

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
```

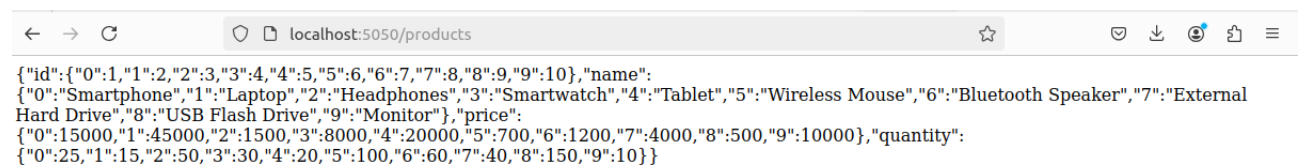
```
flask
pandas
```

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

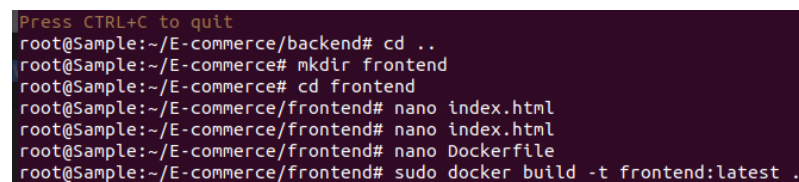
```
docker build -t backend:latest .
docker run -itd -p 5050:5050 backend
docker logs $(docker ps -q --filter "ancestor=backend")
```



```
{
  "id": {
    "0": 1,
    "1": 2,
    "2": 3,
    "3": 4,
    "4": 5,
    "5": 6,
    "6": 7,
    "7": 8,
    "8": 9,
    "9": 10
  },
  "name": {
    "0": "Smartphone",
    "1": "Laptop",
    "2": "Headphones",
    "3": "Smartwatch",
    "4": "Tablet",
    "5": "Wireless Mouse",
    "6": "Bluetooth Speaker",
    "7": "External Hard Drive",
    "8": "USB Flash Drive",
    "9": "Monitor"
  },
  "price": {
    "0": 15000,
    "1": 45000,
    "2": 1500,
    "3": 8000,
    "4": 20000,
    "5": 700,
    "6": 1200,
    "7": 4000,
    "8": 500,
    "9": 10000
  },
  "quantity": {
    "0": 25,
    "1": 15,
    "2": 50,
    "3": 30,
    "4": 20,
    "5": 100,
    "6": 60,
    "7": 40,
    "8": 150,
    "9": 10
  }
}
```

Create a Frontend

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



```
Press CTRL+C to quit
root@Sample:~/E-commerce/backend# cd ..
root@Sample:~/E-commerce# mkdir frontend
root@Sample:~/E-commerce# cd frontend
root@Sample:~/E-commerce/frontend# nano index.html
root@Sample:~/E-commerce/frontend# nano index.html
root@Sample:~/E-commerce/frontend# nano Dockerfile
root@Sample:~/E-commerce/frontend# sudo docker build -t frontend:latest .
```

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:

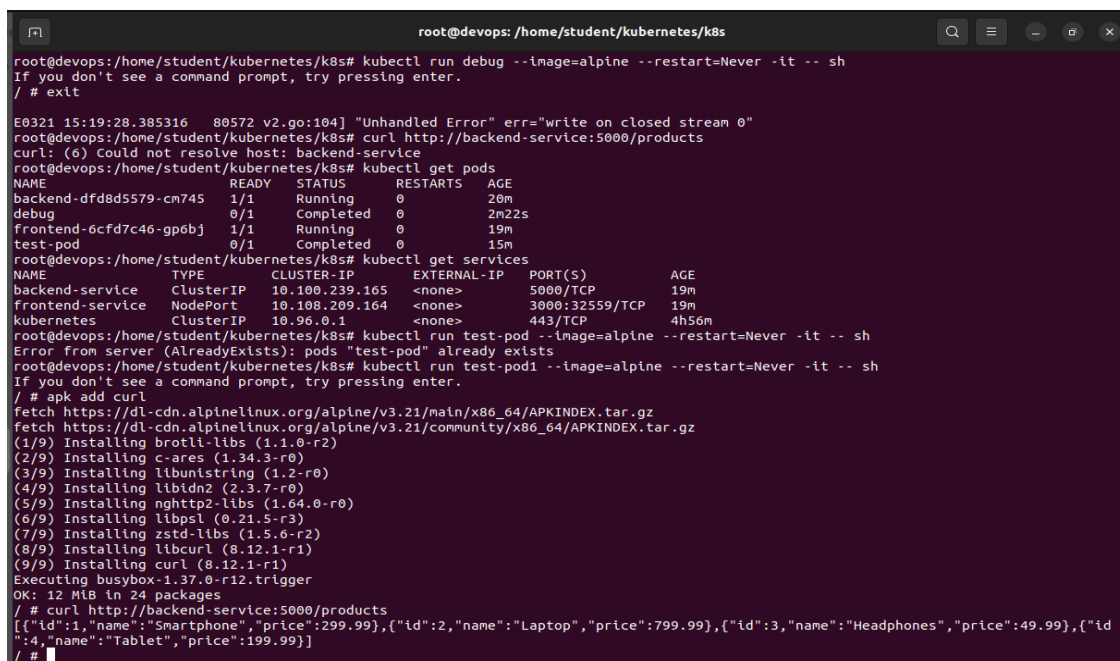
```
kubectl port-forward svc/<service-name> <local-port>:<service-port>
```

Example:

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

Then access the service at:

<http://localhost:5000>

A terminal window titled 'root@devops: /home/student/kubernetes/k8s' showing a series of commands and their outputs. The user runs 'kubectl run debug --image=alpine --restart=Never -it -- sh', then 'curl http://backend-service:5000/products' which fails with a host resolution error. They then run 'kubectl get pods' and 'kubectl get services' to check the cluster state. Finally, they run 'kubectl run test-pod --image=alpine --restart=Never -it -- sh' and use 'apk add curl' to install curl, then 'curl http://backend-service:5000/products' which successfully returns a JSON array of product data.

```
root@devops: /home/student/kubernetes/k8s# kubectl run debug --image=alpine --restart=Never -it -- sh
If you don't see a command prompt, try pressing enter.
/ # exit

E0321 15:19:28.385316 80572 v2.go:104] "Unhandled Error" err="write on closed stream 0"
root@devops: /home/student/kubernetes/k8s# curl http://backend-service:5000/products
curl: (6) could not resolve host: backend-service
root@devops: /home/student/kubernetes/k8s# kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
backend-dfd8d5579-cm745  1/1     Running   0           20m
debug          0/1     Completed 0           2m22s
frontend-6cfd7c46-gp6bj  1/1     Running   0           19m
test-pod       0/1     Completed 0           15m
root@devops: /home/student/kubernetes/k8s# kubectl get services
NAME          TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
backend-service  ClusterIP    10.100.239.165  <none>        5000/TCP          19m
frontend-service  NodePort     10.108.209.164  <none>        3000:32559/TCP    19m
kubernetes      ClusterIP    10.96.0.1      <none>        443/TCP           4h56m
root@devops: /home/student/kubernetes/k8s# kubectl run test-pod --image=alpine --restart=Never -it -- sh
Error from server (AlreadyExists): pods "test-pod" already exists
root@devops: /home/student/kubernetes/k8s# kubectl run test-pod1 --image=alpine --restart=Never -it -- sh
If you don't see a command prompt, try pressing enter.
/ # apk add curl
fetch https://dl-cdn.alpinelinux.org/alpine/v3.21/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.21/community/x86_64/APKINDEX.tar.gz
(1/9) Installing brotli-libs (1.1.0-r2)
(2/9) Installing c-ares (1.34.3-r0)
(3/9) Installing libunistring (1.2-r0)
(4/9) Installing libidn2 (2.3.7-r0)
(5/9) Installing nghttp2-libs (1.64.0-r0)
(6/9) Installing libpsl (0.21.5-r3)
(7/9) Installing zstd-libs (1.5.6-r2)
(8/9) Installing libcurl (8.12.1-r1)
(9/9) Installing curl (8.12.1-r1)
Executing busybox-1.37.0-r12.trigger
OK: 12 MiB in 24 packages
/ # curl http://backend-service:5000/products
[{"id":1,"name":"Smartphone","price":299.99}, {"id":2,"name":"Laptop","price":799.99}, {"id":3,"name":"Headphones","price":49.99}, {"id":4,"name":"Tablet","price":199.99}]
/ #
```

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.

