Terna Engineering College

Computer Engineering Department
Program: Sem VI

Course: Cloud Computing Lab(CSL605)

PART A
(PART A: TO BE REFFERED BY STUDENTS)

Experiment No.10

A.1 Aim:

To study and implement container orchestration using Kubernetes .

A.2 Prerequisite:

Knowledge of container technology, Docker, basics of node.js

A.3 Objective:

To understand the steps to deploy Kubernetes Cluster on local systems, deploy applications on Kubernetes, creating a Service in Kubernetes, develop Kubernetes configuration files in YAML and creating a deployment in Kubernetes using YAML

A.4 Outcome: (LO 4)

After successful completion of this experiment student will be able to,
Understand the concept of Kubernetes cluster.

A.5 Theory:

Container orchestration tools provide a framework for managing containers and microservices architecture at scale. There are many container orchestration tools that can be used for container lifecycle management. Some popular options are Kubernetes, Docker Swarm, and Apache Mesos.

Kubernetes orchestration allows you to build application services that span multiple containers, schedule containers across a cluster, scale those containers, and manage their health over time.

Kubernetes eliminates many of the manual processes involved in deploying and scaling containerized applications.

Main components of Kubernetes:

- **Cluster:** A control plane and one or more compute machines, or nodes.
- **Control plane:** The collection of processes that control Kubernetes nodes. This is where all task assignments originate.
- **Kubelet:** This service runs on nodes and reads the container manifests and ensures the defined containers are started and running.
- **Pod:** A group of one or more containers deployed to a single node. All containers in a pod share an IP address, IPC, hostname, and other resources.

The following instructions show you how to set up a simple, single node Kubernetes cluster using Docker.

1. Create a Kubernetes cluster.
2. Deploy an app.
3. Explore your app.
4. Expose your app publicly.
5. Scale up your app.
6. Update your app.

Steps:

1. Open the terminal on Ubuntu.

2. Install the necessary dependencies by using the following command:

```
$ sudo apt-get update

$ sudo apt-get install -y apt-transport-https
```

3. Install Docker Dependency by using the following command:

```
$ sudo apt install docker.io
```

Start and enable Docker with the following commands:

```
$ sudosystemctl start docker

$ sudosystemctl enable docker
```

4. Install the necessary components for Kubernetes.

First, install the curl command:

```
$ sudo apt-get install curl
```

Then download and add the key for the Kubernetes install:

```
$ sudo curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add
```

Change permission by using the following command:

```
$ sudochmod 777 /etc/apt/sources.list.d/
```

Then, add a repository by creating the file /etc/apt/sources.list.d/kubernetes.list and enter the following content:

deb http://apt.kubernetes.io/ kubernetes-xenial main

Save and close that file.

Install Kubernetes with the following commands:

```
$ apt-get update

$ apt-get install -y kubeletkubeadmkubectlkubernetes-cni
```

5. Before initializing the master node, we need to swap off by using the following command:

```
$ sudoswapoff -a
```

6. Initialize the master node using the following command:

```
$ sudokubeadminit
```

You get three commands: copy and paste them and press and “enter.”

```
$ mkdir -p $HOME/.kube

$ sudocp -i /etc/kubernetes/admin.conf $HOME/.kube/config

$ sudochown $(id -u):$(id -g) $HOME/.kube/config
```

7. Deploy pods using the following command:

```
$ $ sudokubectl apply -f https://raw.githubusercontent.com/coreos/flannel/
master/Documentation/kube-flannel.yml

$ sudokubectl apply -f https://raw.githubusercontent.com/coreos/flannel/
master/Documentation/k8s-manifests/kube-flannel-rbac.yml
```

8. To see all pods deployed, use the following command:

```
$ sudokubectl get pods –all-namespaces
```

9. To deploy an NGINX service (and expose the service on port 80), run the following commands:

```
$ sudokubectl run --image=nginxnginx-app --port=80 --env="DOMAIN=cluster"

$ sudokubectl expose deployment nginx-app --port=80 --name=nginx-http
```

10. To see the services listed, use the following command:

```
$ sudo docker ps -a
```

PART B

(PART B: TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the ERP or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no ERP access available)

Roll No.: B30	Name: Pranjal Bhatt
Class : TE B COMPs	Batch : B2
Date of Experiment:	Date of Submission:
Grade :	

B.1Question of Curiosity:

(To be answered by student based on the practical performed and learning/observations)

Q.1 What is Kubernetes?

Kubernetes is an open-source container orchestration platform developed by Google. It automates the **deployment**, **scaling**, **management**, and **operation** of containerized applications. It groups containers into **pods**, which are deployed and managed across a cluster of nodes.

Q.2 What are the features of Kubernetes?

1. Automated Deployment & Rollbacks
2. Self-healing (auto-restarts failed containers)
3. Horizontal Scaling (scale apps up/down automatically)
4. Service Discovery & Load Balancing
5. Storage Orchestration (manages volumes dynamically)
6. Secret & Configuration Management
7. Multi-cloud & Hybrid support

8. Infrastructure as Code with YAML

Q.3 What are the different services within Kubernetes?

- **ClusterIP** – Exposes service on an internal IP; accessible only within the cluster.
- **NodePort** – Exposes service on each Node's IP at a static port.
- **LoadBalancer** – Exposes the service externally using a cloud provider's load balancer.
- **ExternalName** – Maps the service to an external DNS name.
- **Headless Service** – Used when you want direct access to pod IPs.

Q.4.What is ClusterIP ,NodePort, LoadBalancer in kubernetes?

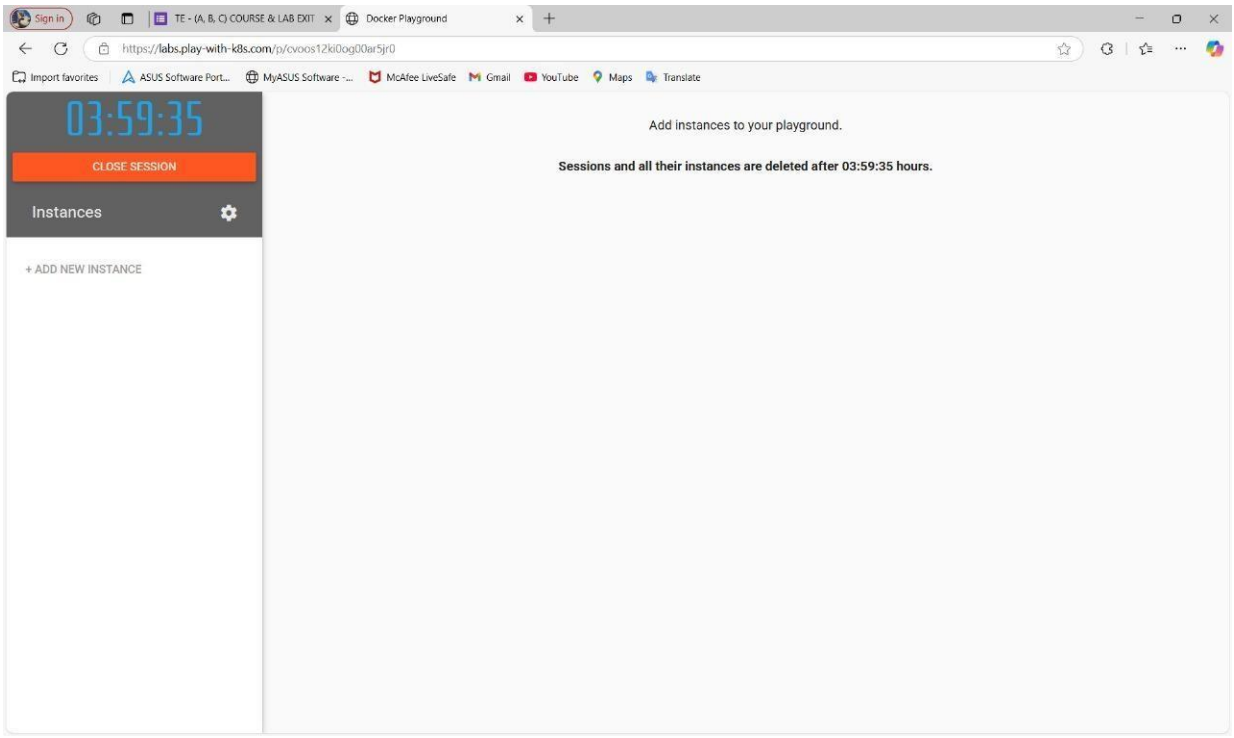
Type	Description
ClusterIP	Default type; service is only accessible **within the cluster**.
NodePort	Opens a specific port on **each node** to access the service externally.
LoadBalancer	Provisions a **cloud load balancer** and exposes the service to the internet.

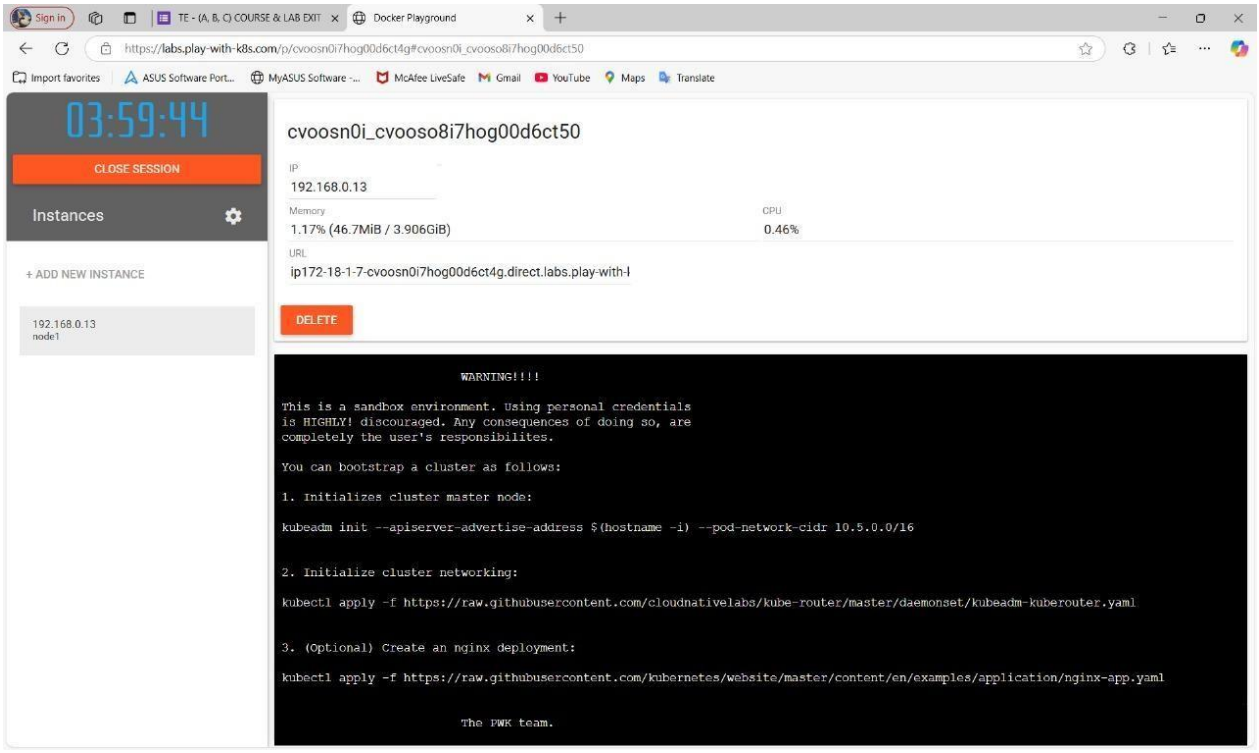
B.3 Conclusion:

(Students must write the conclusion as per the attainment of individual outcome listed above

This experiment introduced the concept of **container orchestration** using **Kubernetes**, demonstrating how it automates the deployment, scaling, and management of containerized applications. It showed how Kubernetes ensures high availability, load balancing, and fault tolerance, making it ideal for managing large-scale, production-ready applications.

OUTPUT SS:





1. Initialize cluster master node

kubeadm init --apiserver-advertise-address $(hostname -i) --pod-network-cidr 10.5.0.0/16

```
[node1 ~]$ kubeadm init --apiserver-advertise-address $(hostname -i) --pod-network-cidr 10.5.0.0/16
Initializing machine ID from random generator.
W0405 20:18:49.752843     902 initconfiguration.go:120] Usage of CRI endpoints without URL scheme is deprecated and can cause kubelet
errors in the future. Automatically prepending scheme "unix" to the "criSocket" with value "/run/docker/containerd/containerd.sock". P
lease update your configuration!
I0405 20:18:50.078901     902 version.go:256] remote version is much newer: v1.32.3; falling back to: stable-1.27
[init] Using Kubernetes version: v1.27.16
[preflight] Running pre-flight checks
[preflight] The system verification failed. Printing the output from the verification:
KERNEL_VERSION: 4.4.0-210-generic
OS: Linux
CGROUPS_CPU: enabled
CGROUPS_CPUACCT: enabled
CGROUPS_CPUSET: enabled
CGROUPS_DEVICES: enabled
CGROUPS_FREEZER: enabled
CGROUPS_MEMORY: enabled
CGROUPS_PIDS: enabled
CGROUPS_HUGETLB: enabled
CGROUPS_BLKIO: enabled
        [WARNING SystemVerification]: failed to parse kernel config: unable to load kernel module: "configs", output: "", err: exit st
atus 1
        [WARNING FileContent--proc-sys-net-bridge-bridge-nf-call-iptables]: /proc/sys/net/bridge/bridge-nf-call-iptables does not exis
t
[preflight] Pulling images required for setting up a Kubernetes cluster
```

```
CGROUPS_BLKIO: enabled
        [WARNING SystemVerification]: failed to parse kernel config: unable to load kernel module: "configs", output: "", err: exit st
atus 1
        [WARNING FileContent--proc-sys-net-bridge-bridge-nf-call-iptables]: /proc/sys/net/bridge/bridge-nf-call-iptables does not exis
t
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
W0405 20:18:50.565411     902 images.go:80] could not find officially supported version of etcd for Kubernetes v1.27.16, falling back
to the nearest etcd version (3.5.7-0)
W0405 20:18:59.052391     902 checks.go:835] detected that the sandbox image "registry.k8s.io/pause:3.6" of the container runtime is i
nconsistent with that used by kubeadm. It is recommended that using "registry.k8s.io/pause:3.9" as the CRI sandbox image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cl
uster.local node1] and IPs [10.96.0.1 192.168.0.13]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [localhost node1] and IPs [192.168.0.13 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [localhost node1] and IPs [192.168.0.13 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
```

```
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Starting the kubelet
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"
W0405 20:19:11.678922     902 images.go:80] could not find officially supported version of etcd for Kubernetes v1.27.16, falling back
to the nearest etcd version (3.5.7-0)
[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from directory "/etc/kubernetes/manifests". T
his can take up to 4m0s
[apiclient] All control plane components are healthy after 35.004169 seconds
[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config" in namespace kube-system with the configuration for the kubelets in the cluster
[upload-certs] Skipping phase. Please see --upload-certs
[mark-control-plane] Marking the node node1 as control-plane by adding the labels: [node-role.kubernetes.io/control-plane node.kuberne
tes.io/exclude-from-external-load-balancers]
[mark-control-plane] Marking the node node1 as control-plane by adding the taints [node-role.kubernetes.io/control-plane:NoSchedule]
[bootstrap-token] Using token: lctjt9.i16j00hiy5ws0y8h
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to get nodes
```

```
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 192.168.0.13:6443 --token lctjt9.i16j00hiy5ws0y8h \
        --discovery-token-ca-cert-hash sha256:0b0ff785df571740961086fd1ee063e587f5244134e71a3241b89a629cfc7800
Waiting for api server to startup
Warning: resource daemonsets/kube-proxy is missing the kubectl.kubernetes.io/last-applied-configuration annotation which is required b
y kubectl apply. kubectl apply should only be used on resources created declaratively by either kubectl create --save-config or kubect
l apply. The missing annotation will be patched automatically.
daemonset.apps/kube-proxy configured
No resources found
[node1 ~]$
```

2. Initialize cluster networking

kubectl apply -f http://www.githubusercontent.com/cloudnativelabs/kube-router/master/daemonset/kubeadm-kuberouter.yaml

```
  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 192.168.0.13:6443 --token lctjt9.i16j00hiy5ws0y8h \
       --discovery-token-ca-cert-hash sha256:0b0ff785df571740961086fd1ee063e587f5244134e71a3241b89a629cfc7800
Waiting for api server to startup
Warning: resource daemonsets/kube-proxy is missing the kubectl.kubernetes.io/last-applied-configuration annotation which is required b
y kubectl apply. kubectl apply should only be used on resources created declaratively by either kubectl create --save-config or kubect
l apply. The missing annotation will be patched automatically.
daemonset.apps/kube-proxy configured
No resources found
[node1 ~]$ ^C
[node1 ~]$ kubectl apply -f http://www.githubusercontent.com/cloudnativelabs/kube-router/master/daemonset/kubedm-kuberouter.yaml
error: unable to read URL "http://www.githubusercontent.com/cloudnativelabs/kube-router/master/daemonset/kubedm-kuberouter.yaml", serv
er reported 500 Domain Not Found, status code=500
[node1 ~]$ kubectl apply -f http://www.githubusercontent.com/cloudnativelabs/kube-router/master/daemonset/kubeadm-kuberouter.yaml
error: unable to read URL "http://www.githubusercontent.com/cloudnativelabs/kube-router/master/daemonset/kubeadm-kuberouter.yaml", ser
ver reported 500 Domain Not Found, status code=500
[node1 ~]$ kubectl apply -f http://raw.githubusercontent.com/cloudnativelabs/kube-router/master/daemonset/kubeadm-kuberouter.yaml
configmap/kube-router-cfg created
daemonset.apps/kube-router created
serviceaccount/kube-router created
clusterrole.rbac.authorization.k8s.io/kube-router created
clusterrolebinding.rbac.authorization.k8s.io/kube-router created
[node1 ~]$
```

3. Deploy NGINX app

kubectl apply -f http://raw.githubusercontent.com/kubernetes/website/master/content/en/examples/application/nginx-app.yaml

```
l apply. The missing annotation will be patched automatically.
daemonset.apps/kube-proxy configured
No resources found
[node1 ~]$ ^C
[node1 ~]$ kubectl apply -f http://www.githubusercontent.com/cloudnativelabs/kube-router/master/daemonset/kubedm-kuberouter.yaml
error: unable to read URL "http://www.githubusercontent.com/cloudnativelabs/kube-router/master/daemonset/kubedm-kuberouter.yaml", serv
er reported 500 Domain Not Found, status code=500
[node1 ~]$ kubectl apply -f http://www.githubusercontent.com/cloudnativelabs/kube-router/master/daemonset/kubeadm-kuberouter.yaml
error: unable to read URL "http://www.githubusercontent.com/cloudnativelabs/kube-router/master/daemonset/kubeadm-kuberouter.yaml", ser
ver reported 500 Domain Not Found, status code=500
[node1 ~]$ kubectl apply -f http://raw.githubusercontent.com/cloudnativelabs/kube-router/master/daemonset/kubeadm-kuberouter.yaml
configmap/kube-router-cfg created
daemonset.apps/kube-router created
serviceaccount/kube-router created
clusterrole.rbac.authorization.k8s.io/kube-router created
clusterrolebinding.rbac.authorization.k8s.io/kube-router created
[node1 ~]$ kubectl apply -f http://raw.githubusercontent.com/kubernetes/master/content/an/examples/application/nginx-app.yaml
error: unable to read URL "http://raw.githubusercontent.com/kubernetes/master/content/an/examples/application/nginx-app.yaml", server
reported 404 Not Found, status code=404
[node1 ~]$ kubectl apply -f http://raw.githubusercontent.com/kubernetes/master/content/en/examples/application/nginx-app.yaml
error: unable to read URL "http://raw.githubusercontent.com/kubernetes/master/content/en/examples/application/nginx-app.yaml", server
reported 404 Not Found, status code=404
[node1 ~]$ kubectl apply -f http://raw.githubusercontent.com/kubernetes/website/master/content/en/examples/application/nginx-app.yaml
service/my-nginx-svc created
deployment.apps/my-nginx created
[node1 ~]$
```