ECOMMERCE

STEP 1: CREATING AND ORGANIZING AN E-COMMERCE PROJECT DIRECTORY

A project directory named **"ecommerce"** is created.

1. Two subdirectories, **"frontend"** and **"backend"**, are added inside it.
2. The user navigates to the **"backend"** directory.
3. A CSV file named **"products.csv"** is created using the `nano` editor.
4. The file contains product details with columns: **Id, Name, Price, and Quantity**.
5. The contents of the CSV file are displayed using the `cat` command.

```
poojz@zZz:~$ mkdir ecommerce
poojz@zZz:~$ cd ecommerce
poojz@zZz:~/ecommerce$ mkdir frontend
poojz@zZz:~/ecommerce$ mkdir backend
poojz@zZz:~/ecommerce$ ls
backend  frontend
poojz@zZz:~/ecommerce$ cd backend
poojz@zZz:~/ecommerce/backend$ nano products.csv
poojz@zZz:~/ecommerce/backend$ cat products.csv
Id,Name,Price,Quantity
1,pooj,1000,10
2,sowmi,2000,20
3,dharshu,3000,30
3,fasee,4000,40
4,munima,5000,50
```

STEP 2: CONVERTING CSV DATA TO JSON IN PYTHON

1. The image shows a terminal session where a user creates and edits a Python script named "main.py" using the nano editor.
2. The script reads data from a CSV file ("products.csv") using the csv module.
3. It converts each row into a dictionary and appends it to a list.
4. The data is then printed in JSON format using the json module with proper indentation.

```
poojz@zZz:~/ecommerce/backend$ nano main.py
```

```
  GNU nano 7.2                                                  main.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")
        json_data=df.to_json()
        return json_data
if __name__ == "__main__":
        app.run(host="0.0.0.0",port=9000)
```

STEP 3: EXECUTING THE CSV TO JSON CONVERSION SCRIPT

1. The image shows the execution of the main.py script in a terminal.
2. The script reads data from "products.csv", converts it into a JSON format, and prints the structured output.
3. Each product entry is displayed as a JSON object with fields: Id, Name, Price, and Quantity.
4. The output confirms the successful transformation of CSV data into JSON.

```
poojz@zZz:~/ecommerce/backend$ python3 main.py
[
    {
        "Id": "1",
        "Name": "pooj",
        "Price": "1000",
        "Quantity": "10"
    },
    {
        "Id": "2",
        "Name": "sowmi",
        "Price": "2000",
        "Quantity": "20"
    },
    {
        "Id": "3",
        "Name": "dharshu",
        "Price": "3000",
        "Quantity": "30"
    },
    {
        "Id": "4",
        "Name": "fasee",
        "Price": "4000",
        "Quantity": "40"
    },
    {
        "Id": "5",
        "Name": "munima",
        "Price": "5000",
        "Quantity": "50"
    }
]
```

STEP 4: CHECKING ACTIVE LISTENING PORTS AND SERVICES WITH NETSTAT

1. The sudo netstat -lp command is executed in the terminal.
2. Displays active listening ports for both TCP and UDP connections.
3. Shows details like protocol, local/foreign addresses, state, and associated process (PID/Program Name).
4. Lists active UNIX domain sockets used by various system services.
5. Helps in network diagnostics, monitoring open ports, and identifying running services.

```
poojz@zZz:~/ecommerce/backend$ sudo netstat -lp
[sudo] password for poojz:
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 10.255.255.254:domain   0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:90              0.0.0.0:*               LISTEN      676/docker-proxy
tcp        0      0 0.0.0.0:81              0.0.0.0:*               LISTEN      245/nginx: master p
tcp        0      0 127.0.0.54:domain       0.0.0.0:*               LISTEN      119/systemd-resolve
tcp        0      0 127.0.0.53:domain       0.0.0.0:*               LISTEN      119/systemd-resolve
tcp        0      0 localhost:35253         0.0.0.0:*               LISTEN      257/containerd
tcp6       0      0 [::]:90                 [::]:*                  LISTEN      682/docker-proxy
tcp6       0      0 [::]:81                 [::]:*                  LISTEN      245/nginx: master p
tcp6       0      0 [::]:http-alt           [::]:*                  LISTEN      177/java
udp        0      0 127.0.0.54:domain       0.0.0.0:*                           119/systemd-resolve
udp        0      0 127.0.0.53:domain       0.0.0.0:*                           119/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     459      119/systemd-resolve  /run/systemd/resolve/io.systemd.Resolve
unix  2      [ ACC ]     STREAM     LISTENING     19579    2/init               /run/WSL/2_interop
unix  2      [ ACC ]     STREAM     LISTENING     36       -                    /run/WSL/1_interop
unix  2      [ ACC ]     STREAM     LISTENING     460      119/systemd-resolve  /run/systemd/resolve/io.systemd.Resolve.Monitor
unix  2      [ ACC ]     SEQPACKET  LISTENING     43       -                    /mnt/wslg/weston-notify.sock
unix  2      [ ACC ]     STREAM     LISTENING     17483    -                    /var/run/dbus/system_bus_socket
unix  2      [ ACC ]     STREAM     LISTENING     17774    1/init               /run/apport.socket
unix  2      [ ACC ]     STREAM     LISTENING     17776    1/init               /run/dbus/system_bus_socket
unix  2      [ ACC ]     STREAM     LISTENING     45       -                    /mnt/wslg/runtime-dir/wayland-0
unix  2      [ ACC ]     STREAM     LISTENING     17777    1/init               /run/docker.sock
unix  2      [ ACC ]     STREAM     LISTENING     17779    1/init               /run/snapd.socket
unix  2      [ ACC ]     STREAM     LISTENING     17781    1/init               /run/snapd-snap.socket
unix  2      [ ACC ]     STREAM     LISTENING     46       -                    /tmp/.X11-unix/X0
unix  2      [ ACC ]     STREAM     LISTENING     17783    1/init               /run/uuidd/request
```

STEP 5: DOCKERFILE FOR A PYTHON APPLICATION

1. Creates a Dockerfile for a Python application using Python 3.11.
2. Installs dependencies from requirements.txt and copies application files.
3. Exposes port 9000 and runs the app with python app.py.

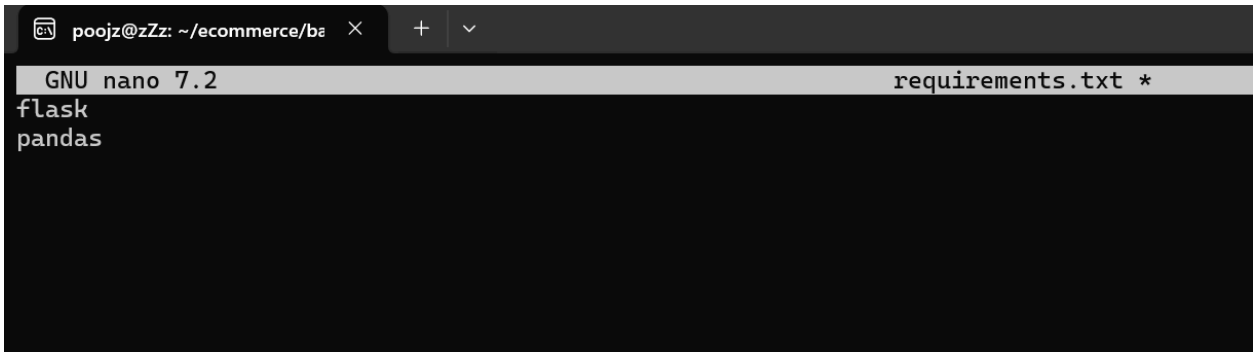
```
poojz@zZz:~/ecommerce/backend$ nano Dockerfile
```

```
  GNU nano 7.2                                                  Dockerfile *
FROM python:3.11
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY . .
EXPOSE 9000
CMD ["python", "app.py"]
```

STEP 6: DEFINING PROJECT DEPENDENCIES IN REQUIREMENTS.TXT

1. A **requirements.txt** file is created using the **nano** editor.
2. It lists **Flask** and **pandas** as dependencies for the Python project.
3. This file is used for installing required packages via **pip install -r requirements.txt**.

```
poojz@zZz:~/ecommerce/backend$ nano requirements.txt
```



STEP 7:CONFIGURING DOCKER SERVICES WITH DOCKER-COMPOSE.YML

1. A docker-compose.yml file is created to define a Docker service for a web application.
2. It maps port 9000 on the host to port 9000 in the container and mounts the ./app directory.
3. The service is configured to restart always, ensuring automatic recovery.

```
poojz@zZz:~/ecommerce/backend$ nano docker-compose.yml
```

STEP 8: BUILDING A DOCKER IMAGE FOR A PYTHON BACKEND

1. The user adds themselves to the Docker group to execute Docker commands without sudo.
2. A Docker image is built from a Dockerfile, tagged as backend:latest.
3. The build process uses cached layers and successfully creates a Python-based backend image.

```
poojz@zZz:~/ecommerce/backend$ sudo usermod -aG docker poojz
[sudo] password for poojz:
poojz@zZz:~/ecommerce/backend$ docker build -t backend:latest .
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   7.68kB
Step 1/7 : FROM python:3.11
 ---> 18c0f2265fd9
Step 2/7 : WORKDIR /app
 ---> Using cache
 ---> 0f457dcc3b7a
Step 3/7 : COPY requirements.txt .
 ---> Using cache
 ---> 28f1ef86e346
Step 4/7 : RUN pip install --no-cache-dir -r requirements.txt
 ---> Using cache
 ---> 43ee4d624edd
Step 5/7 : COPY . .
 ---> Using cache
 ---> 8859122c66a1
Step 6/7 : EXPOSE 5002
 ---> Using cache
 ---> 3a4a22f3b734
Step 7/7 : CMD ["python", "app.py"]
 ---> Using cache
 ---> d12618d007d9
Successfully built d12618d007d9
Successfully tagged backend:latest
```

STEP 9: RUNNING A FLASK APPLICATION AND HANDLING REQUESTS

1. The Flask application is running on port 9000, accessible via localhost and the network IP.
2. An HTTP GET request to /products is successfully processed, returning a 200 OK response.

```
(myenv) poojz@zZz:~/ecommerce/backend$ python3 main.py
 * Serving Flask app 'main'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:9000
 * Running on http://192.168.247.184:9000
Press CTRL+C to quit
127.0.0.1 - - [20/Mar/2025 07:36:34] "GET /products HTTP/1.1" 200 -
```

STEP 10:RETRIEVING PRODUCT DATA VIA CURL REQUEST

1. A curl request is made to http://127.0.0.1:9000/products, retrieving JSON data.
2. The response contains product details, including Id, Name, and Price.

```
poojz@zZz:~$ curl http://127.0.0.1:9000/products
{"Id":{"0":1,"1":2,"2":3,"3":4,"4":5},"Name":{"0":"pooj","1":"sowmi","2":"dharshu","3":"fasee","4":"munima"},"Price":{"0":10
```

STEP 11: EDITING AN HTML FILE TO FETCH AND DISPLAY PRODUCT DATA

1. The HTML file (index.html) being edited in the nano text editor, containing a script to fetch product data from http://localhost:9000/products.
2. The script dynamically updates the webpage by displaying the product names and prices inside the with id="product-list".

```
poojz@zZz:~/ecommerce/frontend$ nano index.html
```

```
  GNU nano 7.2                                                  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:9000/products");
            const products = await response.json();
            let output = "<h2>Product List</h2><ul>";
            products.forEach(product => {
                output += `<li>${product.name} - $${product.price}</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>
```

STEP 12:DOCKERFILE FOR NGINX-BASED FRONTEND

1. Base Image: Uses nginx:alpine as the lightweight web server.
2. File Deployment: Copies index.html to Nginx's default serving directory for static hosting.

```
poojz@zZz:~/ecommerce/frontend$ nano dockerfile
```

```
  GNU nano 7.2                                                  dockerfile
FROM nginx:alpine
COPY index.html /usr/share/nginx/html/index.html
```

STEP 13:

The process of building a Docker image named frontend:latest using an nginx:alpine base and copying index.html into the Nginx server directory**.**

```
poojz@zZz:~/ecommerce/frontend$ sudo docker build -t frontend:latest .
[sudo] password for poojz:
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
 ---> d1144f5c4d7f
Successfully built d1144f5c4d7f
Successfully tagged frontend:latest
```

STEP 14:

The creation of a k8s directory and navigation into it using the mkdir and cd commands in a terminal.

```
poojz@zZz:~/ecommerce$ mkdir k8s
poojz@zZz:~/ecommerce$ cd k8s
```

STEP 15:

The Kubernetes deployment YAML file being edited using the nano editor, defining a backend deployment with one replica and exposing port 9000.

```
poojz@zZz:~/ecommerce/k8s$ nano backend-deployment.yaml
```

```
 GNU nano 7.2                                  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: 9000
```

STEP 16:

The Kubernetes deployment YAML file being edited using nano, defining a frontend deployment with one replica and exposing port 7000.

```
poojz@zZz:~/ecommerce/k8s$ nano frontend-deployment.yaml
```

```
 GNU nano 7.2                                  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: 7000
```

STEP 17:

The Kubernetes service YAML file defining a ClusterIP service for the backend on port 9000 and a NodePort service for the frontend on port 7000.

```
poojz@zZz:~/ecommerce/k8s$ nano service.yaml
```

```
 GNU nano 7.2                                  service.yaml *
apiVersion: v1
kind: Service
metadata:
  name: backend-service
spec:
  selector:
    app: backend
  ports:
    - protocol: TCP
      port: 9000
      targetPort: 9000
  type: ClusterIP

apiVersion: v1
kind: Service
metadata:
  name: frontend-service
spec:
  selector:
    app: frontend
  ports:
    - protocol: TCP
      port: 7000
      targetPort: 7000
  type: NodePort
```

STEP 18:

The Kubernetes ConfigMap YAML file defining a configuration for the backend, specifying a database file path as /backend/products.csv.

```
poojz@zZz:~/ecommerce/k8s$ nano configmap.yaml
```

```
  GNU nano 7.2                                          configmap.yaml *
apiVersion: v1
kind: ConfigMap
metadata:
  name: backend-config
data:
  DATABASE_FILE: "/backend/products.csv"
```

STEP 19:

The terminal running sudo apt update, fetching package lists and indicating that 122 packages can be upgraded.

```
poojz@zZz:~/ecommerce/k8s$ sudo apt update
[sudo] password for poojz:
Ign:1 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:2 https://pkg.jenkins.io/debian-stable binary/ Release
Hit:4 http://archive.ubuntu.com/ubuntu noble InRelease
Get:5 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:6 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:7 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [8956 B]
Get:9 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [151 kB]
Get:10 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [51.9 kB]
Get:11 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [212 B]
Get:12 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [208 B]
Get:13 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [364 kB]
Get:14 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Get:15 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]
Get:16 http://archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [208 B]
Get:17 http://archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [19.9 kB]
Get:18 http://archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:19 http://archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Fetched 976 kB in 7s (130 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
122 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

STEP 20:

The terminal output confirming that `docker.io` is already installed with the latest version, and no upgrades or new installations were performed.

```
poojz@zZz:~/ecommerce/k8s$ sudo apt install docker.io -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
docker.io is already the newest version (26.1.3-0ubuntu1~24.04.1).
0 upgraded, 0 newly installed, 0 to remove and 122 not upgraded.
```

STEP 21:

The terminal command using curl -LO to download the latest Minikube binary for Linux (AMD64) from Google APIs, with a completed download of 119MB.

```
poojz@zZz:~/ecommerce/k8s$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  119M  100  119M    0     0   446k      0  0:04:34  0:04:34 --:--:--  428k
```

STEP 22:

The terminal command using curl -LO to download the latest stable release of kubectl for Linux (AMD64) from the Kubernetes official repository.

```
poojz@zZz:~/ecommerce/k8s$ curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kub
ectl"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0    295      0 --:--:-- --:--:-- --:--:--   296
100 54.6M  100 54.6M    0     0   451k      0  0:02:03  0:02:03 --:--:--  425k
```

STEP 23:

STOP ALL KUBERNETES PODS AND DEPLOYMENTS

```
poojz@zZz:~$ kubectl delete all --all --force --grace-period=0
E0321 04:05:12.732178    6916 memcache.go:265] "Unhandled Error" err=<
        couldn't get current server API group list: <html><head><meta http-equiv='refresh' content='1;url=/login?from=%2Fapi%3Ftimeou
t%3D32s'/><script id='redirect' data-redirect-url='/login?from=%2Fapi%3Ftimeout%3D32s' src='/static/85cd95ae/scripts/redirect.js'></s
cript></head><body style='background-color:white; color:white;'>
        Authentication required
        <!--
        -->

        </body></html>
 >
E0321 04:05:12.769544    6916 memcache.go:265] "Unhandled Error" err=<
        couldn't get current server API group list: <html><head><meta http-equiv='refresh' content='1;url=/login?from=%2Fapi%3Ftimeou
t%3D32s'/><script id='redirect' data-redirect-url='/login?from=%2Fapi%3Ftimeout%3D32s' src='/static/85cd95ae/scripts/redirect.js'></s
cript></head><body style='background-color:white; color:white;'>
        Authentication required
        <!--
        -->

        </body></html>
 >
E0321 04:05:12.778208    6916 memcache.go:265] "Unhandled Error" err=<
        couldn't get current server API group list: <html><head><meta http-equiv='refresh' content='1;url=/login?from=%2Fapi%3Ftimeou
t%3D32s'/><script id='redirect' data-redirect-url='/login?from=%2Fapi%3Ftimeout%3D32s' src='/static/85cd95ae/scripts/redirect.js'></s
cript></head><body style='background-color:white; color:white;'>
        Authentication required
        <!--
        -->

        </body></html>
```

```
poojz@zZz:~$ kubectl delete namespace kube-system --force --grace-period=0
E0321 04:09:21.180614    6968 memcache.go:265] "Unhandled Error" err=<
        couldn't get current server API group list: <html><head><meta http-equiv='refresh' content='1;url=/login?from=%2Fapi%3Ftimeou
t%3D32s'/><script id='redirect' data-redirect-url='/login?from=%2Fapi%3Ftimeout%3D32s' src='/static/85cd95ae/scripts/redirect.js'></s
cript></head><body style='background-color:white; color:white;'>
        Authentication required
        <!--
        -->

        </body></html>
 >
E0321 04:09:21.191214    6968 memcache.go:265] "Unhandled Error" err=<
        couldn't get current server API group list: <html><head><meta http-equiv='refresh' content='1;url=/login?from=%2Fapi%3Ftimeou
t%3D32s'/><script id='redirect' data-redirect-url='/login?from=%2Fapi%3Ftimeout%3D32s' src='/static/85cd95ae/scripts/redirect.js'></s
cript></head><body style='background-color:white; color:white;'>
        Authentication required
        <!--
        -->

        </body></html>
 >
E0321 04:09:21.197864    6968 memcache.go:265] "Unhandled Error" err=<
        couldn't get current server API group list: <html><head><meta http-equiv='refresh' content='1;url=/login?from=%2Fapi%3Ftimeou
t%3D32s'/><script id='redirect' data-redirect-url='/login?from=%2Fapi%3Ftimeout%3D32s' src='/static/85cd95ae/scripts/redirect.js'></s
cript></head><body style='background-color:white; color:white;'>
        Authentication required
        <!--
        -->

        </body></html>
```

STOP AND DELETE MINIKUBE

STOP AND REMOVE DOCKER CONTAINERS / FORCE-STOP ALL RUNNING CONTAINERS

```
poojz@zZz:~$ minikube stop
🤷  Profile "minikube" not found. Run "minikube profile list" to view all profiles.
👉  To start a cluster, run: "minikube start"
poojz@zZz:~$ minikube delete --all --purge
🔥  Successfully deleted all profiles
💀  Successfully purged minikube directory located at - [/home/poojz/.minikube]
poojz@zZz:~$ docker kill $(docker ps -q)
41c8b5b9eef1
poojz@zZz:~$ docker rm -f $(docker ps -aq)
2143f001931e
41c8b5b9eef1
78496f06ce11
510d2b8cd4ad
74489c586d0e
```

STOP DOCKER AND RESTART
KILL ALL KUBERNETES, MINIKUBE, AND DOCKER PROCESSES

```
poojz@zZz:~$ systemctl stop docker
Failed to stop docker.service: Interactive authentication required.
See system logs and 'systemctl status docker.service' for details.
poojz@zZz:~$ systemctl start docker
Failed to start docker.service: Interactive authentication required.
See system logs and 'systemctl status docker.service' for details.
poojz@zZz:~$ pkill -f docker
pkill: killing pid 398 failed: Operation not permitted
poojz@zZz:~$ pkill -f minikube
poojz@zZz:~$ pkill -f kubectl
pkill: killing pid 6901 failed: Operation not permitted
poojz@zZz:~$ pkill -f containerd
pkill: killing pid 257 failed: Operation not permitted
pkill: killing pid 398 failed: Operation not permitted
```

CLEANUP DOCKER SYSTEM

```
poojz@zZz:~$ docker system prune -a --volumes -f
Deleted Networks:
devops_default
dev-ops-training_default

Deleted Images:
deleted: sha256:d12618d007d9deaa7b7be74ad982385955b30d3ddddd1e11cc0fac4bcdbed987
deleted: sha256:3a4a22f3b7343ee73c1b87277ec95f9032a50738c9d2f4e83f02d4c51ddd4c8f
deleted: sha256:8859122c66a1753801a0e0898cd4639be9d58998991908306c7332311c1c4253
deleted: sha256:200420168109e2e20b018b756627b1541c96466ce6cdb1bbc116c768761a4651
deleted: sha256:6f183c1a985558bf70cee837d3a3c28414ae6129f75743fcb43bc07345d94289
deleted: sha256:f2ce17d73d476d71f0f4fef3ab9e3b68438616ae20d092f61929ea42630d96bd
deleted: sha256:e71e7ed730a74ef0d631a2e4e591021ed4087174d5ffa05b296cfe9f80f8f6b5
untagged: test:latest
deleted: sha256:b597cf24bcdc7fd65d50f6318a6cefdd5d4f2f2a1c35111b6ad0cf1a5c09e234
deleted: sha256:54f72935352a9e287e1f36ad2c5e14583cbd034a61f649a375fdf231db7c7a68
deleted: sha256:50cfd855ef3d5e4c9f7f5808298eb5d5ec5c2b6ae01ebea01a7387f6b15c3f68
deleted: sha256:86dccdddffe0a27de527a7f6ea9e0f7be5e466afe09cd11c1cd1e2bb4aecd22d
deleted: sha256:edd56606df8af91df8727982d8f56808b33ed016ae31e632ef0f51d8b4a29606
deleted: sha256:a457ed97a36880db68b034cef57f6e4115d2a7f5bb5f6a941b68cf20f787d8e0
deleted: sha256:2131b8a4cec8a0815f97603d61dbbbeeeea6544e8dffcdffa3c2b8205871783b
deleted: sha256:708180e34278d4c38951a3df704d3fa991e6405861b0c1c7bb945d06d8eb16cd
untagged: backend:latest
deleted: sha256:22f1ad41f09d6282a4f9092b069d0f7e1a70a2d702f7f576de46c255d3895b80
deleted: sha256:7eeb3611f9810d023facb5e1e86bef95ba025bcd118452c61341815dfc14a223
deleted: sha256:9e4a613a277aff191ef8beddf6391ef286b9ac0d5a3dcd90d6622d047c09270e
deleted: sha256:a1515bf2aa4fee3757139eea3cc2ff5970f6e5fcc73a1b4495af0a813a94dd9b
untagged: nginx:latest
```

COMMANDS TO STOP ALL IP’S UTILISING 8080

```
poojz@zZz:~$ sudo netstat -tulnp | grep ":8080"
[sudo] password for poojz:
tcp6       0      0 :::8080                 :::*                    LISTEN      177/java
poojz@zZz:~$ sudo kill -9 <PID>
-bash: syntax error near unexpected token `newline'
```

STEP 24:

1. UPDATE PACKAGE INDEX
2. INSTALL DOCKER
3. START AND ENABLE DOCKER
4. VERIFY INSTALLATION
5. DOWNLOAD MINIKUBE BINARY
6. INSTALL MINIKUBE
7. VERIFY INSTALLATION
8. DOWNLOAD KUBECTL
9. MAKE IT EXECUTABLE
10. MOVE TO A SYSTEM PATH
11. VERIFY INSTALLATION
12. START MINIKUBE
13. VERIFY MINIKUBE IS RUNNING

```
poojz@zZz:~$ sudo apt update
Ign:1 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:2 https://pkg.jenkins.io/debian-stable binary/ Release
Hit:3 http://archive.ubuntu.com/ubuntu noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:6 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:7 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
122 packages can be upgraded. Run 'apt list --upgradable' to see them.
poojz@zZz:~$ sudo apt install docker.io -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
docker.io is already the newest version (26.1.3-0ubuntu1~24.04.1).
0 upgraded, 0 newly installed, 0 to remove and 122 not upgraded.
poojz@zZz:~$ sudo systemctl start docker
Warning: The unit file, source configuration file or drop-ins of docker.service changed on disk. Run 'systemctl daemon-reload' to reload units.
poojz@zZz:~$ sudo systemctl enable docker
poojz@zZz:~$ docker --version
Docker version 26.1.3, build 26.1.3-0ubuntu1~24.04.1
poojz@zZz:~$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  119M  100  119M    0     0   618k      0  0:03:17  0:03:17 --:--:--  861k
poojz@zZz:~$ sudo install minikube-linux-amd64 /usr/local/bin/minikube
poojz@zZz:~$ minikube version
minikube version: v1.35.0
commit: dd5d320e41b5451cdf3c01891bc4e13d189586ed-dirty
poojz@zZz:~$ curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0    255      0 --:--:-- --:--:-- --:--:--   255
100 54.6M  100 54.6M    0     0   832k      0  0:01:07  0:01:07 --:--:--  676k
poojz@zZz:~$ chmod +x kubectl
poojz@zZz:~$ sudo mv kubectl /usr/local/bin/
poojz@zZz:~$ kubectl version --client
Client Version: v1.32.3
Kustomize Version: v5.5.0
```

```
poojz@zZz:~$ minikube start
😄  minikube v1.35.0 on Ubuntu 24.04 (amd64)
E0321 04:53:05.788552    8340 start.go:812] api.Load failed for minikube: filestore "minikube": Docker machine "minikube" does not exist. Use "docker-machin
e ls" to list machines. Use "docker-machine create" to add a new one.
✨  Using the docker driver based on existing profile
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.46 ...
💾  Downloading Kubernetes v1.32.0 preload ...
    > preloaded-images-k8s-v18-v1...:  333.57 MiB / 333.57 MiB  100.00% 407.20
    > index.docker.io/kicbase/sta...:  500.31 MiB / 500.31 MiB  100.00% 325.45
❗  minikube was unable to download gcr.io/k8s-minikube/kicbase:v0.0.46, but successfully downloaded docker.io/kicbase/stable:v0.0.46@sha256:fd2d445ddcc33eb
c5c6b68a17e6219ea207ce63c005095ea1525296da2d1a279 as a fallback image
🔥  Creating docker container (CPUs=2, Memory=2200MB) ...
🐳  Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
    ▪ Generating certificates and keys ...
    ▪ Booting up control plane ...
    ▪ Configuring RBAC rules ...
🔗  Configuring bridge CNI (Container Networking Interface) ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Enabled addons: storage-provisioner, default-storageclass
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
poojz@zZz:~$ kubectl get nodes
NAME       STATUS   ROLES           AGE    VERSION
minikube   Ready    control-plane   111s   v1.32.0
```

STEP 25:

1. The user clones a Git repository from GitHub (kubernetes project).
2. Sets up the Minikube Docker environment using eval $(minikube docker-env).
3. Navigates to the "backend" directory within the cloned kubernetes project.

```
poojz@zZz:~$ git clone https://github.com/PadmavathyNarayanan/kubernetes.git
Cloning into 'kubernetes'...
remote: Enumerating objects: 34, done.
remote: Counting objects: 100% (34/34), done.
remote: Compressing objects: 100% (30/30), done.
remote: Total 34 (delta 7), reused 15 (delta 1), pack-reused 0 (from 0)
Receiving objects: 100% (34/34), 8.67 KiB | 493.00 KiB/s, done.
Resolving deltas: 100% (7/7), done.
```

```
poojz@zZz:~/kubernetes/backend$ cd
poojz@zZz:~$ eval $(minikube docker-env)
poojz@zZz:~$ cd kubernetes
poojz@zZz:~/kubernetes$ cd backend
```

STEP 26:

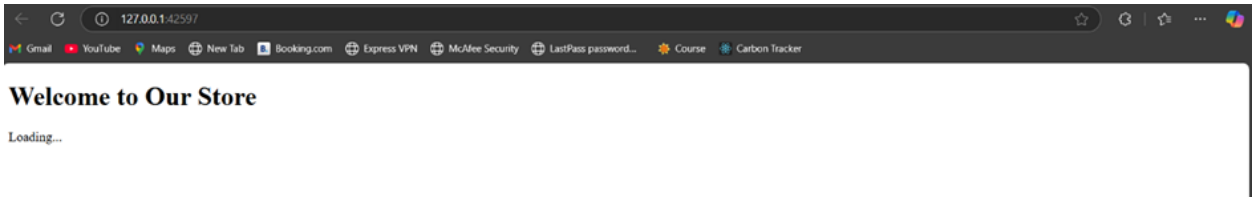
1. Navigating to Directories: The user navigates to the kubernetes directory and attempts to access frontend (initially missing).
2. Cloning Repository: The user re-clones the kubernetes repository from GitHub.
3. Building Docker Image: Builds a Docker image for the frontend using: docker build -t frontend:latest . The build process uses nginx:alpine as the base image.
4. Verifying Image: Lists Docker images and confirms frontend:latest is created successfully (47.9 MB).
5. Loading Image to Minikube: The user attempts to load the backend:latest image into Minikube using: minikube image load backend:latest

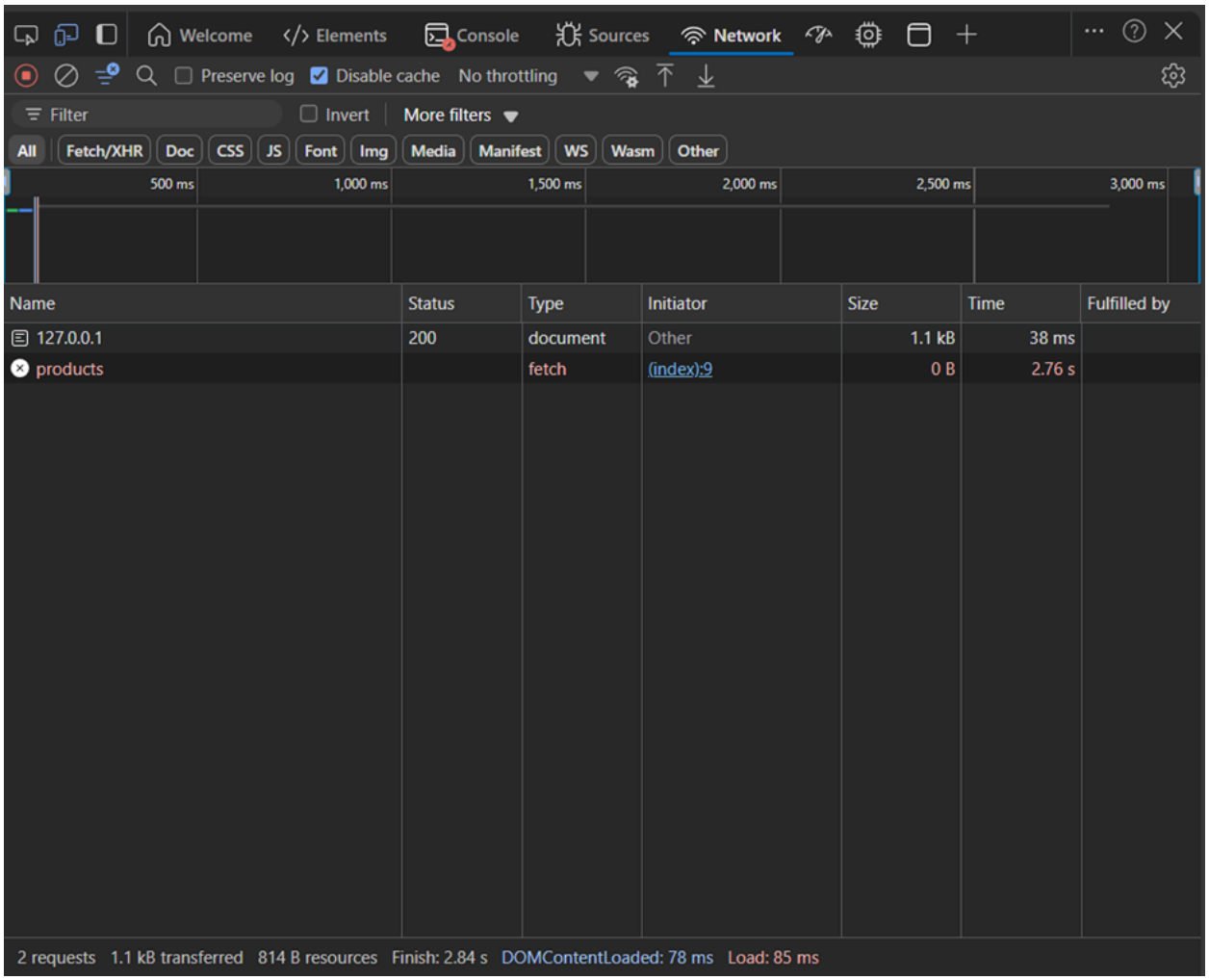
```
poojz@zZz:~$ cd kubernetes
poojz@zZz:~/kubernetes$ cd frontend
-bash: cd: frontend: No such file or directory
poojz@zZz:~/kubernetes$ ls
README.md              commands-to-stop-instances
commands-to-execute    pre-requisites-command
poojz@zZz:~/kubernetes$ cd
poojz@zZz:~$ rm -rf kubernetes
poojz@zZz:~$ git clone https://github.com/PadmavathyNarayanan/kubernetes.git
Cloning into 'kubernetes'...
remote: Enumerating objects: 34, done.
remote: Counting objects: 100% (34/34), done.
remote: Compressing objects: 100% (30/30), done.
remote: Total 34 (delta 7), reused 15 (delta 1), pack-reused 0 (from 0)
Receiving objects: 100% (34/34), 8.67 KiB | 555.00 KiB/s, done.
Resolving deltas: 100% (7/7), done.
poojz@zZz:~$ cd kubernetes
poojz@zZz:~/kubernetes$ cd frontend
poojz@zZz:~/kubernetes/frontend$ docker build -t frontend:latest .
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
 ---> 1ff4bb0faebc
Step 2/2 : COPY index.html /usr/share/nginx/html/index.html
 ---> 6de323fac859
Successfully built 6de323fac859
Successfully tagged frontend:latest
poojz@zZz:~/kubernetes/frontend$ docker images | grep frontend
frontend     latest    6de323fac859   39 seconds ago   47.9MB
poojz@zZz:~/kubernetes/frontend$ minikube image load fronten
d:latest
```

```
poojz@zZz:~/kubernetes/backend$ minikube image load backend:latest
```

STEP 27:

OUTPUT:





Note : Since, we are expected this kind of output, because we are running this frontend in localhost.