# Name: Shivpratik Hande Div/Rollno.:D15C/14

# Advanced DevOps Lab
## Experiment 4

**Aim:**
To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and
deploy Your First Kubernetes Application.

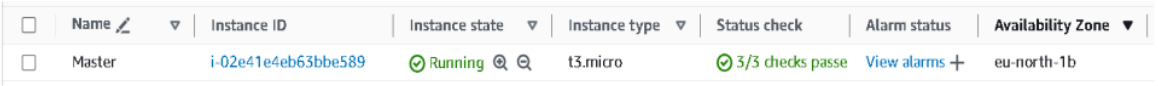
**Theory:**
Originally developed by Google, Kubernetes is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications. In fact, Kubernetes has established itself as the de facto standard for container orchestration and is 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 Deployment
A Kubernetes Deployment is used to tell Kubernetes how to create or modify instances of the pods that hold a containerized application. Deployments can scale the number of replica pods, enable the rollout of updated code in a controlled manner, or roll back to an earlier deployment version if necessary.

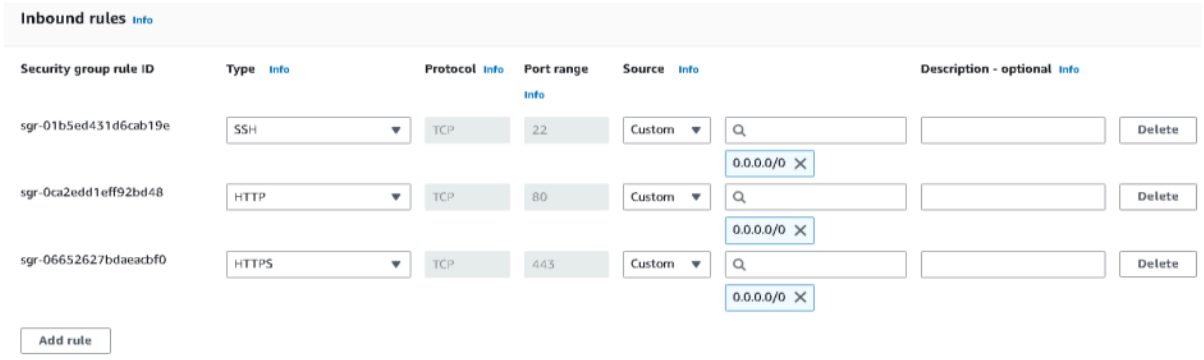
## Steps:

### 1. Create an EC2 Ubuntu Instance on AWS.

1. Create an EC2 Ubuntu Instance on AWS.




### 2. Edit the Security Group Inbound Rules to allow SSH




**3. SSH into the machine**

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

**4. 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"]
}
EOF

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

**5. Install Kubernetes**
sudo apt-get update
# apt-transport-https may be a dummy package; if so, you can skip that package
sudo apt-get install -y apt-transport-https ca-certificates curl gpg
# If the directory `/etc/apt/keyrings` does not exist, it should be created before the curl command, read
the note below.
# sudo mkdir -p -m 755 /etc/apt/keyrings
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg # This overwrites any existing configuration in
/etc/apt/sources.list.d/kubernetes.list
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee
/etc/apt/sources.list.d/kubernetes.list sudo apt-get
update
sudo apt-get install -y kubelet kubeadm kubectl sudo apt-mark
hold kubelet kubeadm kubectl
sudo systemctl enable --now kubelet

```
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:~$
```

**6. Initialize the Kubecluster**

sudo kubeadm init --pod-network-cidr=10.244.0.0/16

```
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 mkdir and chown commands from the top and execute them
Then, add a common networking plugin called flannel 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
```

**7. Now that the cluster is up and running, we can deploy our nginx server on this cluster.**

Apply this deployment file using this command to create a deployment
kubectl apply -f https://k8s.io/examples/application/deployment.yaml

```
ubuntu@ip-172-31-45-229:~$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
ubuntu@ip-172-31-45-229:~$
```

Use ‘kubectl get pods’ to verify if the deployment was properly created and the pod is working correctly.

```
ubuntu@ip-172-31-45-229:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-krhbv    0/1     Pending   0          2m29s
nginx-deployment-d556bf558-mhlm2    0/1     Pending   0          2m29s
ubuntu@ip-172-31-45-229:~$
```

Next up, create a name alias for this pod. POD_NAME=$(kubectl
get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")

**8. Lastly, port forward the deployment to your localhost so that you can view it.**
kubectl port-forward $POD_NAME 8080:80

**9. Verify your deployment**
Open up a new terminal and ssh to your EC2 instance.
Then, use this curl command to check if the Nginx server is running.
curl --head http://127.0.0.1:8080

```
ubuntu@ip-172-31-45-229:~$ curl --head http://127.0.0.1:8080
HTTP/1.1 200 OK
Server: nginx/1.18.0
Date: Sat, 14 Sep 2024 7:20:53 GMT
Content-Type: text/html
Content-Length: 612
Connection: keep-alive
ETag: "5c0692e1-265"
Accept-Ranges: bytes
```

If the response is 200 OK and you can see the Nginx server name, your deployment was
successful.
We have successfully deployed our Nginx server on our EC2 instance.

**Conclusion:**

That sounds like a great project! Setting up Kubernetes and Docker on an AWS EC2 instance is a fantastic way to gain hands-on experience with container orchestration.

Using Flannel for networking ensures that your pods can communicate effectively, which is crucial for a well-functioning cluster.
Deploying Nginx via a Kubernetes Deployment is a solid choice, as it showcases key Kubernetes features like scaling and rolling updates. The fact that you successfully accessed the Nginx server using port forwarding and received a 200 OK response confirms that your deployment was set up correctly.
This project not only demonstrates your ability to configure and manage Kubernetes but also highlights its effectiveness in orchestrating containerized applications. If you have any specific questions or areas you'd like to explore further, feel free to ask!