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

* 1. **Aim:**

To study and implement container orchestration using Kubernetes .

### Prerequisite:

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

### 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

### Outcome: (LO 4)

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

### 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

1. 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

1. 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

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

$ sudoswapoff -a

1. 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

1. 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

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

$ sudokubectl get pods –all-namespaces

1. 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

1. 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)

* 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.

# 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

# 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. |

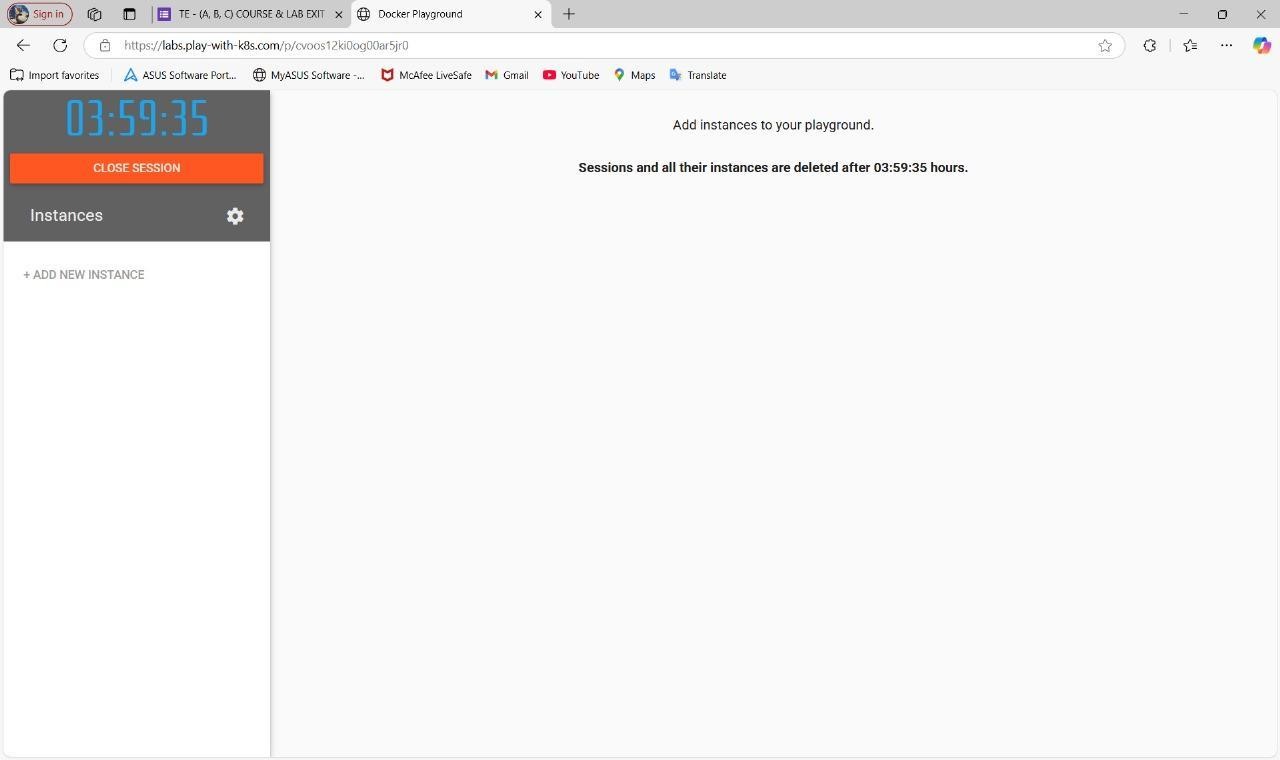
|  |  |
| --- | --- |
| **LoadBalancer** | Provisions a **cloud load balancer** and exposes the service to the  internet. |

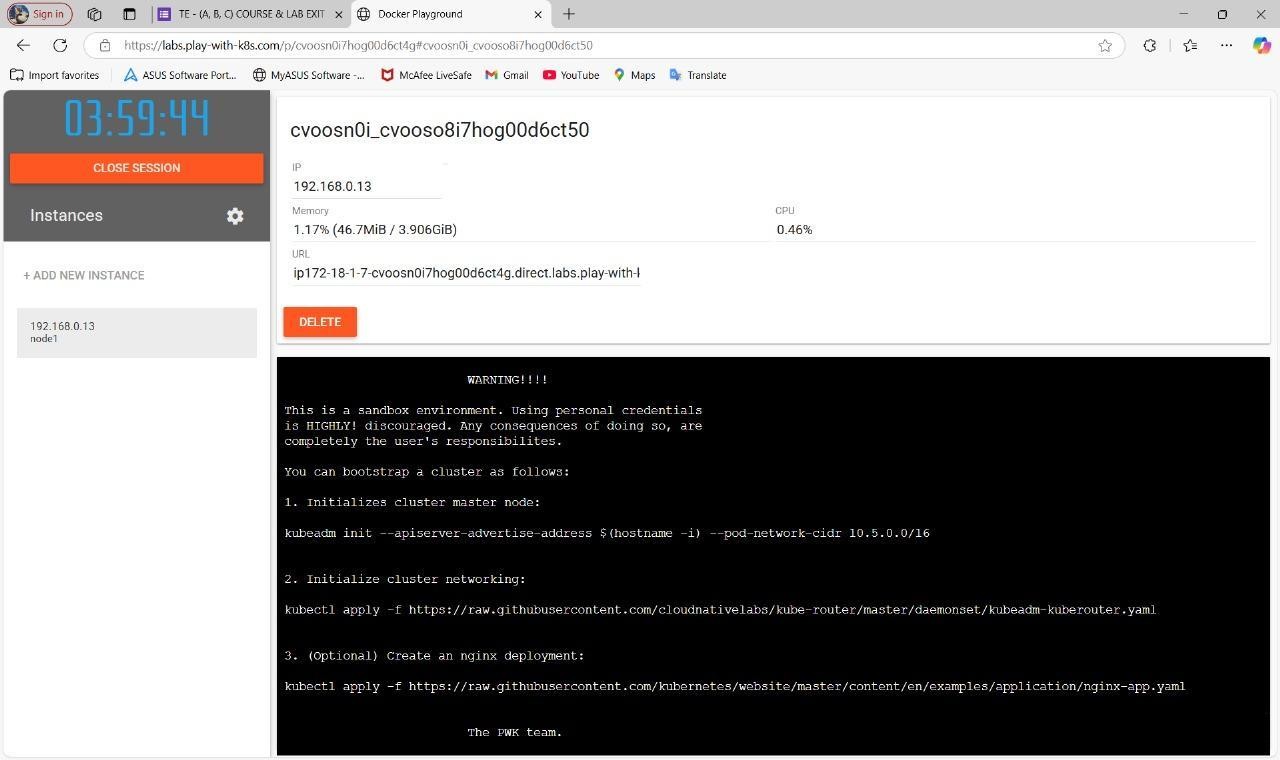
### B.3 Conclusion:

*(****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.

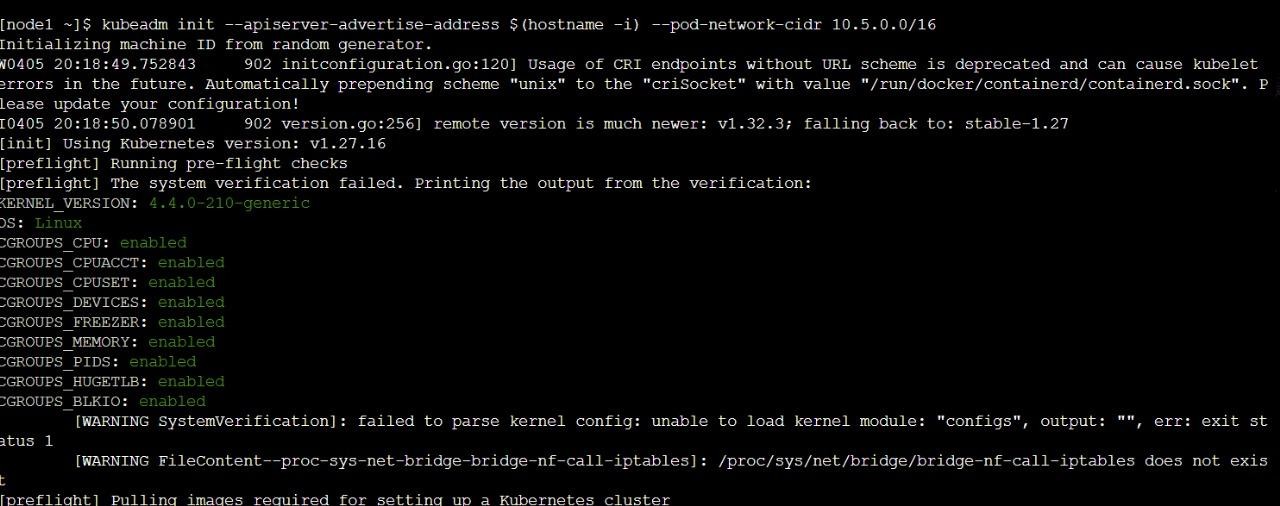
## OUTPUT SS:

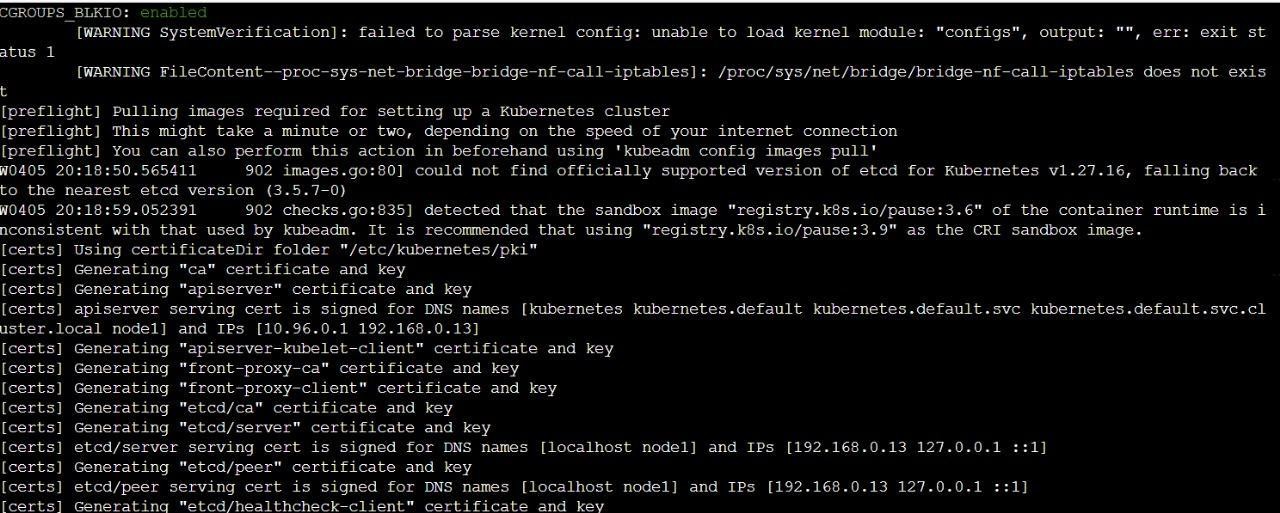
****

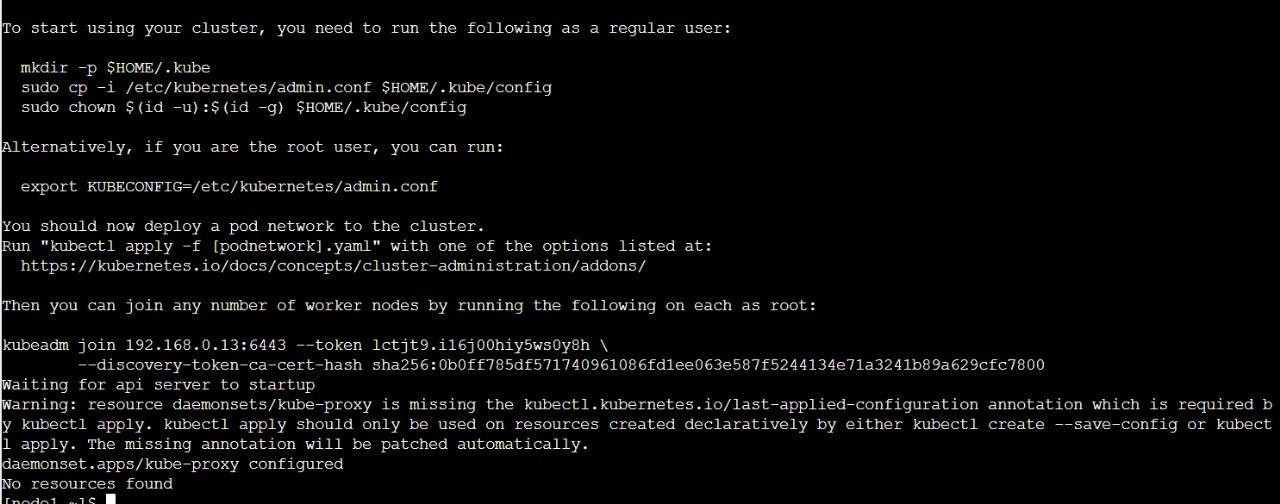
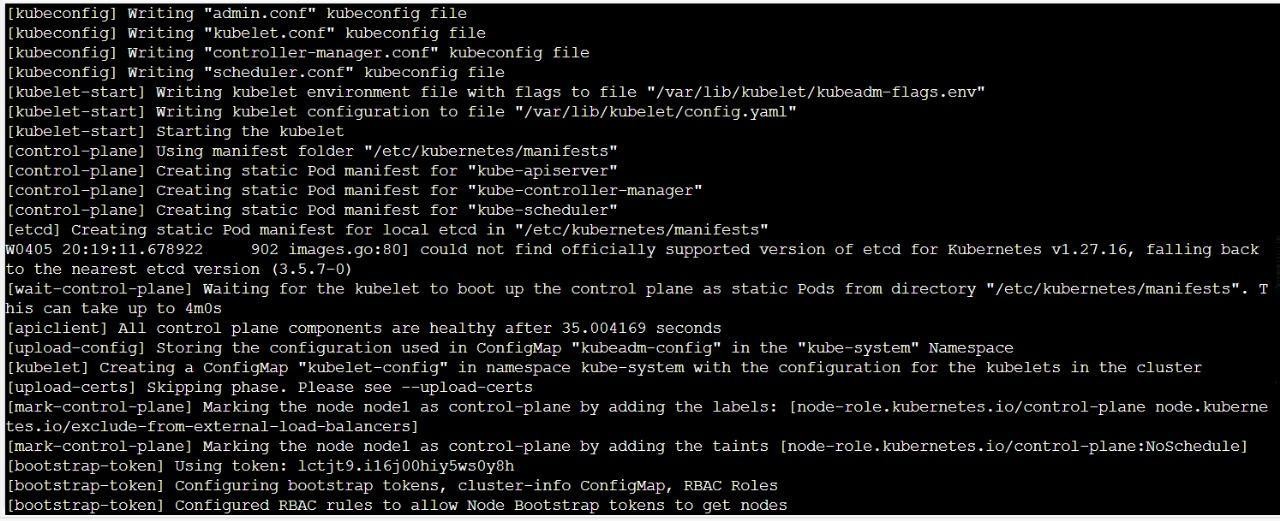


# 1. Initialize cluster master node

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

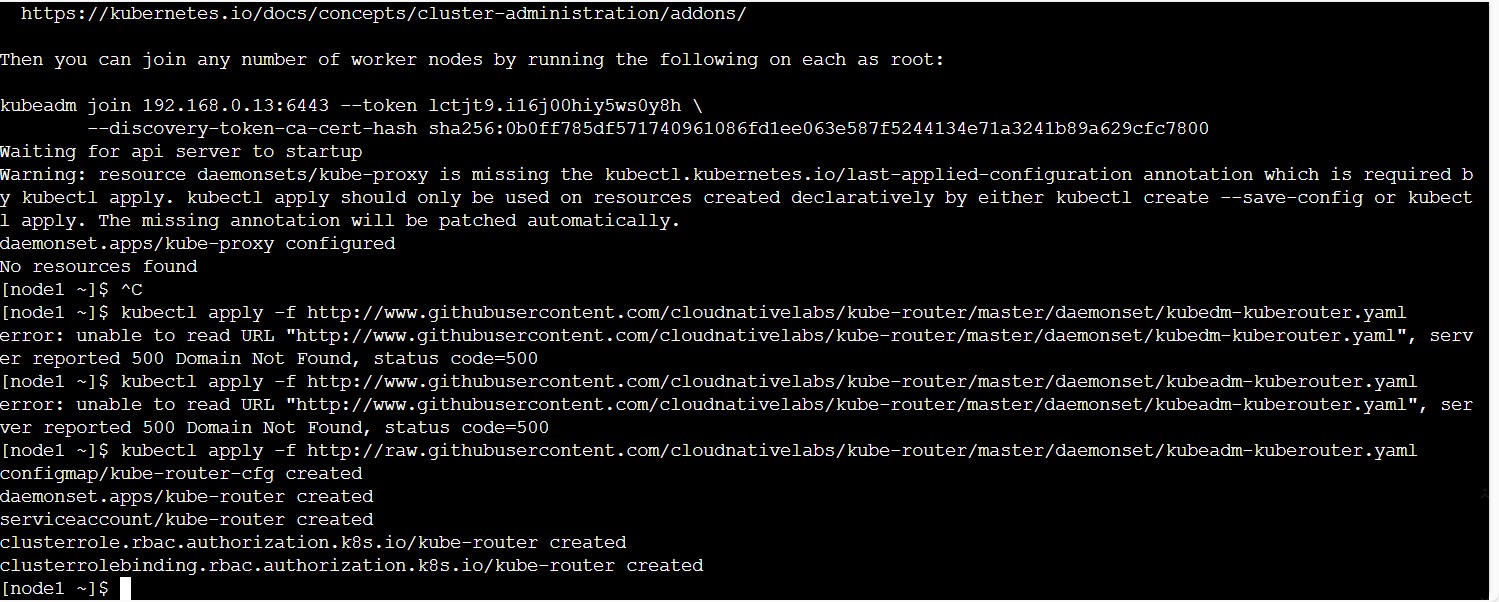






# 2. Initialize cluster networking

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



# 3. Deploy NGINX app kubectl apply -f

<http://raw.githubusercontent.com/kubernetes/website/master/content/en/examples/ap> plication/nginx-app.yaml

