Day 4: Kubernetes

Create a directory ‘e-commerce’ and its required folders and files

Create a **products.csv** file and **app.py**

```
student@mcacc1-6:~/docker-python-app/jenkins-docker-demo$ cd ..
student@mcacc1-6:~/docker-python-app$ cd ..
student@mcacc1-6:~$ mkdir e-commerce
student@mcacc1-6:~$ ls
docker-python-app  e-commerce
student@mcacc1-6:~$ cd e-commerce/
student@mcacc1-6:~/e-commerce$ mkdir frontend backend
student@mcacc1-6:~/e-commerce$ ls
backend  frontend
student@mcacc1-6:~/e-commerce$ cd backend/
student@mcacc1-6:~/e-commerce/backend$ nano products.csv
student@mcacc1-6:~/e-commerce/backend$ cat products.csv
id, name, price, qty
1, pen, 20, 100
2, book, 400, 50
3, laptop, 50000, 5,
4, shirt, 500, 50,
5, pants, 750, 50,

student@mcacc1-6:~/e-commerce/backend$ ls
products.csv
student@mcacc1-6:~/e-commerce/backend$ nano app.py
student@mcacc1-6:~/e-commerce/backend$ nano products.csv
student@mcacc1-6:~/e-commerce/backend$ ls
products.csv
student@mcacc1-6:~/e-commerce/backend$ nano app.py
student@mcacc1-6:~/e-commerce/backend$ nano products.csv
student@mcacc1-6:~/e-commerce/backend$ nano products.csv
student@mcacc1-6:~/e-commerce/backend$ cat products.csv
id,name,price,qty
1,pen,20,100
2,book,400,50
3,laptop,50000,5
4,shirt,500,50
5,pants,750,50
```

```
student@mcacc1-6:~/e-commerce/backend$ cat app.py
import pandas as pd
from flask import Flask

app = Flask(__name__)

@app.route("/products", method=['GET'])
def read_data():
        df = pd.read_csv("./products.csv")
        print(df.head())
        json_data = df.to_json()
        print(json_data)
        return json_data

if __name__ == "__main__":
        app.run(host="0.0.0.0", port=7000)
```

Install the **pandas** library:

```
student@mcacc1-6:~/e-commerce/backend$ sudo apt update
sudo apt install python3-pandas
[sudo] password for student:
```

Ensure that the CSV file is read and correctly parsed into **JSON format**.

To verify the **available port numbers**

```
student@mcacc1-6:~/e-commerce/backend$ sudo netstat -lp
[sudo] password for student:
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.53:domain       0.0.0.0:*               LISTEN      104/systemd-resolve
tcp        0      0 localhost:34075         0.0.0.0:*               LISTEN      238/containerd
tcp        0      0 0.0.0.0:http            0.0.0.0:*               LISTEN      208/nginx: master p
tcp        0      0 10.255.255.254:domain   0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.54:domain       0.0.0.0:*               LISTEN      104/systemd-resolve
tcp6       0      0 [::]:http               [::]:*                  LISTEN      208/nginx: master p
tcp6       0      0 [::]:http-alt           [::]:*                  LISTEN      151/java
udp        0      0 127.0.0.54:domain       0.0.0.0:*                           104/systemd-resolve
udp        0      0 127.0.0.53:domain       0.0.0.0:*                           104/systemd-resolve
udp        0      0 10.255.255.254:domain   0.0.0.0:*                           -
udp        0      0 localhost:323           0.0.0.0:*                           -
udp6       0      0 ip6-localhost:323       [::]:*                              -
Active UNIX domain sockets (only servers)
Proto RefCnt Flags       Type       State         I-Node   PID/Program name     Path
unix  2      [ ACC ]     STREAM     LISTENING     25614    2/init               /run/WSL/2_interop
unix  2      [ ACC ]     STREAM     LISTENING     19471    -                    /run/WSL/1_interop
unix  2      [ ACC ]     SEQPACKET  LISTENING     20867    -                    /mnt/wslg/weston-notify.sock
unix  2      [ ACC ]     STREAM     LISTENING     27649    -                    /var/run/dbus/system_bus_socket
unix  2      [ ACC ]     STREAM     LISTENING     24587    -                    /mnt/wslg/runtime-dir/wayland-0
unix  2      [ ACC ]     STREAM     LISTENING     24588    -                    /tmp/.X11-unix/X0
unix  2      [ ACC ]     STREAM     LISTENING     18603    104/systemd-resolve  /run/systemd/resolve/io.systemd.Resolve
unix  2      [ ACC ]     STREAM     LISTENING     18604    104/systemd-resolve  /run/systemd/resolve/io.systemd.Resolve.Monitor
unix  2      [ ACC ]     STREAM     LISTENING     19509    -                    /mnt/wslg/runtime-dir/pulse/native
unix  2      [ ACC ]     STREAM     LISTENING     23832    -                    /mnt/wslg/PulseAudioRDPSource
unix  2      [ ACC ]     STREAM     LISTENING     19682    1/init               /run/apport.socket
unix  2      [ ACC ]     STREAM     LISTENING     19684    1/init               /run/dbus/system_bus_socket
unix  2      [ ACC ]     STREAM     LISTENING     19685    1/init               /run/docker.sock
unix  2      [ ACC ]     STREAM     LISTENING     31887    864/systemd          /run/user/1000/systemd/private
unix  2      [ ACC ]     STREAM     LISTENING     19687    1/init               /run/snapd.socket
unix  2      [ ACC ]     STREAM     LISTENING     31894    864/systemd          /run/user/1000/bus
unix  2      [ ACC ]     STREAM     LISTENING     19688    1/init               /run/snapd-snap.socket
unix  2      [ ACC ]     STREAM     LISTENING     19690    1/init               /run/uuidd/request
unix  2      [ ACC ]     STREAM     LISTENING     31896    864/systemd          /run/user/1000/gnupg/S.dirmngr
unix  2      [ ACC ]     STREAM     LISTENING     31898    864/systemd          /run/user/1000/gnupg/S.gpg-agent.browser
```

Create **requirements.txt** file

The `requirements.txt` file is used in Python projects to list all the dependencies (packages) that the application needs to run.

```
student@mcacc1-6:~/e-commerce/backend$ nano requirements.txt
student@mcacc1-6:~/e-commerce/backend$ cat requirements.txt
flask
pandas
```

Create **docker-compose.yml** file

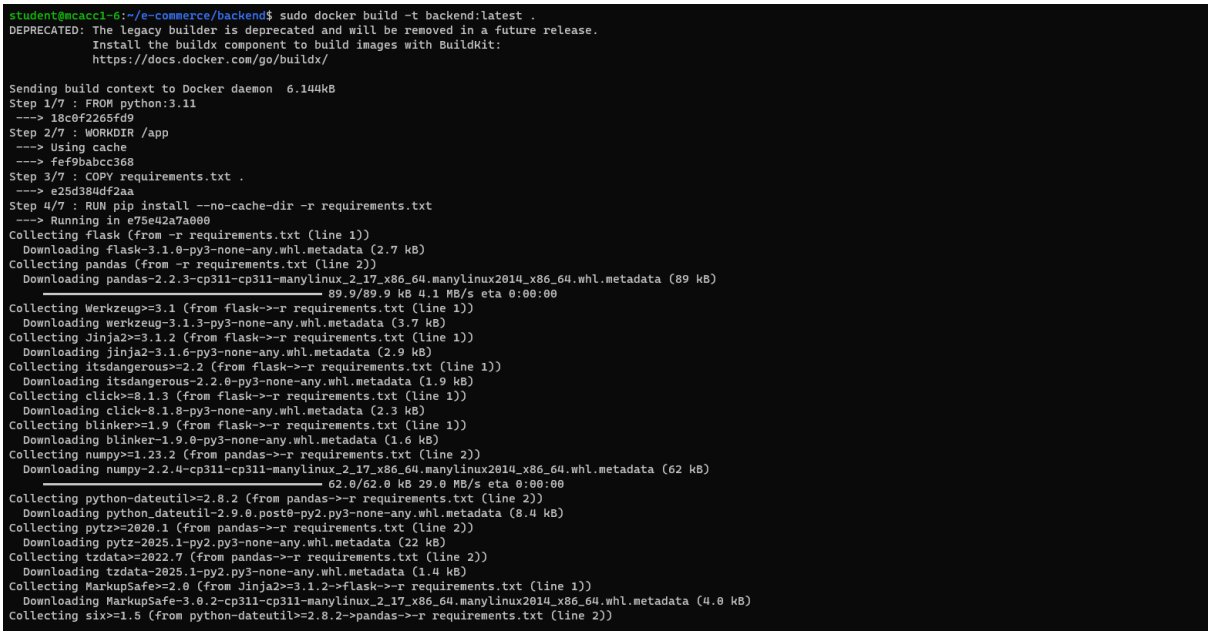
`docker-compose.yml` is a YAML configuration file used to define and run multi-container Docker applications.

```
student@mcacc1-6:~/e-commerce/backend$ nano docker-compose.yml
student@mcacc1-6:~/e-commerce/backend$ cat docker-compose.yml
version: '3.8'

services:
  web:
    build: .
    ports:
      - "7000:7000"
    volumes:
      - .:/app
    restart: always
```

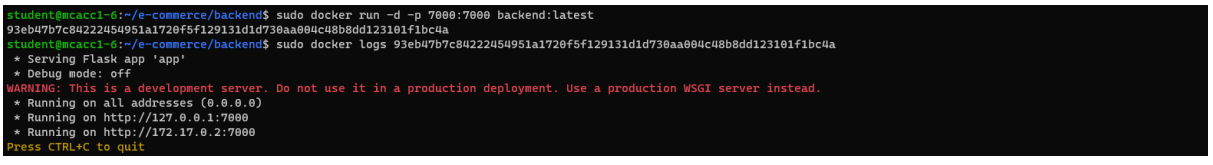
Build Docker image

Sudo docker build -t backend:latest



Run the docker:

sudo docker run -d -p 7000:7000 backend:latest
sudo docker logs <Generated number>



Run the application in the 7000/products

The JSON data is displayed at our port: 7000/products.

Create a container in frontend

Create **index.html file** and **Dockerfile**

```
student@mcacc1-6:~/e-commerce/backend$ cd ..
student@mcacc1-6:~/e-commerce$ ls
backend  frontend
student@mcacc1-6:~/e-commerce$ cd frontend/
student@mcacc1-6:~/e-commerce/frontend$ nano index.html
student@mcacc1-6:~/e-commerce/frontend$ nano Dockerfile
student@mcacc1-6:~/e-commerce/frontend$ cat Dockerfile
FROM nginx:alpine
COPY index.html /usr/share/nginx/html/index.html
student@mcacc1-6:~/e-commerce/frontend$
```

Build the image using the command:

sudo docker build -t frontend:latest.

```
student@mcacc1-6:~/e-commerce/frontend$ sudo docker build -t frontend:latest .
[sudo] password for student:
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  3.584kB
Step 1/2 : FROM nginx:alpine
alpine: Pulling from library/nginx
f18232174bc9: Pull complete
ccc35e35d420: Pull complete
43f2ec460bdf: Pull complete
984583bcf083: Pull complete
8d27c072a58f: Pull complete
ab3286a73463: Pull complete
6d79cc6084d4: Pull complete
0c7e4c092ab7: Pull complete
Digest: sha256:4ff102c5d78d254a6f0da062b3cf39eaf07f01eec0927fd21e219d0af8bc0591
Status: Downloaded newer image for nginx:alpine
 ---> 1ff4bb4faebc
Step 2/2 : COPY index.html /usr/share/nginx/html/index.html
 ---> ef6c27374482
Successfully built ef6c27374482
Successfully tagged frontend:latest
```

Kubernetes Deployment YAML Files

Create **backend-deployment.yaml file** and **frontend-deployment.yaml** in a folder k8s

These files define how our application should be deployed in the cluster.

```
student@mcacc1-6:~/e-commerce/k8s$ cat 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: 7000
student@mcacc1-6:~/e-commerce/k8s$ cat 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: 7500
student@mcacc1-6:~/e-commerce/k8s$
```

Create **service.yaml** file

It exposes our application within or outside the cluster.

```
student@mcacc1-6:~/e-commerce/k8s$ nano service.yaml
student@mcacc1-6:~/e-commerce/k8s$ cat service.yaml
apiVersion: v1
kind: Service
metadata:
  name: backend-service
spec:
  selector:
    app: backend
  ports:
    - protocol: TCP
      port: 7000
      targetPort: 7000
  type: ClusterIP

apiVersion: v1
kind: Service
metadata:
  name: frontend-service
spec:
  selector:
    app: frontend
  ports:
    - protocol: TCP
      port: 7500
      targetPort: 7500
  type: NodePort
```

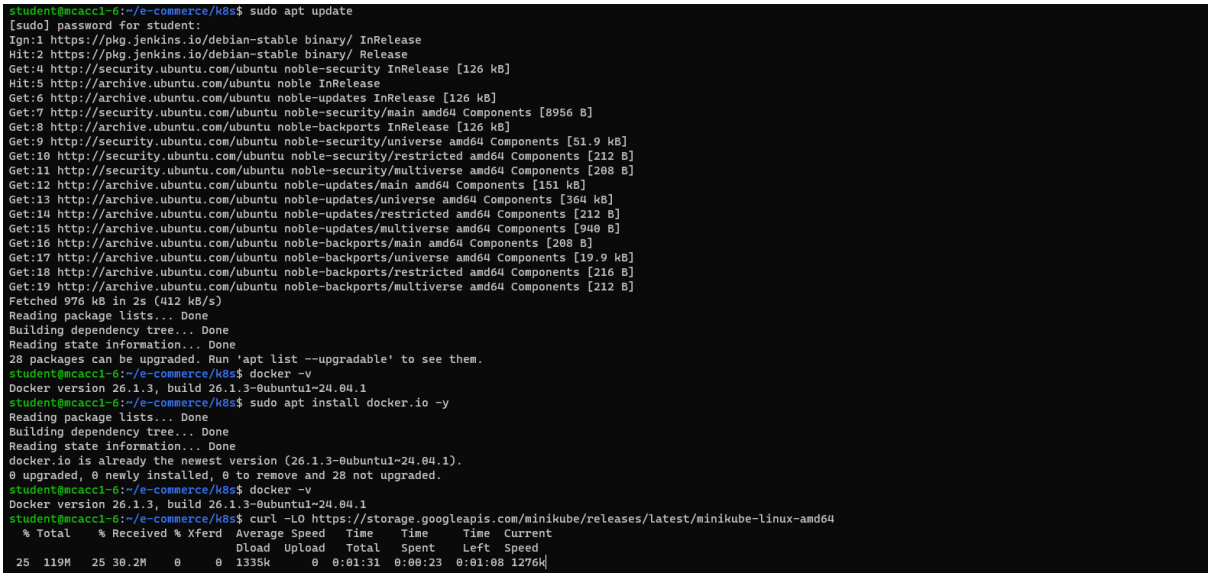
Create **configmap.yaml** file

Stores configuration data as key-value pairs.

```
student@mcacc1-6:~/e-commerce/k8s$ nano configmap.yaml
student@mcacc1-6:~/e-commerce/k8s$ cat configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: backend-config
data:
  DATABASE_FILE: "/backend/products.csv"
```

Install minikube

`Minikube` is a tool that allows you to run a Kubernetes cluster locally on our machine. It is designed for developers who want to test and experiment with Kubernetes without needing a full-scale cloud-based cluster.



Install kubectl

`kubectl` is the command-line tool used to interact with a Kubernetes cluster. It allows you to deploy applications, inspect and manage cluster resources, and troubleshoot issues.

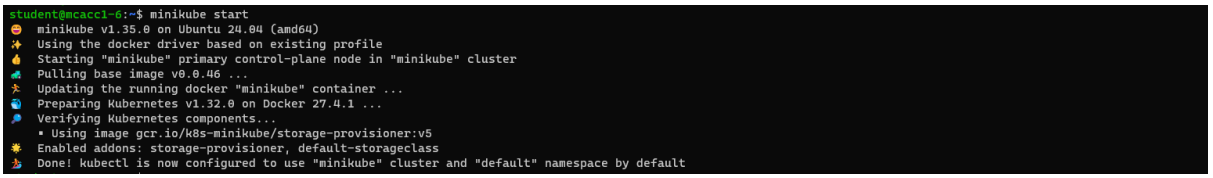
Grant permission for **kubectl**

chmod +x kubectl

Move to kubectl to root

Check the **minikube** and **kubectl** installed properly

Start minicube: **minikube start**



Verify minikube is running

Load the image to the minikube

Befor loading images

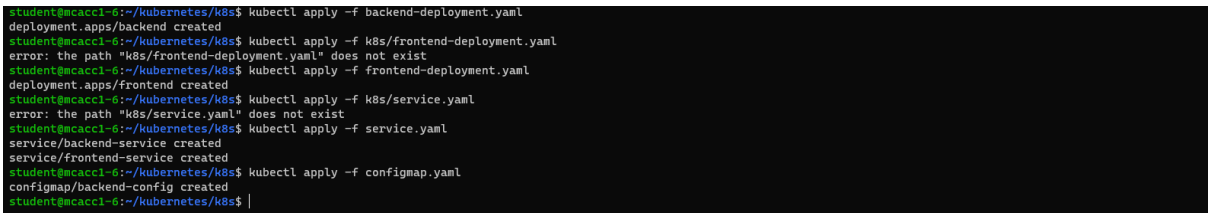
Perform this commend: eval $(minikube docker-env)

minikube image load frontend:latest

minikube image load backend:latest

Check the images are loaded

Commands are used to deploy your application components (backend and frontend), expose them through a service, and provide them with the necessary configuration via a ConfigMap.



These commands are used to list and inspect the running resources in your Kubernetes cluster:

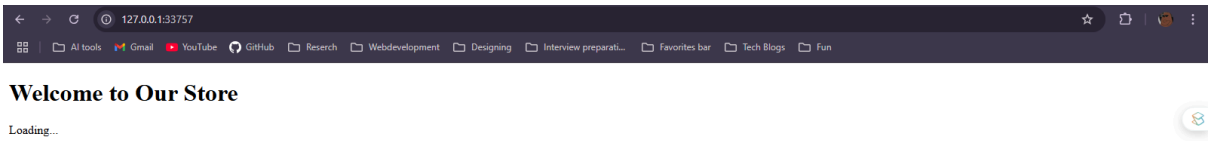
kubectl get pods

kubectl get svc

```
student@mcacc1-6:~/kubernetes/k8s$ kubectl get pods
NAME                       READY   STATUS    RESTARTS   AGE
backend-dfd8d5579-xz2xp    1/1     Running   0          3m46s
frontend-6cfd7c46-dsj9c    1/1     Running   0          3m14s
student@mcacc1-6:~/kubernetes/k8s$ kubectl get svc
NAME               TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
backend-service    ClusterIP   10.104.89.56     <none>        5000/TCP         3m12s
frontend-service   NodePort    10.105.136.172   <none>        3000:30520/TCP   3m12s
kubernetes         ClusterIP   10.96.0.1        <none>        443/TCP          3h53m
```

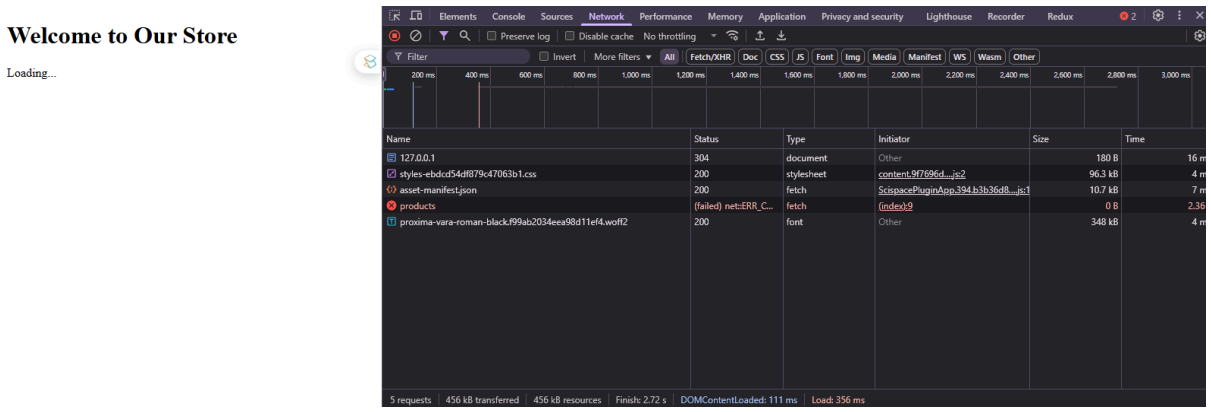
To test Frontend

```
student@mcacc1-6:~/kubernetes/k8s$ minikube service frontend-service --url
http://127.0.0.1:37341
❗  Because you are using a Docker driver on linux, the terminal needs to be open to run it.
```



To Test backend

```
student@mcacc1-6:~/kubernetes/k8s$ kubectl run test-pod --image=alpine --restart=Never -it -- sh
If you don't see a command prompt, try pressing enter.
/ # kubectl get pod test-pod
sh: kubectl: not found
/ # kubectl exec -it test-pod -- sh
sh: kubectl: not found
/ # 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":{"0":1,"1":2,"2":3,"3":4,"4":5},"name":{"0":"pen","1":"book","2":"laptop","3":"shirt","4":"pants"},"price":{"0":20,"1":400,"2":50000,"3":500,"4":750},"qty":{"0":100,"1"
:50,"2":5,"3":50,"4":50}}/ #
```



Note: We expect this kind of output because we are running this frontend on localhost.

— COMPLETED —