**E-Commerce App Deployment Using Kubernetes**

This guide provides step-by-step instructions for setting up a simple e-commerce application using Flask (Python) for the backend and Nginx for the frontend, and deploying it in Kubernetes with Minikube.

**1. Setup Directory Structure**

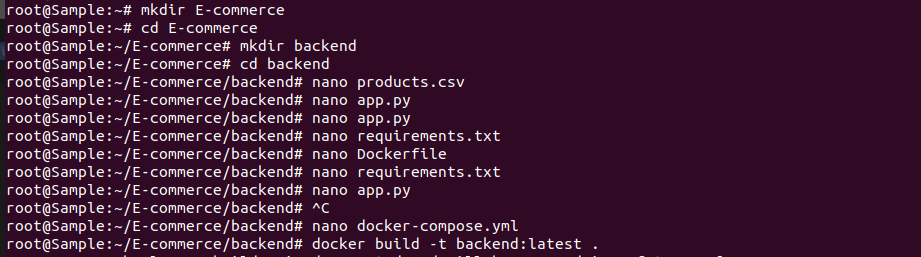
First, create a project directory to keep all files organized.

mkdir E-commerce && cd E-commerce

**Backend Setup**

Create a backend directory and navigate into it.

mkdir backend && cd backend



**Create products.csv**

This file will store product details in CSV format.

nano products.csv

Paste the following sample data:

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**

This script sets up a Flask server to read the CSV file and return product data as JSON.

nano app.py

Paste the following Python script:

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**

This file lists the dependencies required for the backend.

nano requirements.txt

Add dependencies:

flask

pandas

**Create Dockerfile**

This Dockerfile defines how to package the backend application into a container.

nano Dockerfile

Paste the following:

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"]

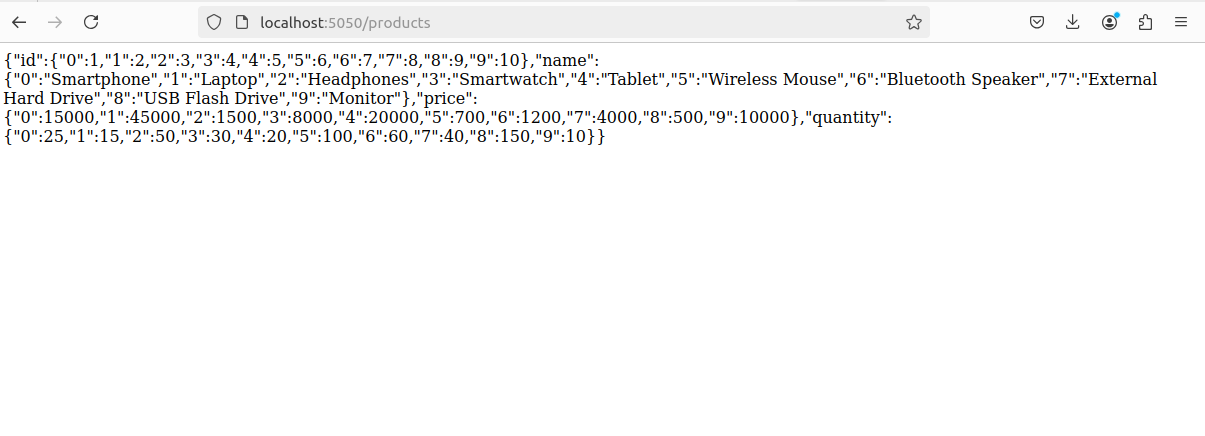
**Build & Run Backend Container**

Build and run the backend container.

docker build -t backend:latest .

docker run -itd -p 5050:5050 backend

docker logs $(docker ps -q --filter "ancestor=backend")



**Frontend Setup**

Create a frontend directory and navigate into it.

cd ..

mkdir frontend && cd frontend

**Create index.html**

This HTML file loads the product list from the backend.

nano index.html

Paste the following:

<!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**

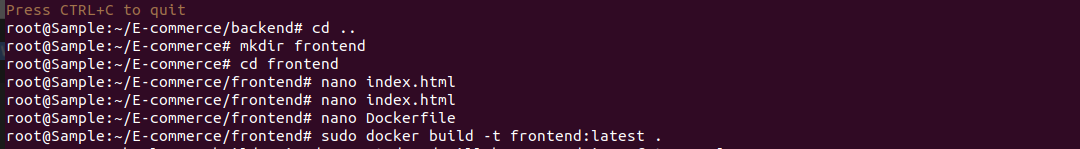
This Dockerfile packages the frontend as an Nginx container.

nano Dockerfile

Paste:

FROM nginx:alpine

COPY index.html /usr/share/nginx/html/index.html



**Build & Run Frontend Container**

docker build -t frontend:latest .

**2. Kubernetes Deployment**

Create a k8s directory for Kubernetes configuration files.

cd ..

mkdir k8s && cd k8s

**Backend Deployment (backend-deployment.yaml)**

Defines a backend pod in Kubernetes.

nano backend-deployment.yaml

Paste:

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)**

Defines a frontend pod in Kubernetes.

nano frontend-deployment.yaml

Paste:

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

**Connecting Frontend & Backend (service.yaml)**

Defines services for communication between frontend and backend.

nano service.yaml

Paste:

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)**

Stores backend configuration values.

nano configmap.yaml

Paste:

apiVersion: v1

kind: ConfigMap

metadata:

name: backend-config

data:

DATABASE\_FILE: "/backend/products.csv"

**3. Installing Kubernetes**

Instructions to install Minikube and kubectl.

**Step 1: Install Docker**

sudo apt update

sudo apt install -y docker.io

**Step 2: Verify Docker Installation**

docker --version

You should see output similar to: Docker version 20.10.12, build e91ed57

**Step 3: Enable and Start Docker**

sudo systemctl enable docker

sudo systemctl start docker

Check Docker status:

sudo systemctl status docker

Kubectl is the command-line tool used to interact with a Kubernetes cluster.

**Step 1: Download Kubectl**

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

**Step 2: Install Kubectl**

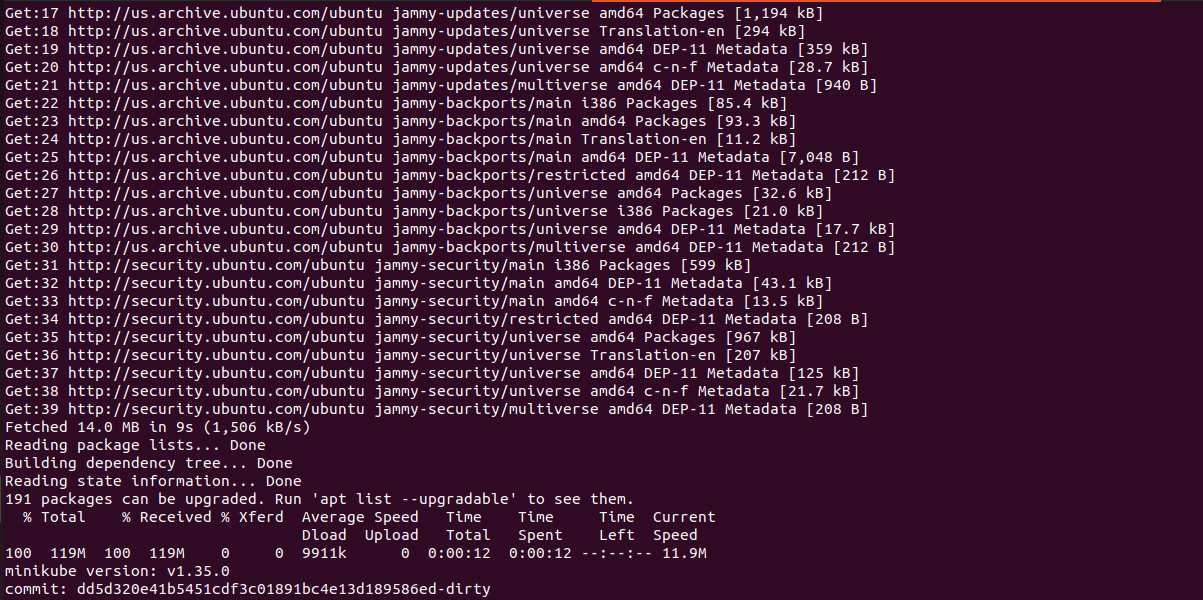
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

**Step 3: Verify Installation**

kubectl version --client

If the installation is successful, it will display the version information.

**4. Installing Minikube**

Minikube provides a local Kubernetes cluster, making it ideal for development and testing.

**Step 1: Download Minikube**

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

**Step 2: Install Minikube**

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

**Step 3: Verify Minikube Installation**

minikube version

This should return the installed Minikube version.

**5. Starting Minikube**

Once Minikube is installed, you can start it using the Docker driver.

minikube start --driver=docker

This command will:

* Download the necessary Kubernetes images.
* Start a single-node Kubernetes cluster.
* Configure kubectl to interact with the cluster.

Check the status of the Minikube cluster:

minikube status

Verify that Kubernetes is running:

kubectl get nodes

Expected output:

NAME STATUS ROLES AGE VERSION

minikube Ready control-plane,master 3m24s v1.32.0

**6. Enabling the Kubernetes Dashboard (Optional)**

Minikube includes a web-based Kubernetes dashboard. To enable it, run:

minikube dashboard

This will open a web browser with the Kubernetes dashboard.

**7. Managing Minikube**

**Stopping Minikube**

To stop the Minikube cluster without deleting it:

minikube stop

**Deleting Minikube Cluster**

To remove Minikube completely:

minikube delete

**Checking Running Services**

To list all Kubernetes services:

kubectl get services

**8. Troubleshooting Tips**

**1. If Minikube Fails to Start**

Try deleting and restarting Minikube:

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**

Deploying the application using Kubernetes and 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.

