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:

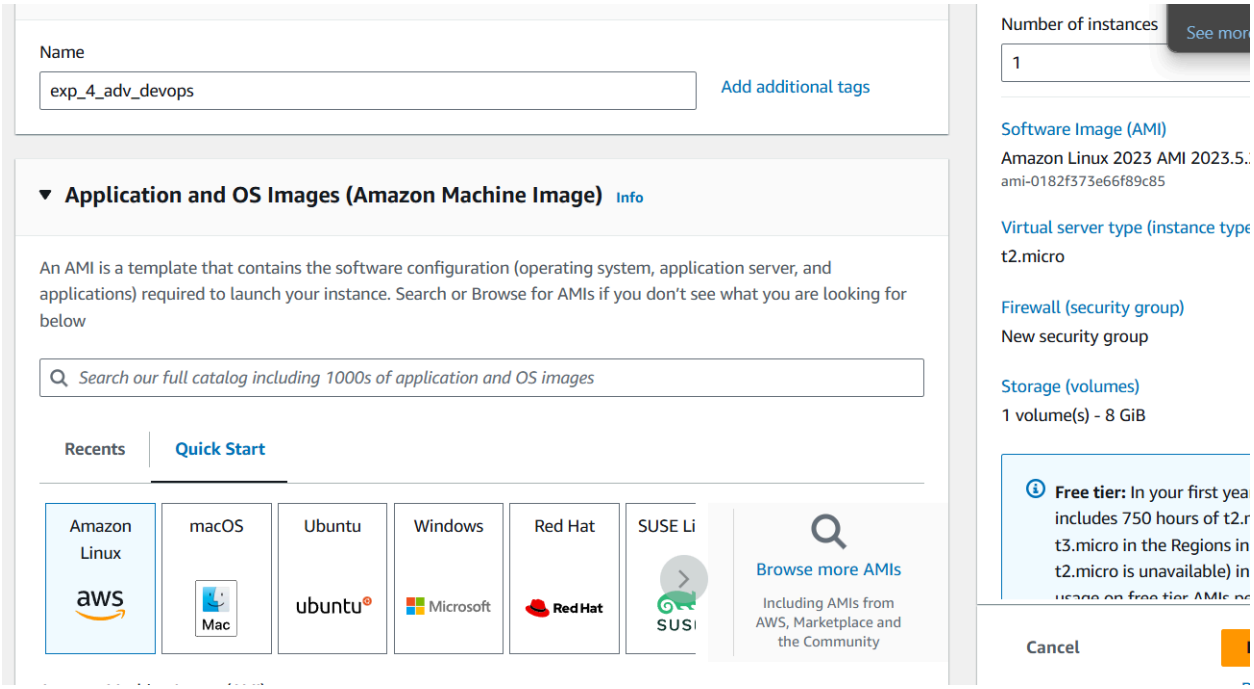
Originally developed by Google, Kubernetes is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications. In fact, Kubernetes has established itself as the defacto standard for container orchestration and is the flagship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

Kubernetes 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. A] Creation Of EC-2 instance
→ Launch an AWS EC2 instance named exp_4_adv_devops using an AWS Linux AMI. Configure the Security Group's Inbound Rules to allow SSH access, then choose the t2.micro instance type.



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
exp_4_adv_de...	i-05b0c08fcc2ebe34f	Running	t2.micro	Initializing	View alarms	us-east-1a	ec2-54-2
Ayush_exp4	i-026702c557caf29ff	Shutting-d...	t2.micro	–	View alarms	us-east-1f	ec2-3-23

B] Connecting to an AWS EC2 Instance via SSH

→ To connect to an AWS EC2 instance via SSH, change the key file's permission using chmod 400 "keyname.pem", then run ssh -i .pem ubuntu@ to establish the connection. ssh -i "master.pem" ec2-user@ec2-44-223-225-252.compute-1.amazonaws.com

```
Ayush Maurya@LAPTOP-2FM8RHB4 MINGW64 ~ (master)
$ cd download
bash: cd: download: No such file or directory

Ayush Maurya@LAPTOP-2FM8RHB4 MINGW64 ~ (master)
$ cd downloads

Ayush Maurya@LAPTOP-2FM8RHB4 MINGW64 ~/downloads (master)
$ chmod 400 "master2.pem"

Ayush Maurya@LAPTOP-2FM8RHB4 MINGW64 ~/downloads (master)
$ ssh -i "master2.pem" ec2-user@ec2-44-222-225-252.compute-1.amazonaws.com
The authenticity of host 'ec2-44-222-225-252.compute-1.amazonaws.com (44.222.225
.252)' can't be established.
ED25519 key fingerprint is SHA256:hmdf3OgLshMCD9uOK1ppCmNzcmv2eBBe3OvtaLUZwzI.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-44-222-225-252.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
       #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-26-1 ~]$
```

2. Installation of Docker →

Run the following command: sudo yum install docker -y

```
[ec2-user@ip-172-31-29-131 ~]$ sudo yum install docker -y
Last metadata expiration check: 0:12:03 ago on Sat Sep 14 13:30:51 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture        Version                 Repository
                                                                              Size
================================================================================
Installing:
 docker                  x86_64              25.0.6-1.amzn2023.0.2   amazonlinux
                                                                             44 M
Installing dependencies:
 containerd              x86_64              1.7.20-1.amzn2023.0.1   amazonlinux
                                                                             33 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
 libnfnetlink            x86_64              1.0.1-19.amzn2023.0.2   amazonlinux
                                                                             30 k
 libnftnl                x86_64              1.2.2-2.amzn2023.0.2    amazonlinux
```

Then, configure cgroup in a daemon.json file by using following commands :

→ 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

```
[ec2-user@ip-172-31-29-131 docker]$ 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
{
"exec-opts":
["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
[ec2-user@ip-172-31-29-131 docker]$
```

After this run the following command to enable and start docker and also to load the daemon.json file.
→ sudo systemctl enable docker → sudo systemctl daemon-reload
→ sudo systemctl restart docker → docker -v

```
[ec2-user@ip-172-31-29-131 docker]$ sudo systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
[ec2-user@ip-172-31-29-131 docker]$ sudo systemctl daemon-reload
[ec2-user@ip-172-31-29-131 docker]$ sudo systemctl restart docker
[ec2-user@ip-172-31-29-131 docker]$ docker -v
Docker version 25.0.5, build 5dc9bcc
[ec2-user@ip-172-31-29-131 docker]$
```

3. Install Kubernetes →
a]SELinux needs to be disabled before configuring kubelet Run the following command
→sudo setenforce 0
→ sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config

b] We are adding kubernetes using the repository whose command is given below.
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
enabled=1
gpgcheck=1

gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.x
ml.key exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF

```
[ec2-user@ip-172-31-29-131 docker]$ sudo setenforce 0
[ec2-user@ip-172-31-29-131 docker]$ cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo/' /etc/selinux/config
[kubernetes]-172-31-29-131 docker]$ cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
enabled=1ttps://pkgs.k8s.io/core:/stable:/v1.30/rpm/
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.x
ml.key exclude=kubelet kubeadm kubectl cri-tools kubernetes-cniomd.x
EOFkey exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.x
ml.key exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
[ec2-user@ip-172-31-29-131 docker]$
```

c] After that Run following command to make the updation and also to install kubelet, kubeadm, kubectl: → sudo yum update
→sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes

```
[ec2-user@ip-172-31-29-131 docker]$ sudo yum update
Kubernetes                                113 kB/s |  17 kB     00:00
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-29-131 docker]$ sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Last metadata expiration check: 0:01:14 ago on Sat Sep 14 13:55:34 2024.
Dependencies resolved.
================================================================================
 Package                 Arch    Version                  Repository    Size
================================================================================
Installing:
 kubeadm                 x86_64  1.30.5-150500.1.1        kubernetes    10 M
 kubectl                 x86_64  1.30.5-150500.1.1        kubernetes    10 M
 kubelet                 x86_64  1.30.5-150500.1.1        kubernetes    17 M
Installing dependencies:
 conntrack-tools         x86_64  1.4.6-2.amzn2023.0.2     amazonlinux  208 k
 cri-tools               x86_64  1.30.1-150500.1.1        kubernetes   8.6 M
 kubernetes-cni          x86_64  1.4.0-150500.1.1         kubernetes   6.7 M
 libnetfilter_cthelper   x86_64  1.0.0-21.amzn2023.0.2    amazonlinux   24 k
 libnetfilter_cttimeout  x86_64  1.0.0-19.amzn2023.0.2    amazonlinux   24 k
 libnetfilter_queue      x86_64  1.0.5-2.amzn2023.0.2     amazonlinux   30 k

Transaction Summary
================================================================================
Install  9 Packages

Total download size: 53 M
```

d] After installing Kubernetes, we need to configure internet options to allow bridging.
1. sudo swapoff -a
2. echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
3. sudo sysctl -p

```
[ec2-user@ip-172-31-29-131 docker]$ 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
[ec2-user@ip-172-31-29-131 docker]$
```

4. Initialize the Kubecluster

→ a] Initializes a Kubernetes cluster with kubeadm, sets up the pod network CIDR to 10.244.0.0/16 for network communication, and ignores preflight checks for CPU and memory requirements.

sudo kubeadm init --ignore-preflight-errors=NumCPU,Mem

```
[root@ip-172-31-26-1 docker]# sudo systemctl enable --now kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /usr/lib/systemd/system/kubelet.service.
[root@ip-172-31-26-1 docker]# 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
error execution phase preflight: [preflight] Some fatal errors occurred:
        [ERROR NumCPU]: the number of available CPUs 1 is less than the required 2
        [ERROR Mem]: the system RAM (949 MB) is less than the minimum 1700 MB
[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
[root@ip-172-31-26-1 docker]# kubeadm init --ignore-preflight-errors=NumCPU,Mem
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
        [WARNING NumCPU]: the number of available CPUs 1 is less than the required 2
        [WARNING Mem]: the system RAM (949 MB) is less than the minimum 1700 MB
```

b] copy the token and save for future use .

kubeadm join 172.31.26.1:6443 --token s9t4e8.ygz2u9mm5okam1vq \

--discovery-token-ca-cert-hash

sha256:45eca28e5c3fee977bf721e09d48012e10734c339937e7742b68c3bfd538c26c

c] 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

```
kubeadm join 172.31.26.1:6443 --token s9t4e8.ygz2u9mm5okam1vq \
        --discovery-token-ca-cert-hash sha256:45eca28e5c3fee977bf721e09d48012e10734c339937e7742b68c3bfd538c26c
```

d] Then, add a common networking plugin called flannel as mentioned in the code.

kubectl apply -f

https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

```
[root@ip-172-31-26-1 docker]# 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
```

5. Deploying an NGINX Server on Your Kubernetes Cluster → a]Now that the cluster is up and running,we can deploy our nginx server on this cluster. Apply deployment using this following command:

kubectl apply -f https://k8s.io/examples/pods/simple-pod.yaml

```
bash: https://k8s.io/examples/pods/simple-pod.yaml: No such file or directory
[root@ip-172-31-26-1 docker]# kubectl apply -f https://k8s.io/examples/pods/simple-pod.yaml
pod/nginx created
[root@ip-172-31-26-1 docker]#
```

b]Then use ***kubectl get pods*** to check whether the pod gets created or not.

```
[root@ip-172-31-26-1 docker]# kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   0/1     Pending   0          54s
```

c]To convert state from pending to running use following command:
kubectl describe pod nginx This command will help to describe the pods it gives reason for failure as it shows the untolerated taints which need to be untainted

```
[ec2-user@ip-172-31-26-1 ~]$ sudo nano nginx-deployment.yaml
[ec2-user@ip-172-31-26-1 ~]$ kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx created
[ec2-user@ip-172-31-26-1 ~]$ kubectl get pods
No resources found in default namespace.
[ec2-user@ip-172-31-26-1 ~]$ kubectl get deployments
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nginx   0/1     1            0           78s
[ec2-user@ip-172-31-26-1 ~]$ kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
nginx-7769f8f85b-wlmzm   0/1     Pending   0          80s
[ec2-user@ip-172-31-26-1 ~]$ kubectl describe deployment nginx
Name:                   nginx
Namespace:              default
CreationTimestamp:      Sat, 14 Sep 2024 19:16:31 +0000
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=nginx
```

FOR ERROR (paste this in nginx-deployment.ymal & follow above commands)

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      tolerations:
      - key: "node-role.kubernetes.io/control-plane"
        operator: "Exists"
        effect: "NoSchedule"
      containers:
      - name: nginx
        image: nginx
```

ports:
- containerPort: 80

```
  ----           ------  ------
  Available      False   MinimumReplicasUnavailable
  Progressing    True    ReplicaSetUpdated
OldReplicaSets:  <none>
NewReplicaSet:   nginx-7769f8f85b (1/1 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  98s   deployment-controller  Scaled up replica set nginx-7769f8f85b to 1
[ec2-user@ip-172-31-26-1 ~]$ kubectl get events
LAST SEEN   TYPE      REASON                    OBJECT                                MESSAGE
10m         Normal    NodeHasSufficientMemory   node/ip-172-31-26-1.ec2.internal   Node ip-172-31-26-1.ec2.internal status is now: NodeHasSufficientMemory
10m         Normal    NodeHasNoDiskPressure     node/ip-172-31-26-1.ec2.internal   Node ip-172-31-26-1.ec2.internal status is now: NodeHasNoDiskPressure
10m         Normal    NodeHasSufficientPID      node/ip-172-31-26-1.ec2.internal   Node ip-172-31-26-1.ec2.internal status is now: NodeHasSufficientPID
10m         Normal    NodeAllocatableEnforced   node/ip-172-31-26-1.ec2.internal   Updated Node Allocatable limit across pods
```

6. Now check pod status is is running

```
NAME                           STATUS   ROLES           AGE   VERSION
ip-172-31-26-1.ec2.internal    Ready    control-plane   83s   v1.31.1
[ec2-user@ip-172-31-26-1 ~]$
```

7.Lastly, mention the port you want to host. Here i have used localhost 8081 then check it.
kubectl port-forward nginx 8081:80

ERROR:
Error: Unable to find a match: kubeadm kubelet kubectl
sudo: kubeadm: command not found

REASONS
→Incomplete Cluster Setup: If the Kubernetes cluster hasn't been properly initialized using kubeadm, commands like kubectl port-forward may not work.
→Kubeconfig Not Set: kubectl uses a configuration file, typically located at /etc/kubernetes/admin.conf, to interact with the Kubernetes cluster.
→Pod Not Running: kubectl port-forward forwards traffic from a local port to a port on a Kubernetes pod. If the NGINX pod isn't running (e.g., it's in a Pending or CrashLoopBackOff state), the port-forward operation will fail because there's no running pod to connect to.

8. 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
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: We successfully set up a Kubernetes cluster on AWS EC2, addressing issues related to component setup and residual configurations. We ensured proper cleanup of previous Kubernetes files and mounts, verified the kubelet service, and applied Flannel for networking. Finally, we resolved connectivity issues, and after a thorough review of logs and configuration, we deployed and exposed an NGINX server using Kubernetes services, preparing the cluster for efficient traffic management and scaling.