MOHAMED SHAHUL ANAS S

22ISR027

Overview

This project demonstrates the implementation of a DevOps pipeline using Docker, Kubernetes, and Jenkins. It automates the build, testing, and deployment of an application.

Technologies Used

- **Docker**: Containerization of the application
- **Jenkins**: CI/CD automation
- **Kubernetes**: Deployment and management of containers
- **Shell Scripting**: Automating deployment tasks
- **YAML**: Configuration management for Kubernetes

Setup Instructions

Prerequisites

Ensure the following tools are installed:

- Docker
- Kubernetes (Minikube or a cluster)
- Jenkins
- Git

Dockerization

Build and run the application using Docker:

```
docker build -t devopstask04 .
docker run -p 80:80 devopstask04
```

CI/CD Pipeline

The **Jenkinsfile** automates:

1. **Cloning the repository**
2. **Building the Docker image**
3. **Pushing the image to Docker Hub**
4. **Deploying to Kubernetes**

Kubernetes Deployment

Apply the Kubernetes configurations:

kubectl apply -f deployment.yaml
kubectl apply -f service.yaml

Check running pods and services:

kubectl get pods
kubectl get services

YAML File Usage

The .yaml files define Kubernetes configurations:

- **deployment.yaml**: Describes the deployment, including the number of replicas, container specifications, and update strategies.
- **service.yaml**: Defines how the application is exposed, including the type of service (e.g., ClusterIP, NodePort, or LoadBalancer).

Deployment Script

The deploy.sh script automates the Kubernetes deployment:

chmod +x deploy.sh
./deploy.sh

Accessing the Application

Find the service IP using:

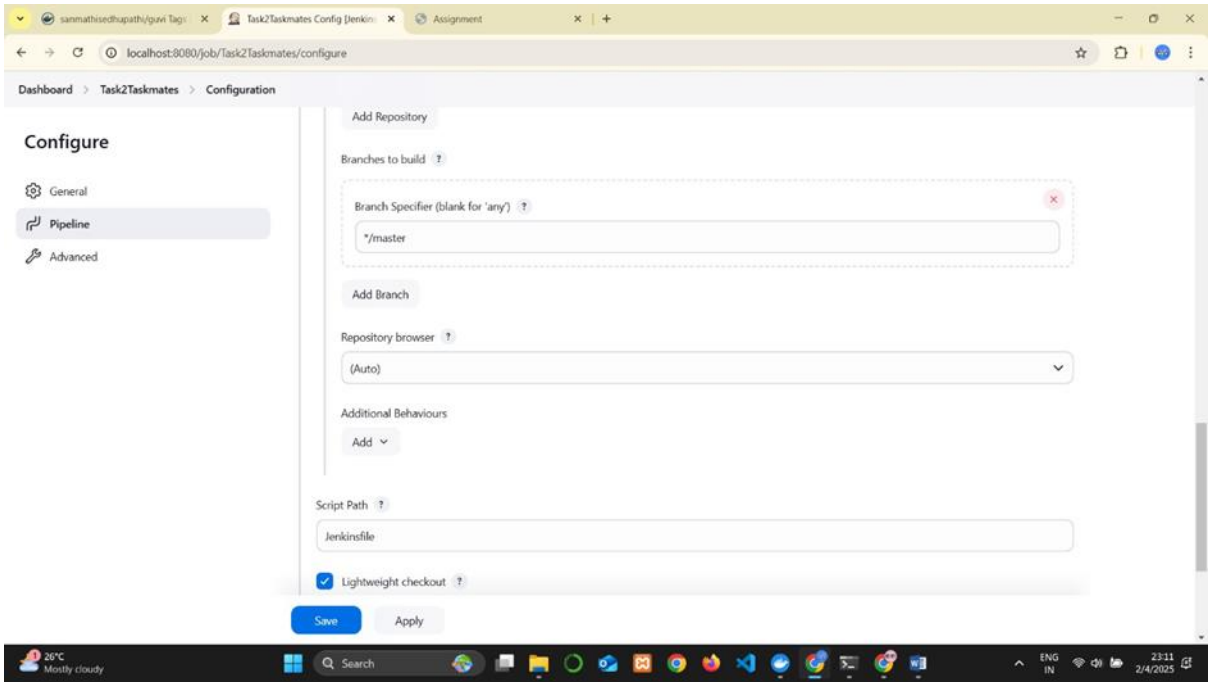
kubectl get svc

Then access the application in the browser or via curl:

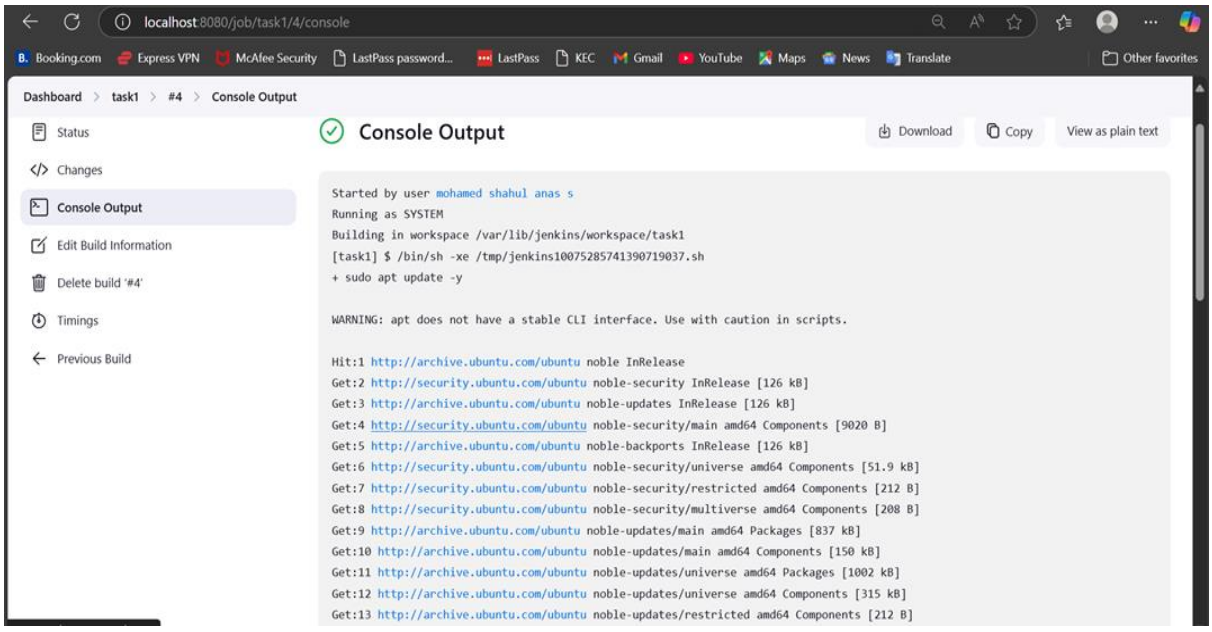
http://<EXTERNAL-IP>:80

Conclusion

This DevOps pipeline ensures continuous integration and delivery, allowing automated testing and seamless deployment.



CONSOLE OUTPUT:



```
maddy@Maddy:~$ kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP          11m
maddy        NodePort    10.102.24.48    <none>        80:31173/TCP     29s
```

```
maddy@Maddy:~$ minikube service maddy
|-----------|-------|-------------|---------------------------|
| NAMESPACE | NAME  | TARGET PORT |            URL            |
|-----------|-------|-------------|---------------------------|
| default   | maddy |          80 | http://192.168.49.2:31173 |
|-----------|-------|-------------|---------------------------|
🏃  Starting tunnel for service maddy.
|-----------|-------|-------------|------------------------|
| NAMESPACE | NAME  | TARGET PORT |          URL           |
|-----------|-------|-------------|------------------------|
| default   | maddy |             | http://127.0.0.1:36209 |
|-----------|-------|-------------|------------------------|
🎉  Opening service default/maddy in default browser...
👉  http://127.0.0.1:36209
❗  Because you are using a Docker driver on linux, the terminal needs to be open to run it.
```