9zq75.bsi7js5f62ridulc \
        --discovery-token-ca-cert-hash sha256:91eae090fdd49337bf70d5bf7478e60bc85820d0996651871129a082db6fa8f1
ubuntu@ip-172-31-45-229:~$
```

Copy the join command and keep it in a notepad, we'll need it later.

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

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

Then, add a common networking plugin called flammel file as mentioned in the code.

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

```
ubuntu@ip-172-31-45-229:~$ 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
```

6. Perform this **ONLY on the worker machines**

```
sudo kubeadm join <ip> --token <token> \
        --discovery-token-ca-cert-hash <hash>
```

Now, notice the changes on the master terminal

```
Every 2.0s: kubectl get nodes                                                    ip-172-31-45-229: Sat Sep 14 12:19:42 20
24
NAME               STATUS   ROLES           AGE   VERSION
ip-172-31-45-229   Ready    control-plane   28m   v1.31.1
```

That's it, we now have a Kubernetes cluster running across 3 AWS EC2 Instances. This cluster can be used to further deploy applications and their loads being distributed across these machines.

Conclusion:

In this experiment, we set up a Kubernetes cluster across three AWS EC2 instances. Docker and Kubernetes components were successfully installed on each instance. The master node was initialized, and the Flannel network plugin was applied. While the master node is functioning correctly, worker nodes encountered issues joining the cluster, likely due to configuration or network problems. To complete the setup, further troubleshooting is needed on the worker nodes to resolve connectivity issues. Once resolved, the cluster will be fully operational, allowing for scalable management of containerized applications.