

**Advanced DevOps Lab
Experiment:4**

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

Overview of Kubernetes and Kubectl**What is Kubernetes?**

Kubernetes, often referred to as K8s, is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. Originally developed by Google, it has become the industry standard for managing container workloads due to its flexibility and robust features.

Core Concepts of Kubernetes

Containers: These are lightweight, portable packages that include everything needed to run an application, ensuring consistency across different environments.

Pods: The smallest deployable units in Kubernetes, pods can contain one or more containers that share storage and network resources.

Nodes: A node is a worker machine in the Kubernetes cluster that runs at least one pod. Nodes can be either physical or virtual machines.

Clusters: A cluster comprises multiple nodes that run containerized applications. The control plane manages the cluster's state.

Services: Services provide stable endpoints for accessing pods and facilitate load balancing and service discovery.

Deployments: A deployment manages the lifecycle of pods, allowing users to specify the number of replicas and facilitating rolling updates and rollbacks.

Architecture of Kubernetes

Kubernetes follows a client-server architecture consisting of:

Control Plane: Manages the cluster and includes components like the API server (the front-end for the control plane), scheduler (assigns pods to nodes), controller manager (regulates cluster state), and etcd (a distributed key-value store for cluster data).

Worker Nodes: Each node runs components like kubelet (ensures containers are running), kube-proxy (manages network communication), and a container runtime (e.g., Docker).

Role of Kubectl in Kubernetes

What is Kubectl?

Kubectl is the command-line interface used to interact with the Kubernetes API server. It enables users to manage resources within a Kubernetes cluster effectively.

Configuration Files

Configuration files are essential for defining how resources should be created or modified within Kubernetes. Users can employ declarative configurations (using YAML/JSON files) or imperative commands directly in the terminal.

Deploying Applications on Kubernetes

Application Deployment Lifecycle

1. **Define Application Requirements:** Identify necessary resources such as CPU, memory, storage, etc.
2. **Create Deployment Configurations:** Write deployment manifests specifying container images, replicas for scaling, health checks, etc.
3. **Deploying with Kubectl:** Use kubectl commands like `kubectl apply` to deploy applications based on these configurations.
4. **Monitoring and Scaling Applications:** Monitor performance metrics and adjust deployments based on traffic demands.
5. **Updating Applications:** Modify deployment configurations for updates; Kubernetes supports rolling updates by default.
6. **Rollback Capabilities:** If an update causes issues, kubectl allows easy rollback to previous versions using commands like `kubectl rollout undo`.

Implementation

Step 1.Creation of 2 EC2 Ubuntu Instances on AWS.

Services

Search

[Alt+S]

Instances (2) Info

Last updated less than a minute ago

Connect

Instance state

Actions

Find Instance by attribute or tag (case-sensitive)

All states

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
<input type="checkbox"/>	worker-node1	i-0526f6be95551ba74	Running	t2.micro	2/2 checks passed	View alarms	us-east-1a	ec2-3-81-7
<input type="checkbox"/>	master1	i-0aef82b0ccd222219	Running	t2.micro	2/2 checks passed	View alarms	us-east-1a	ec2-34-207

Step 2.Edit inbound rules of security group ‘launch-wizard-1’ and set ‘All Traffic’

arch

[Alt+S]

Inbound security group rules successfully modified on security group (sg-020c81f3d7e528326 | launch-wizard-1)

Details

EC2 > Security Groups > sg-020c81f3d7e528326 - launch-wizard-1

sg-020c81f3d7e528326 - launch-wizard-1

Actions

Details

Security group name

launch-wizard-1

Security group ID

sg-020c81f3d7e528326

Description

launch-wizard-1 created 2024-09-18T04:49:14.326Z

VPC ID

vpc-085eda6d3476ae0d2

Owner

605134455955

Inbound rules count

1 Permission entry

Outbound rules count

1 Permission entry

Inbound rules (1)

Manage tags

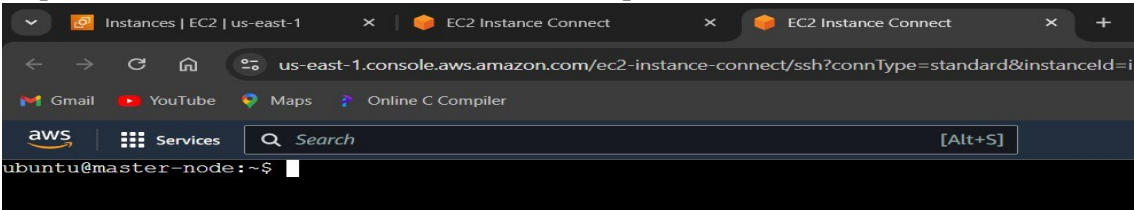
Edit inbound rules

Search

< 1 >

<input type="checkbox"/>	Name	Security group rule...	IP version	Type	Protocol	Port range
<input type="checkbox"/>	-	sgr-0eaddedf6a07229a4	IPv4	All traffic	All	All

Step 3. Set master and worker as hostname on respective servers



Step 4.Installation of docker

4.1 - sudo apt-get update

```
aws | Services | Search | [Icons] | N. Virginia | ShravaniAnilPatil
ubuntu@master-node:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation [15.0 MB]
```

4.2 - sudo apt-get install docker.io

```
ubuntu@master-node:~$ sudo apt-get install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap
  docker-buildx docker-compose-v2 docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc
  ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 133 not upgraded.
Need to get 76.8 MB of archives.
```

4.3 – sudo systemctl enable docker

4.4 – sudo systemctl status docker

```
aws | Services | Search | [Icons] | N. Virginia | ShravaniAnilPatil
ubuntu@master-node:~$ sudo systemctl enable docker
ubuntu@master-node:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset=
   Active: active (running) since Wed 2024-09-18 05:07:22 UTC; 2min 35s ago
   TriggeredBy: ● docker.socket
     Docs: https://docs.docker.com
    Main PID: 2659 (dockerd)
       Tasks: 8
      Memory: 31.3M (peak: 33.2M)
         CPU: 293ms
    CGroup: /system.slice/docker.service
            └─2659 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/c
```

Step 5.Installation of Kubernetes-

5.1 sudo apt-get update

5.2 install ca certificate

```
aws Services [Search] [Alt+S] N. Virginia ShrivaniAnilPatil
ubuntu@master-node:~$ 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 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
ubuntu@master-node:~$ sudo apt-get install -y apt-transport-https ca-certificates curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203).
ca-certificates set to manually installed.
The following additional packages will be installed:
```

5.3 Download the public signing key for the Kubernetes package repositories.

```
aws Services [Search] [Alt+S] N. Virginia ShrivaniAnilPatil
ubuntu@master-node:~$ sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg https://packages.cloud.google.com/apt/doc/apt-key
ubuntu@master-node:~$
```

5.4 Add the appropriate Kubernetes apt repository

```
aws Services [Search] [Alt+S] N. Virginia ShrivaniAnilPatil
ubuntu@workeri:~$ 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.l
ist.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/
ubuntu@workeri:~$ 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 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:5 https://prod-cdn.packages.k8s.io/repositories/ipv1/kubernetes:/core:/stable:/v1.31/deb InRelease [1186 B]
Err:5 https://prod-cdn.packages.k8s.io/repositories/ipv1/kubernetes:/core:/stable:/v1.31/deb InRelease
The following signatures couldn't be verified because the public key is not available: NO_PUBKEY 234654DA9A296436
Reading package lists... Done
W: GPG error: https://prod-cdn.packages.k8s.io/repositories/ipv1/kubernetes:/core:/stable:/v1.31/deb InRelease: The following signatures couldn't be verified bec
ause the public key is not available: NO_PUBKEY 234654DA9A296436
E: The repository 'https://pkgs.k8s.io/core:/stable:/v1.31/deb InRelease' is not signed.
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.
```

5.5 sudo apt-get update

5.6 sudo apt-get install -y kubelet kubeadm kubectl

```
0% [Connecting to pkgs.k8s.io (34.107.204.206)] [Waiting for headers] [Connec
tp://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:2 ht
0% [Connected to pkgs.k8s.io (34.107.204.206)] [Connecting to security.ubuntu
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:5 ht
tp://security.ubuntu.com/ubuntu noble-security InRelease
Get:4 https://prod-cdn.packages.k8s.io/repositories/iscv/kubernetes:/core:/stable:/v1.31/deb InRelease [1186 B]
Get:6 https://prod-cdn.packages.k8s.io/repositories/iscv/kubernetes:/core:/stable:/v1.31/deb Packages [4865 B]
Fetched 6091 B in 1s (7749 B/s)
Reading package lists... Done
ubuntu@master-node:~$ sudo apt-get install -y kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  contrack cri-tools kubeadm kubectl kubelet kubernetescni
The following NEW packages will be installed:
  contrack cri-tools kubeadm kubectl kubelet kubernetescni
0 upgraded, 6 newly installed, 0 to remove and 130 not upgraded.
Need to get 87.4 MB of archives.
After this operation, 314 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 contrack amd64 1:1.4.8-1ubuntu1 [37.9 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 cri-tools amd64 1.31.1-1.1 [15.7 MB]
Get:3 https://prod-cdn.packages.k8s.io/repositories/iscv/kubernetes:/core:/stable:/v1.31/deb kubeadm 1.31.1-1.1 [11.4 MB]
Get:4 https://prod-cdn.packages.k8s.io/repositories/iscv/kubernetes:/core:/stable:/v1.31/deb kubectl 1.31.1-1.1 [11.2 MB]
Get:5 https://prod-cdn.packages.k8s.io/repositories/iscv/kubernetes:/core:/stable:/v1.31/deb kubernetescni 1.5.1-1.1 [33.9 MB]
Get:6 https://prod-cdn.packages.k8s.io/repositories/iscv/kubernetes:/core:/stable:/v1.31/deb kubelet 1.31.1-1.1 [15.2 MB]
```

5.7 sudo apt-mark hold kubelet kubeadm kubectl

```
ubuntu@master-node:~$ sudo apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
ubuntu@master-node:~$
```

Step.6 Kubernetes Deployment

6.1 sudo swapoff -a

6.2 Initialize Kubernetes on Master Node - sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=all

```
table kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

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 172.31.33.243:6443 --token ltp4ao.gzdxdpqly6bxvznz \
--discovery-token-ca-cert-hash sha256:4423cf44f5102d477fa92160e76e03f78f941d4e7a5836cd46bb14517dfab5ad
ubuntu@master-node:~$
```

i-Oaef82b0ccd222219 (master1)

PublicIPs: 34.207.105.187 PrivateIPs: 172.31.33.243

Step 7.to create a directory for the cluster:

7.1mkdir -p \$HOME/.kube

7.2sudo cp -i /etc/kubernetes/admin.conf \$HOME/.kube/config

7.3sudo chown HOME/.kube/config

Step 8. Deploy Pod Network to Cluster and Join Worker Node to Cluster

```
aws Services Search [Alt+S] N. Virginia ShrivaniAnilPatil
ubuntu@worker-nod:~$ sudo [kubeadm join 172.31.33.243:6443 --token ltp4ao.gzdzdpkqly6bxvz --discovery-token-ca-cert-hash sha256:4423cf44f5102d477fa92160e76e03f78f941d4e7a5836cd46bb14517dfab5ad --ignore-preflight-errors=all]
[preflight] Running pre-flight checks
[WARNING FileExisting-socat]: socat not found in system path
[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-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 513.56899ms
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

This node has joined the cluster:
* Certificate signing request was sent to apiservert 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.

ubuntu@worker-nod:~$
```

Verify that everything is running and communicating:

8.1kubectl get pods --all-namespaces

8.2kubectl get nodes

```
ubuntu@master-node:~$ kubectl get pods --all-namespaces
NAME                                READY    STATUS    RESTARTS    AGE
kube-flannel                        kube-flannel-ds-gx9xg          1/1    Running    0          14m
kube-flannel                        kube-flannel-ds-tl79d          1/1    Running    0          6m2s
kube-system                         coredns-7c65d6cfc9-nrns4       1/1    Running    0          23m
kube-system                         coredns-7c65d6cfc9-pnh9p       1/1    Running    0          23m
kube-system                         etcd-master-node                1/1    Running    0          23m
kube-system                         kube-apiserver-master-node      1/1    Running    0          23m
kube-system                         kube-controller-manager-master 1/1    Running    0          23m
kube-system                         kube-proxy-8rz72                0/1    CrashLoopBackOff 7 (3m42s ago) 23m
kube-system                         kube-proxy-w55hg                0/1    CrashLoopBackOff 4 (29s ago)   6m2s
kube-system                         kube-scheduler-master-node      1/1    Running    0          23m
ubuntu@master-node:~$ kubectl get nodes
NAME                STATUS    ROLES    AGE    VERSION
master-node         Ready    control-plane 23m    v1.31.1
worker-nod          Ready    <none>     6m22s  v1.31.1
ubuntu@master-node:~$
```

Step 9. Create one file deploy.yaml

```
ubuntu@master-node:~$ sudo nano deploy.yaml
ubuntu@master-node:~$
```

```
ubuntu@master-node:~$ cat deploy.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
ubuntu@master-node:~$
```

Step 10 : Create Deployment

```
aws | Services | Search [Alt+S]
ubuntu@master-node:~$ kubectl create -f deploy.yaml
deployment.apps/nginx-deployment created
ubuntu@master-node:~$
```

(EXTRA) – kubectl get namespaces

```
ubuntu@master-node:~$ kubectl get namespaces
NAME                STATUS    AGE
default              Active    9h
kube-flannel         Active    8h
kube-node-lease      Active    9h
kube-public          Active    9h
kube-system          Active    9h
ubuntu@master-node:~$
```


Step 11. After deployment verify the same:

```
ubuntu@master-node:~$ kubectl get deploy
NAME                READY    UP-TO-DATE    AVAILABLE    AGE
nginx-deployment    3/3      3              3            3m13s
```

Step 12: Expose the Application: Create a service to expose the deployment.

```
ubuntu@master-node:~$ kubectl expose deployment.apps/nginx-deployment \
> --type="LoadBalancer"
service/nginx-deployment exposed
ubuntu@master-node:~$
```

Step 13: Verfiy the service

```
service/nginx-deployment exposed
ubuntu@master-node:~$ kubectl get svc
NAME                TYPE                CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
kubernetes          ClusterIP           10.96.0.1     <none>         443/TCP          4h43m
nginx-deployment    LoadBalancer      10.101.59.94  <pending>      80:31041/TCP     4m34s
ubuntu@master-node:~$
```

Step 14: Access the application

