DAY 3 & 4

KUBERNETES

Step 1: create a directory

```
mkdir E-commerce && cd E-commerce
```

Create a backend

```
mkdir backend && cd backend
```

```
root@Sample:~# mkdir E-commerce
root@Sample:~# cd E-commerce
root@Sample:~/E-commerce# mkdir backend
root@Sample:~/E-commerce# cd backend
root@Sample:~/E-commerce/backend# nano products.csv
root@Sample:~/E-commerce/backend# nano app.py
root@Sample:~/E-commerce/backend# nano app.py
root@Sample:~/E-commerce/backend# nano requirements.txt
root@Sample:~/E-commerce/backend# nano Dockerfile
root@Sample:~/E-commerce/backend# nano requirements.txt
root@Sample:~/E-commerce/backend# nano app.py
root@Sample:~/E-commerce/backend# ^C
root@Sample:~/E-commerce/backend# nano docker-compose.yml
root@Sample:~/E-commerce/backend# docker build -t backend:latest .
```

Create products.csv

```
nano products.csv

id,name,price,quantity
1,Smartphone,15000,25
2,Laptop,45000,15
3,Headphones,1500,50
4,Smartwatch,8000,30
5,Tablet,20000,20
6,Wireless Mouse,700,100
7,Bluetooth Speaker,1200,60
8,External Hard Drive,4000,40
9,USB Flash Drive,500,150
10,Monitor,10000,10
```

Create `app.py`

```
nano app.py
from flask import Flask
import pandas as pd

app = Flask(__name__)

@app.route("/products", methods=['GET'])
def read_data():
    df = pd.read_csv("products.csv")  # Ensure products.csv exists
    json_data = df.to_json()
    return json_data
```

```
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5050)
```

Create `requirements.txt`

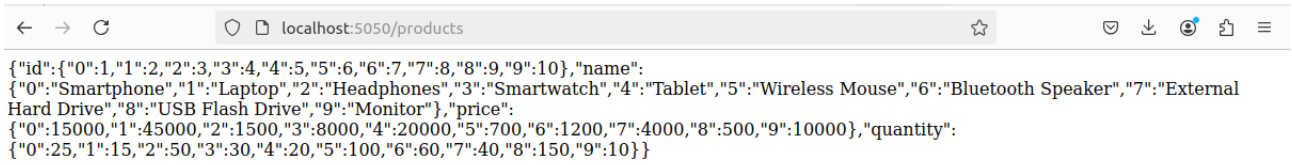
```
nano requirements.txt

flask
pandas
```

Create `Dockerfile`

```
nano Dockerfile
FROM python:3.11
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY . .
EXPOSE 5050
CMD ["python", "app.py"]
```

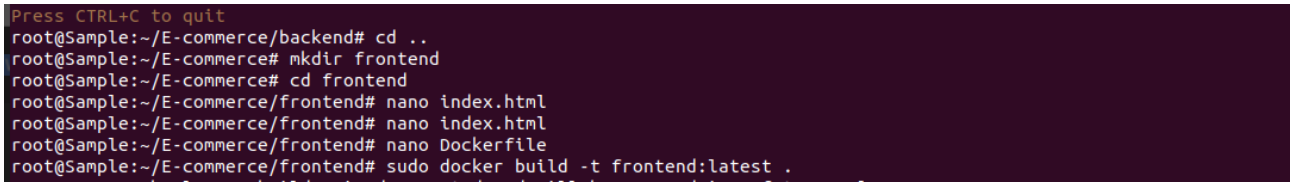
Build & Run Backend Container

```
docker build -t backend:latest .
docker run -itd -p 5050:5050 backend
docker logs $(docker ps -q --filter "ancestor=backend")
```



Create a Frontend

```
cd ..
mkdir frontend && cd frontend
```



Create `index.html`

```
nano index.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>E-Commerce Store</title>
    <script>
        async function fetchProducts() {
            const response = await fetch("http://localhost:5050/products");
            const products = await response.json();
            let output = "<h2>Product List</h2><ul>";
            for (const id in products.name) {
                output += `<li>${products.name[id]} -
$${products.price[id]}</li>`;
            }
            output += "</ul>";
            document.getElementById("product-list").innerHTML = output;
        }
    </script>
</head>
<body onload="fetchProducts()">
    <h1>Welcome to Our Store</h1>
    <div id="product-list">Loading...</div>
</body>
</html>
```

Create `Dockerfile`

```
nano Dockerfile

FROM nginx:alpine
COPY index.html /usr/share/nginx/html/index.html
```

Build & Run Frontend Container

```
docker build -t frontend:latest .
```

Step 2. Kubernetes Deployment

```
cd ..
mkdir k8s && cd k8s
```

Backend Deployment (`backend-deployment.yaml`)

```
nano backend-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend
spec:
  replicas: 1
```

```
  selector:
    matchLabels:
      app: backend
  template:
    metadata:
      labels:
        app: backend
    spec:
      containers:
      - name: backend
        image: backend:latest
        ports:
        - containerPort: 5050
```

Frontend Deployment (`frontend-deployment.yaml`)

```
nano frontend-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
spec:
  replicas: 1
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
      - name: frontend
        image: frontend:latest
        ports:
        - containerPort: 3000
```

Frontend & Backend (`service.yaml`)

```
nano service.yaml
apiVersion: v1
kind: Service
metadata:
  name: backend-service
spec:
  selector:
    app: backend
  ports:
    - protocol: TCP
      port: 5050
      targetPort: 5050
  type: ClusterIP
---
apiVersion: v1
kind: Service
metadata:
  name: frontend-service
spec:
  selector:
```

```
    app: frontend
  ports:
    - protocol: TCP
      port: 3000
      targetPort: 3000
  type: NodePort
```

ConfigMap (`configmap.yaml`)

```
nano configmap.yaml
```

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: backend-config
data:
  DATABASE_FILE: "/backend/products.csv"
```

Step 3. Installing Kubernetes

Install Docker

```
sudo apt update
sudo apt install -y docker.io
```

Verify Docker Installation

```
docker --version
```

Enable and Start Docker

```
sudo systemctl enable docker
sudo systemctl start docker
sudo systemctl status docker
```

Download Kubectl

```
curl -LO https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl
```

Install Kubectl

```
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
```

Verify Installation

```
kubectl version --client
```

Step 4 :Installing Minikube

Download Minikube

```
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-
linux-amd64
```

Install Minikube

```
sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

Verify Minikube Installation

```
minikube version
```

Starting Minikube

```
minikube start --driver=docker
```

Check the status of the Minikube cluster:

```
minikube status
```

Verify that Kubernetes is running:

```
kubectl get nodes
```

output:

```
NAME       STATUS   ROLES                  AGE     VERSION
minikube   Ready    control-plane,master   3m24s   v1.32.0
```

Enabling the Kubernetes Dashboard (Optional)

```
minikube dashboard
```

This will open a web browser with the Kubernetes dashboard

7. Managing Minikube

Stopping Minikube

```
minikube stop
```

Deleting Minikube Cluster

```
minikube delete
```

Checking Running Services

```
kubectl get services
```

Troubleshooting Tips

1. If Minikube Fails to Start

```
minikube delete
minikube start --driver=docker
```

2. If Kubectl Cannot Connect to Minikube

Check if Minikube is running:

```
minikube status
```

If it's stopped, restart it:

```
minikube start
```

3. If Kubernetes Services Are Not Accessible

Use port forwarding to access a service:

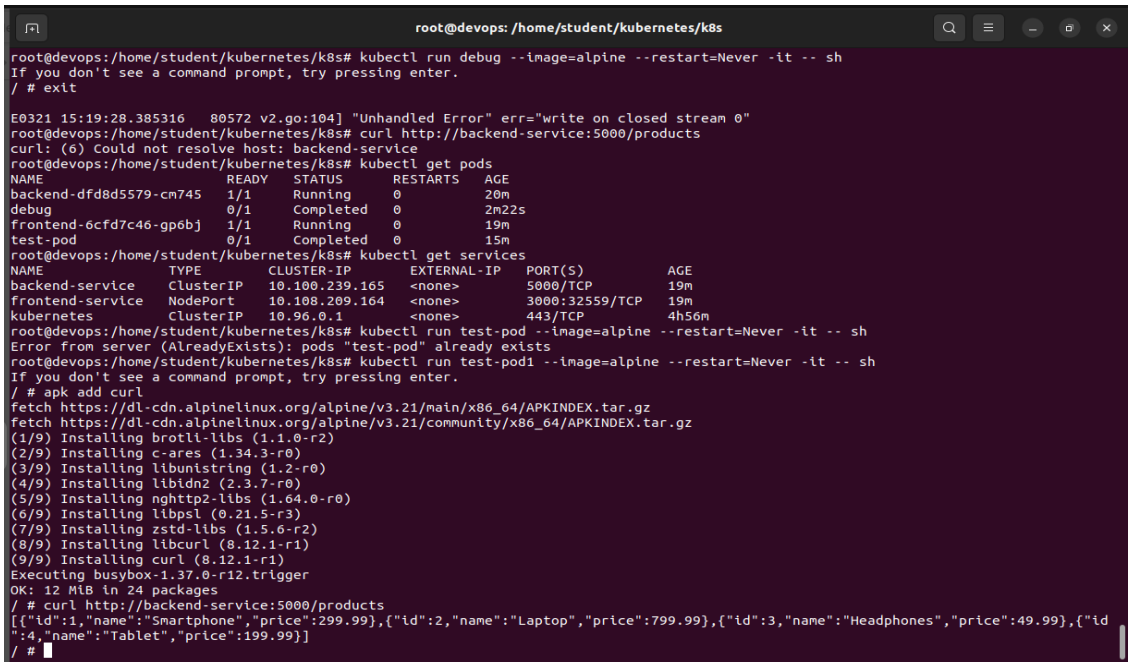
```
kubectl port-forward svc/<service-name> <local-port>:<service-port>
```

Example:

```
kubectl port-forward svc/backend-service 5000:5000
```

Then access the service at:

http://localhost:5000

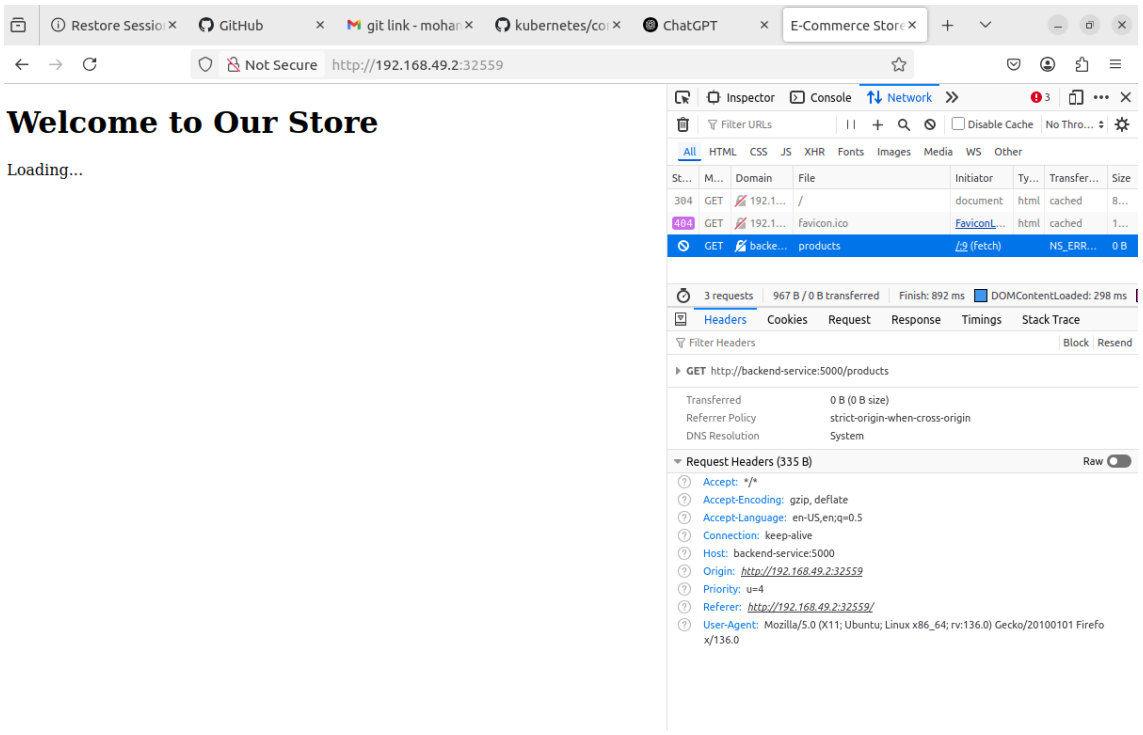


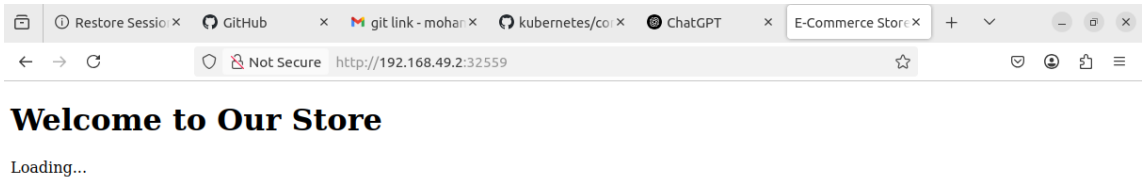
9. Deploying in Minikube

```
minikube start
eval $(minikube docker-env)
kubectl apply -f k8s/
kubectl get pods
```

```
kubectl get services
minikube service frontend-service --url
```

Open the displayed URL in a browser to view the application.





Day 5

Configuring Jenkins Pipeline

Step 1: Create a Jenkinsfile

nano Jenkinsfile

Step 2: Add Jenkins Pipeline Code

```
pipeline {
    agent any
    environment {
        BACKEND_IMAGE = "sujisuki/backend-app:latest"
        FRONTEND_IMAGE = "sujisuki/frontend-app:latest"
        BACKEND_CONTAINER = "backend-running-app"
        FRONTEND_CONTAINER = "frontend-running-app"
        REGISTRY_CREDENTIALS = "docker_suji"
    }
    stages {
stage('Checkout Code') {
            steps {
withCredentials([usernamePassword(credentialsId: 'github_suji',
usernameVariable: 'GIT_USER', passwordVariable: 'GIT_TOKEN')]) {
                    git url:
"https://$GIT_USER:$GIT_TOKEN@github.com/RithanyaR/Jenkins_E-commerce.git",
branch: 'main'
                }
            }
        }
stage('Build Docker Images') {
            parallel {
stage('Build Backend Image') {
                    steps {
dir('backend') {
sh 'docker build -t $BACKEND_IMAGE .'
                        }
                    }
                }
stage('Build Frontend Image') {
                    steps {
dir('frontend') {
sh 'docker build -t $FRONTEND_IMAGE .'
                        }
                    }
                }
            }
        }
stage('Login to Docker Registry') {
            steps {
withCredentials([usernamePassword(credentialsId: 'docker_suji',
usernameVariable: 'DOCKER_USER', passwordVariable: 'DOCKER_PASS')]) {
sh 'echo $DOCKER_PASS | docker login -u $DOCKER_USER --password-stdin'
                }
```

```
            }
        }
stage('Push Images to Docker Hub') {
            parallel {
stage('Push Backend Image') {
                    steps {
sh 'docker push $BACKEND_IMAGE'
                    }
                }
stage('Push Frontend Image') {
                    steps {
sh 'docker push $FRONTEND_IMAGE'
                    }
                }
            }
        }
stage('Stop & Remove Existing Containers') {
            steps {
                script {
sh '''
                    docker stop $BACKEND_CONTAINER $FRONTEND_CONTAINER ||
true
                    docker rm $BACKEND_CONTAINER $FRONTEND_CONTAINER ||
true
                    '''
                }
            }
        }
stage('Run Containers') {
            parallel {
stage('Run Backend Container') {
                    steps {
sh 'docker run -d -p 5000:5000 --name $BACKEND_CONTAINER $BACKEND_IMAGE'
                    }
                }
stage('Run Frontend Container') {
                    steps {
sh 'docker run -d -p 3000:3000 --name $FRONTEND_CONTAINER $FRONTEND_IMAGE'
                    }
                }
            }
        }
    }
    post {
        success {
            echo " Deployment successful! Backend and Frontend are
running."
        }
        failure {
            echo " Deployment failed! Check logs for errors."
        }
    }
```

Pushing the Project to GitHub

Step 1: Clone the Repository

git clone https://github.com/ RithanyaR/Jenkins_E-commerce.git

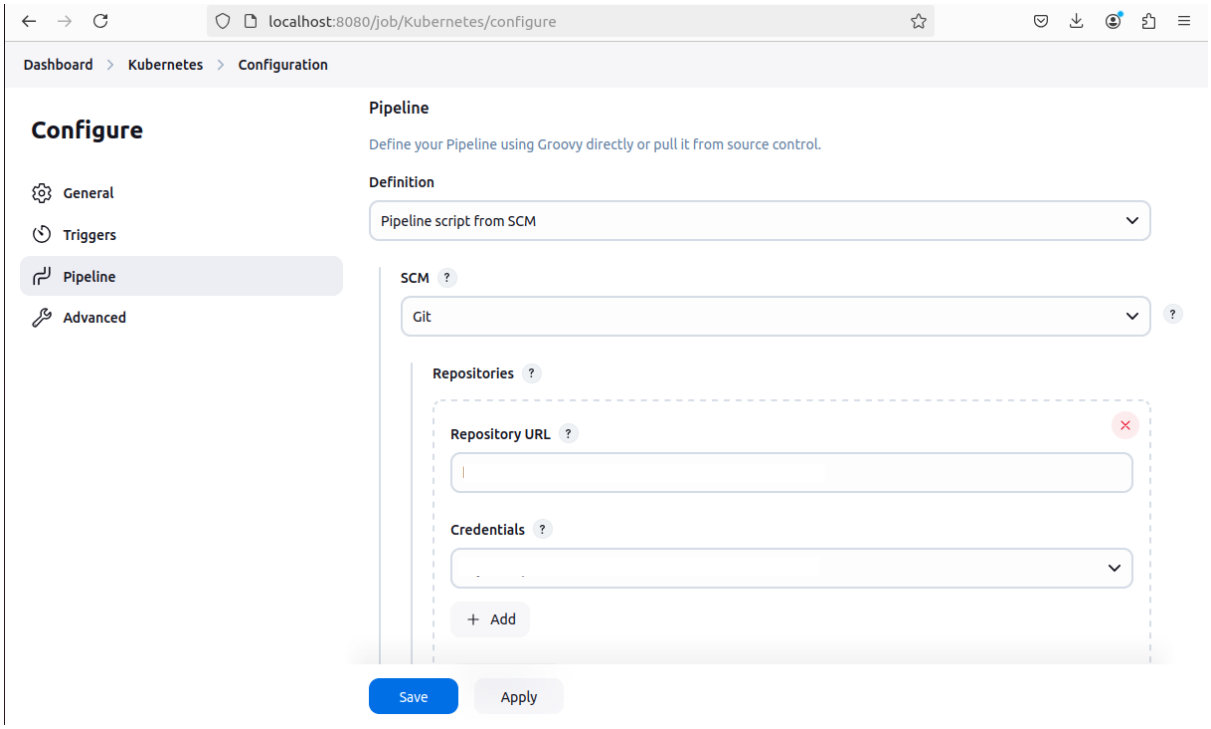
Step 2: Add and Commit the Changes

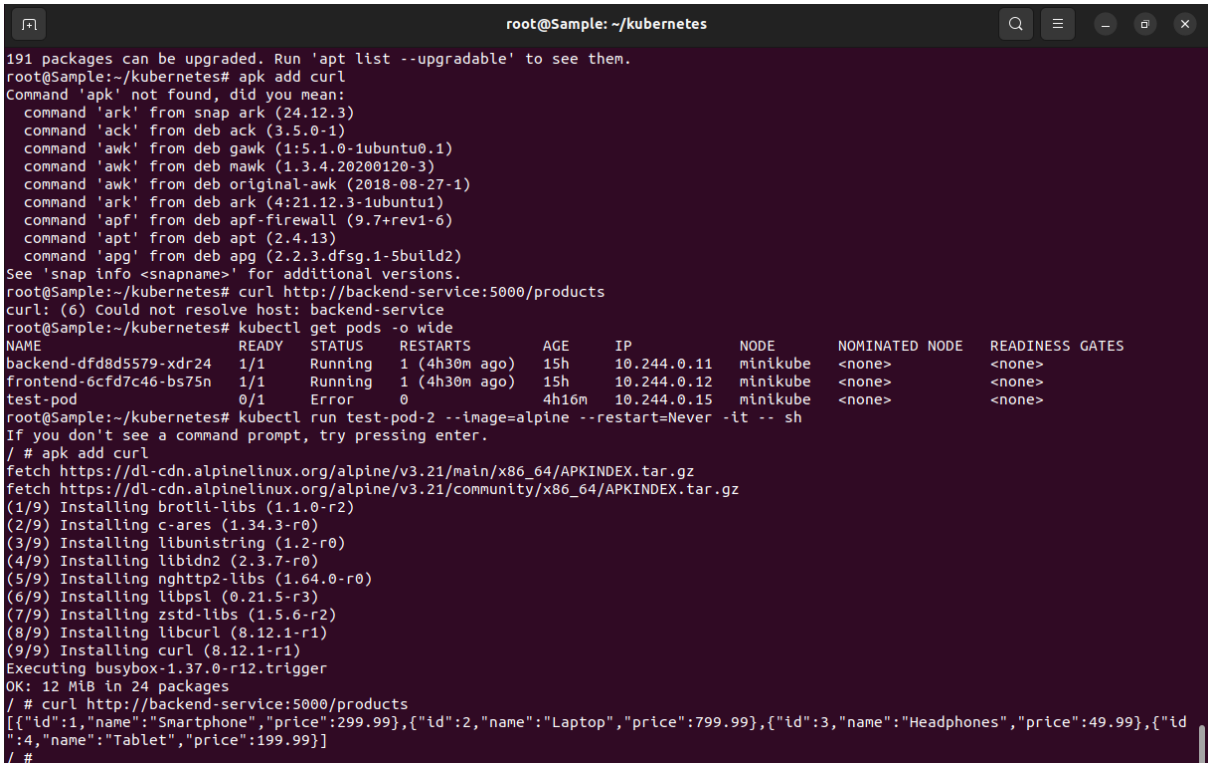
git add --all
git commit -m "Kubernetes"

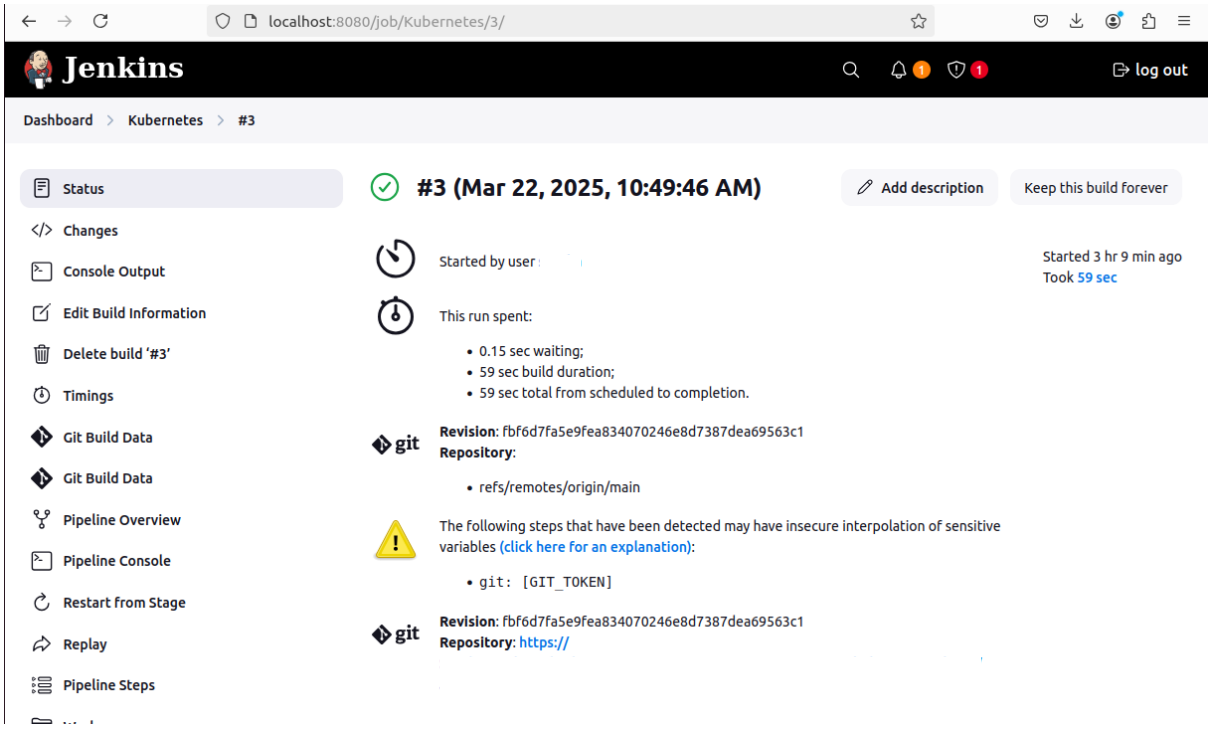
Step 3: Push to GitHub

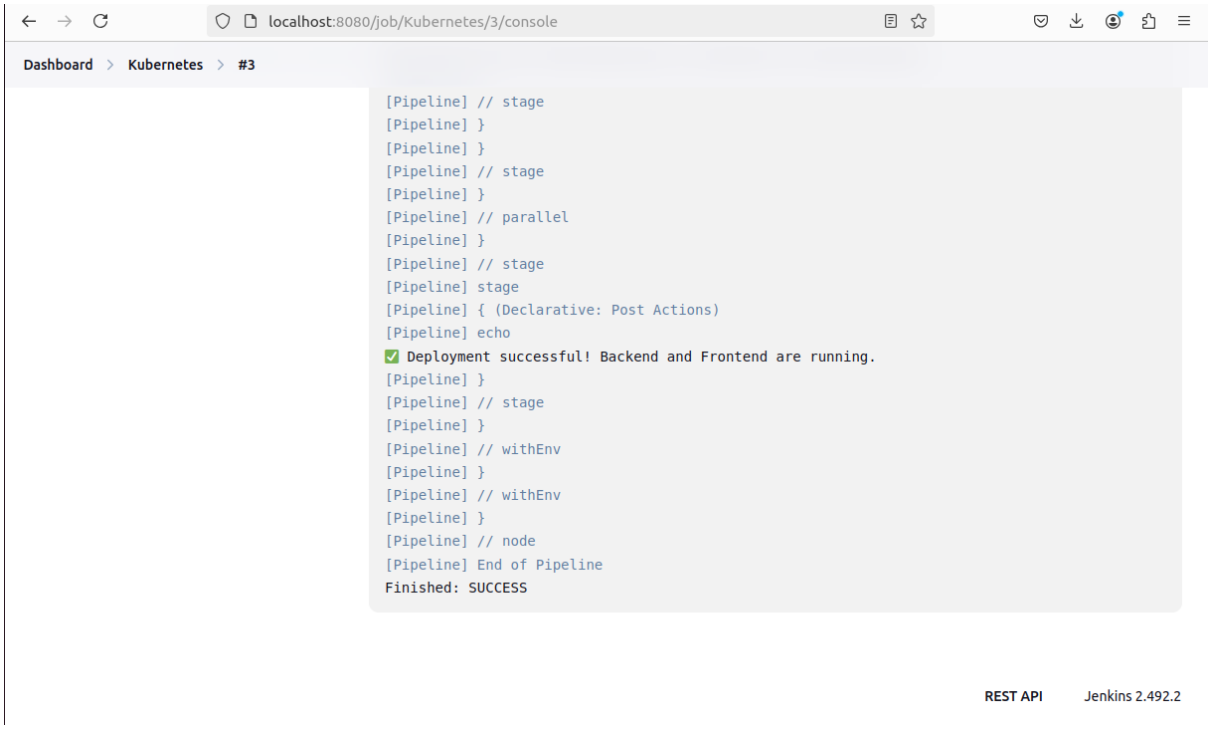
git push origin main

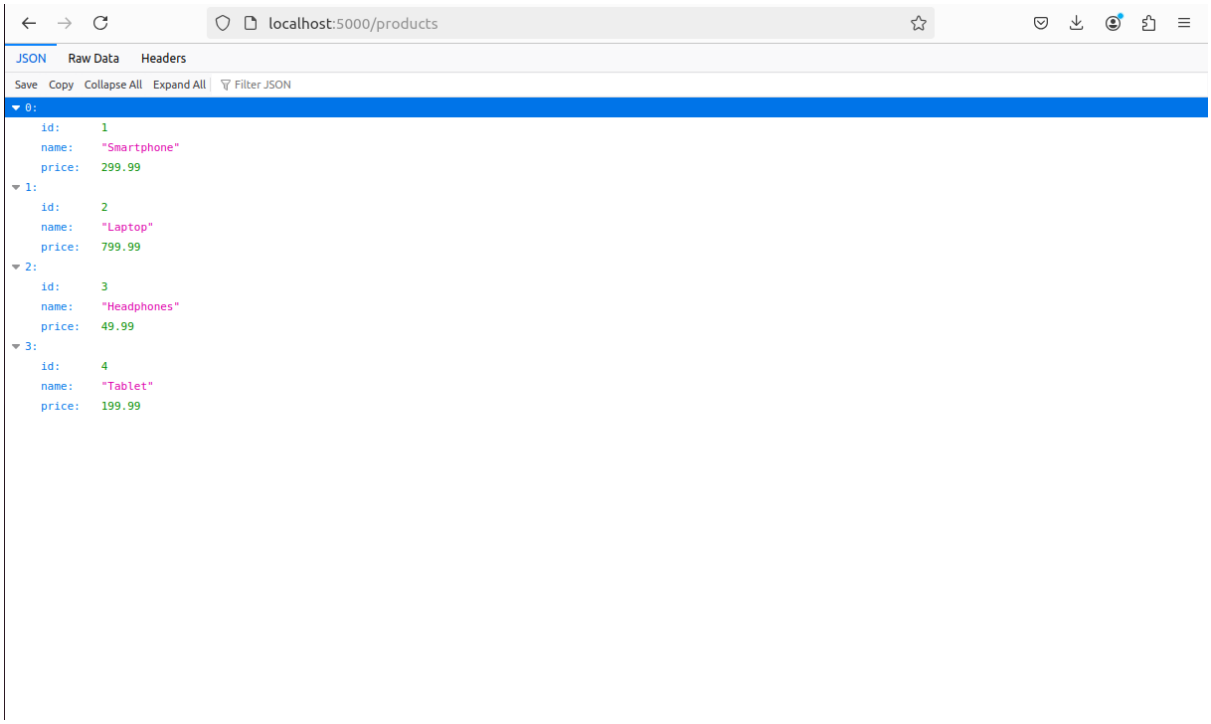
Running Jenkins Build






localhost:8080/job/Kubernetes/3/
Jenkins
log out
Dashboard > Kubernetes > #3
Status
Changes
Console Output
Edit Build Information
Delete build '#3'
Timings
Git Build Data
Git Build Data
Pipeline Overview
Pipeline Console
Restart from Stage
Replay
Pipeline Steps
#3 (Mar 22, 2025, 10:49:46 AM)
Add description
Keep this build forever
Started by user
Started 3 hr 9 min ago
Took 59 sec
This run spent:
0.15 sec waiting;
59 sec build duration;
59 sec total from scheduled to completion.
Revision: fbf6d7fa5e9fea834070246e8d7387dea69563c1
Repository:
refs/remotes/origin/main
The following steps that have been detected may have insecure interpolation of sensitive variables (click here for an explanation):
git: [GIT_TOKEN]
Revision: fbf6d7fa5e9fea834070246e8d7387dea69563c1
Repository: https://


localhost:8080/job/Kubernetes/3/console
Dashboard > Kubernetes > #3
[Pipeline] // stage
[Pipeline] }
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // parallel
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] echo
✅ Deployment successful! Backend and Frontend are running.
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
REST API
Jenkins 2.492.2


localhost:5000/products
JSON
Raw Data
Headers
Save
Copy
Collapse All
Expand All
Filter JSON
0:
id: 1
name: "Smartphone"
price: 299.99
1:
id: 2
name: "Laptop"
price: 799.99
2:
id: 3
name: "Headphones"
price: 49.99
3:
id: 4
name: "Tablet"
price: 199.99