

In-class Assignment 2

Instructor: Qasim Ali

Develop and Deploy a Machine Learning Application using Docker

Group C

Govind Ram Gupta Belde

Assignment Steps

Step 1: Set Up the VM

1. Update the System

The screenshot shows a terminal window titled "SSH-in-browser" with the following output:

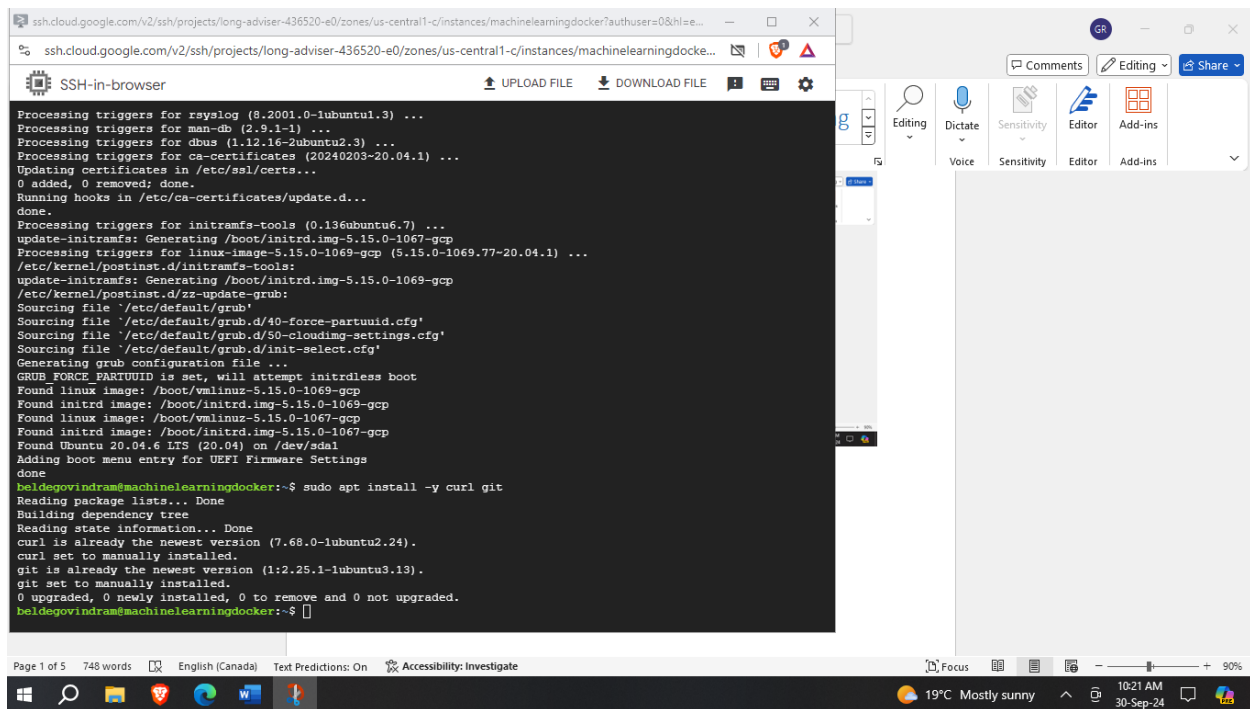
```
Setting up linux-image-5.15.0-1069-gcp (5.15.0-1069.77~20.04.1) ...
I: /boot/vmlinuz is now a symlink to vmlinuz-5.15.0-1069-gcp
I: /boot/initrd.img is now a symlink to initrd.img-5.15.0-1069-gcp
Setting up linux-image-gcp (5.15.0-1069.77~20.04.1) ...
Setting up linux-gcp (5.15.0-1069.77~20.04.1) ...
Processing triggers for mime-support (3.64ubuntu1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.16) ...
Processing triggers for rsyslog (8.2001.0-1ubuntu1.3) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for dbus (1.12.16-2ubuntu2.3) ...
Processing triggers for ca-certificates (20240203-20.04.1) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
Processing triggers for initramfs-tools (0.136ubuntu6.7) ...
update-initramfs: Generating /boot/initrd.img-5.15.0-1067-gcp
Processing triggers for linux-image-5.15.0-1069-gcp (5.15.0-1069.77~20.04.1) ...
/etc/kernel/postinst.d/initramfs-tools:
update-initramfs: Generating /boot/initrd.img-5.15.0-1069-gcp
/etc/kernel/postinst.d/zz-update-grub:
Sourcing file '/etc/default/grub'
Sourcing file '/etc/default/grub.d/40-force-partuuid.cfg'
Sourcing file '/etc/default/grub.d/50-cloudimg-settings.cfg'
Sourcing file '/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
GRUB_FORCE_PARTUUID is set, will attempt initrdless boot
Found linux image: /boot/vmlinuz-5.15.0-1069-gcp
Found initrd image: /boot/initrd.img-5.15.0-1069-gcp
Found linux image: /boot/vmlinuz-5.15.0-1067-gcp
Found initrd image: /boot/initrd.img-5.15.0-1067-gcp
Found Ubuntu 20.04.6 LTS (20.04) on /dev/sdal
Adding boot menu entry for UEFI Firmware Settings
done
beldegovindram@machinelearningdocker:~$
```

The terminal window is overlaid on a Google Docs editor. The document title is "SSH-in-browser". The document content is empty. The Google Docs interface shows the "Editing" tab selected, with options for "Comments", "Editing", and "Share". The bottom status bar shows "Page 2 of 6", "835 words", "English (Canada)", "Text Predictions: On", "Accessibility: Good to go", "Focus", "19°C Mostly sunny", "10:20 AM", "30-Sep-24", and "90%".

2. Install Necessary Packages

- Install curl and git:

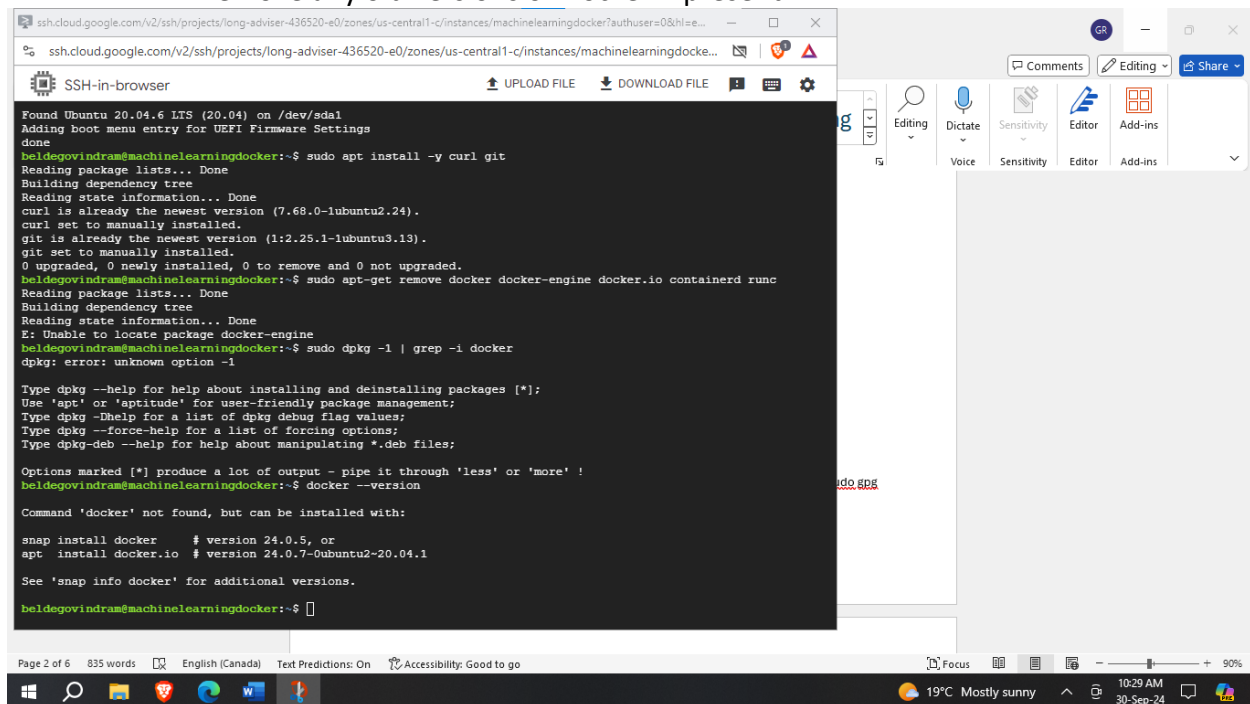
```
sudo apt install -y curl git
```



Step 2: Install Docker

1. Remove Old Versions

- Remove any old versions of Docker if present:



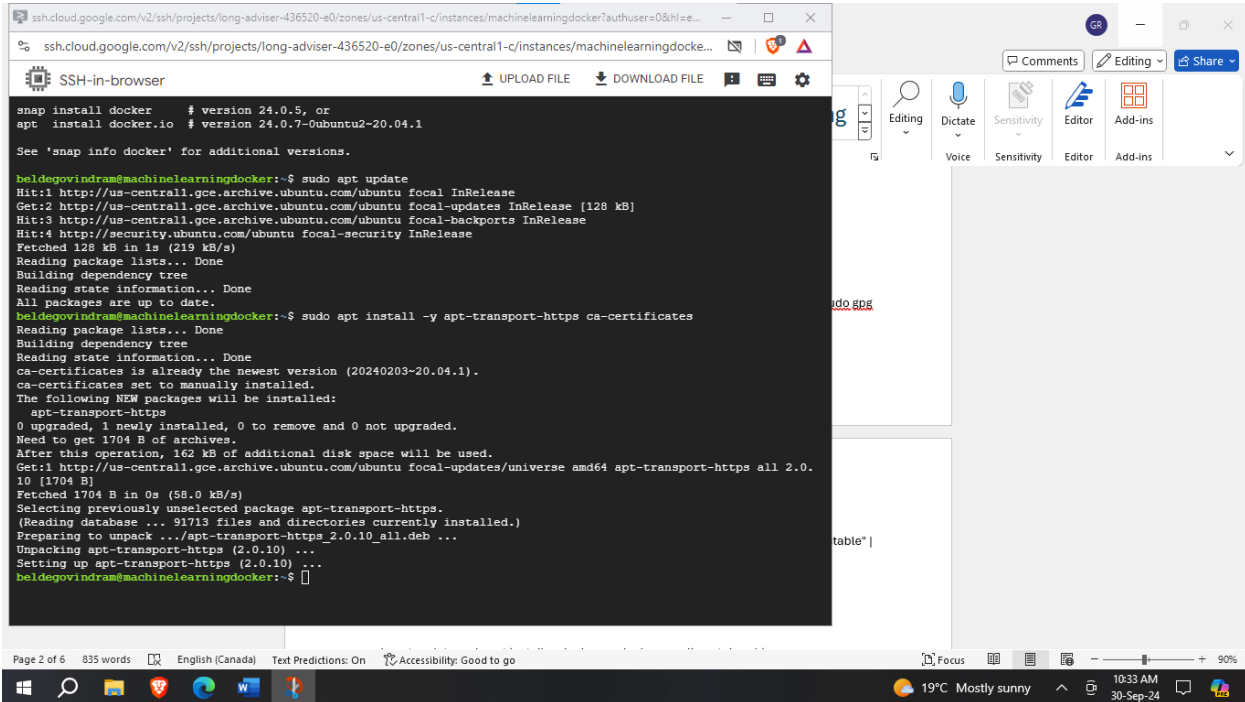
There are no versions of docker installed so we proceed.

2. Set Up the Docker Repository

- Run the following commands to set up the Docker repository:

```
sudo apt update
```

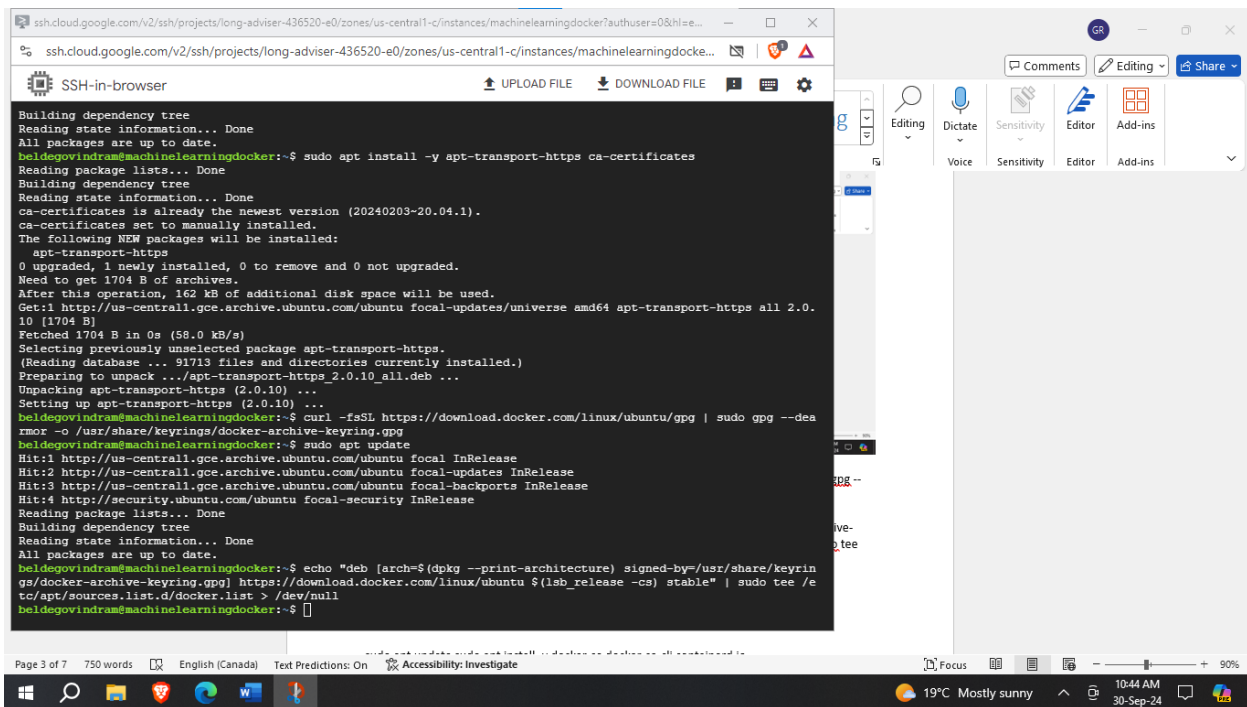
```
sudo apt install -y apt-transport-https ca-certificates
```



```
ssh.cloud.google.com/v2/ssh/projects/long-adviser-436520-e0/zones/us-central1-c/instances/machinelearningdocker?authuser=0&hl=e...
SSH-browser
snap install docker # version 24.0.5, or
apt install docker.io # version 24.0.7-0ubuntu2-20.04.1
See 'snap info docker' for additional versions.
beldegovindram@machinelearningdocker:~$ sudo apt update
Hit:1 http://us-central1.gce.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://us-central1.gce.archive.ubuntu.com/ubuntu focal-updates InRelease [128 kB]
Hit:3 http://us-central1.gce.archive.ubuntu.com/ubuntu focal-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu focal-security InRelease
Fetched 128 kB in 1s (219 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
All packages are up to date.
beldegovindram@machinelearningdocker:~$ sudo apt install -y apt-transport-https ca-certificates
Reading package lists... Done
Building dependency tree
Reading state information... Done
ca-certificates is already the newest version (20240203-20.04.1).
ca-certificates set to manually installed.
The following NEW packages will be installed:
  apt-transport-https
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 1704 B of archives.
After this operation, 162 kB of additional disk space will be used.
Get:1 http://us-central1.gce.archive.ubuntu.com/ubuntu focal-updates/universe amd64 apt-transport-https all 2.0.
10 [1704 B]
Fetched 1704 B in 0s (58.0 kB/s)
Selecting previously unselected package apt-transport-https.
(Reading database ... 91713 files and directories currently installed.)
Preparing to unpack .../apt-transport-https_2.0.10_all.deb ...
Unpacking apt-transport-https (2.0.10) ...
Setting up apt-transport-https (2.0.10) ...
beldegovindram@machinelearningdocker:~$
```

```
curl gnupg lsb-release curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --
dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

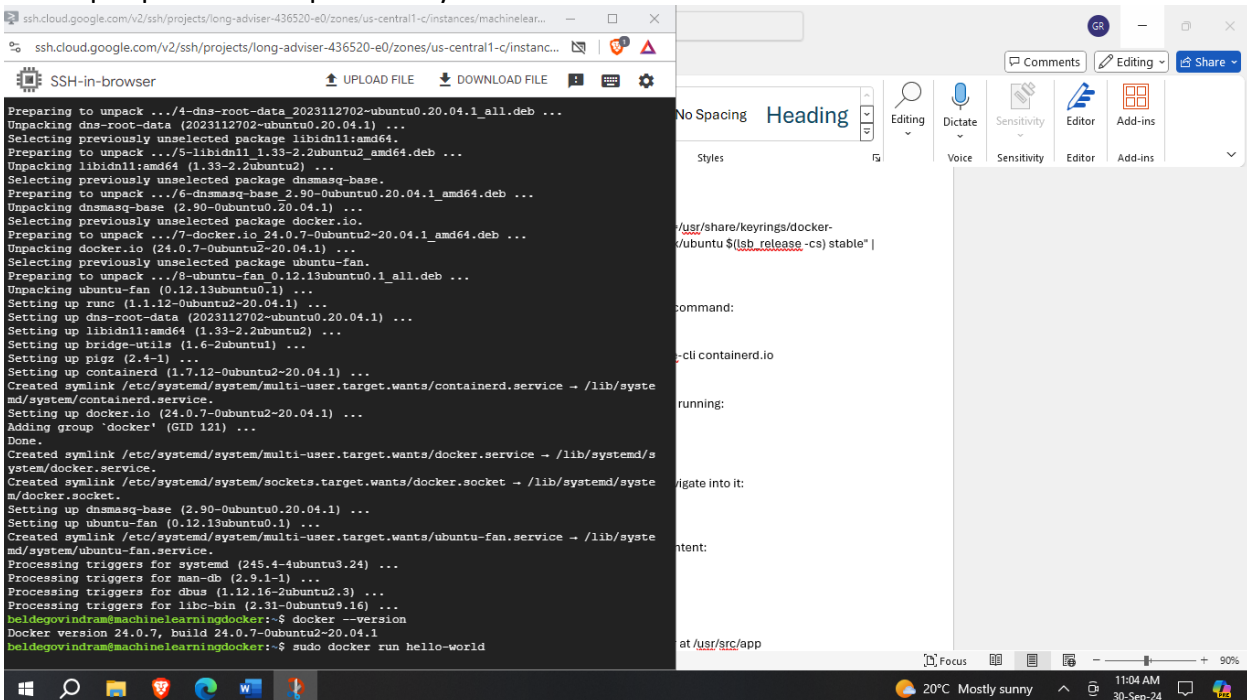
```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-
keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null
```



3. Install Docker Engine

- Install Docker Engine using the following command:

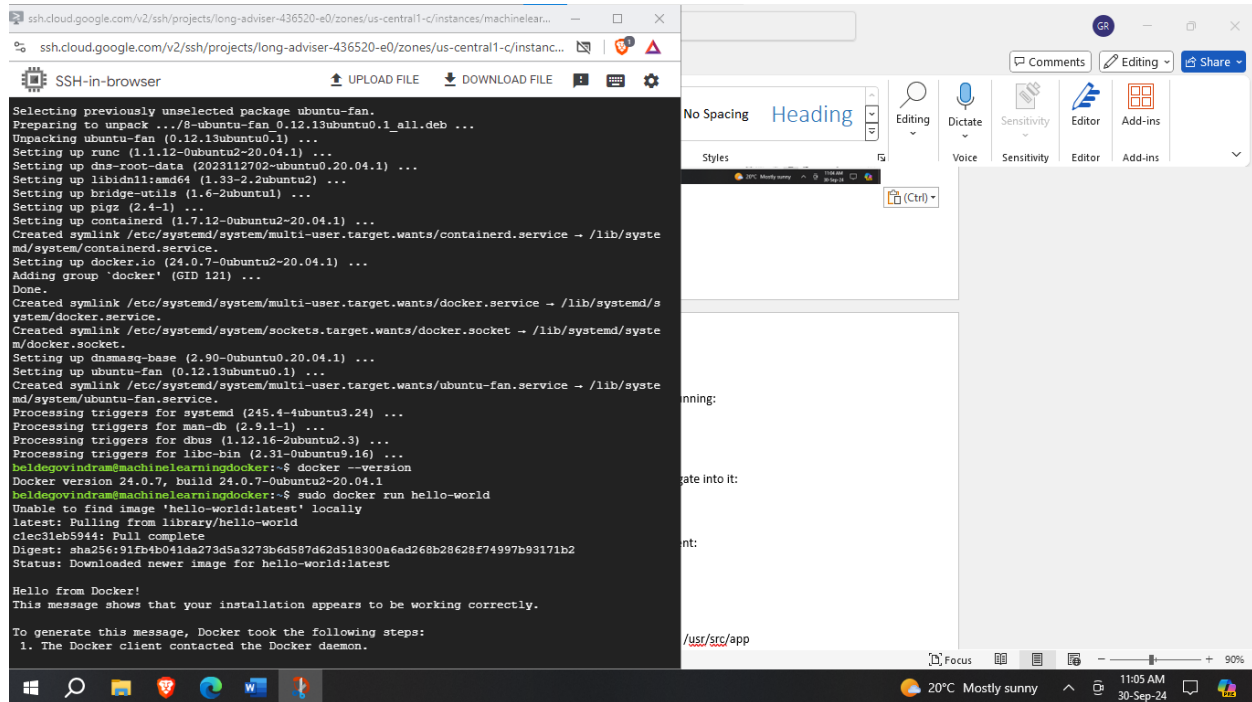
`sudo apt update`
`sudo apt install -y docker-ce docker-ce-cli containerd.io`



4. Verify Docker Installation

- Verify that Docker is installed correctly by running:

`sudo docker run hello-world`



The screenshot shows a terminal window with the following output:

```
Selecting previously unselected package ubuntu-fan.
Preparing to unpack .../8-ubuntu-fan_0.12.13ubuntu0.1_all.deb ...
Unpacking ubuntu-fan (0.12.13ubuntu0.1) ...
Setting up runc (1.1.12-0ubuntu2-20.04.1) ...
Setting up dns-root-data (2023112702-ubuntu0.20.04.1) ...
Setting up libidn11:amd64 (1.33-2.2ubuntu2) ...
Setting up bridge-utils (1.6-2ubuntu1) ...
Setting up pipg (2.4-1) ...
Setting up containerd (1.7.12-0ubuntu2-20.04.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /lib/systemd/system/containerd.service.
Setting up docker.io (24.0.7-0ubuntu2-20.04.1) ...
Adding group 'docker' (GID 121) ...
Done.
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /lib/systemd/system/docker.socket.
Setting up dnsmasq-base (2.90-0ubuntu0.20.04.1) ...
Setting up ubuntu-fan (0.12.13ubuntu0.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/ubuntu-fan.service → /lib/systemd/system/ubuntu-fan.service.
Processing triggers for systemd (245.4-4ubuntu3.24) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for dbus (1.12.16-2ubuntu2.3) ...
Processing triggers for libc-bin (2.31-0ubuntu9.16) ...
beldegovindram@machinelearningdocker:~$ docker --version
Docker version 24.0.7, build 24.0.7-0ubuntu2-20.04.1
beldegovindram@machinelearningdocker:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:91fb4b041da273d5a3273b6d587d62d518300a6ad268b28628f74997b93171b2
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
```

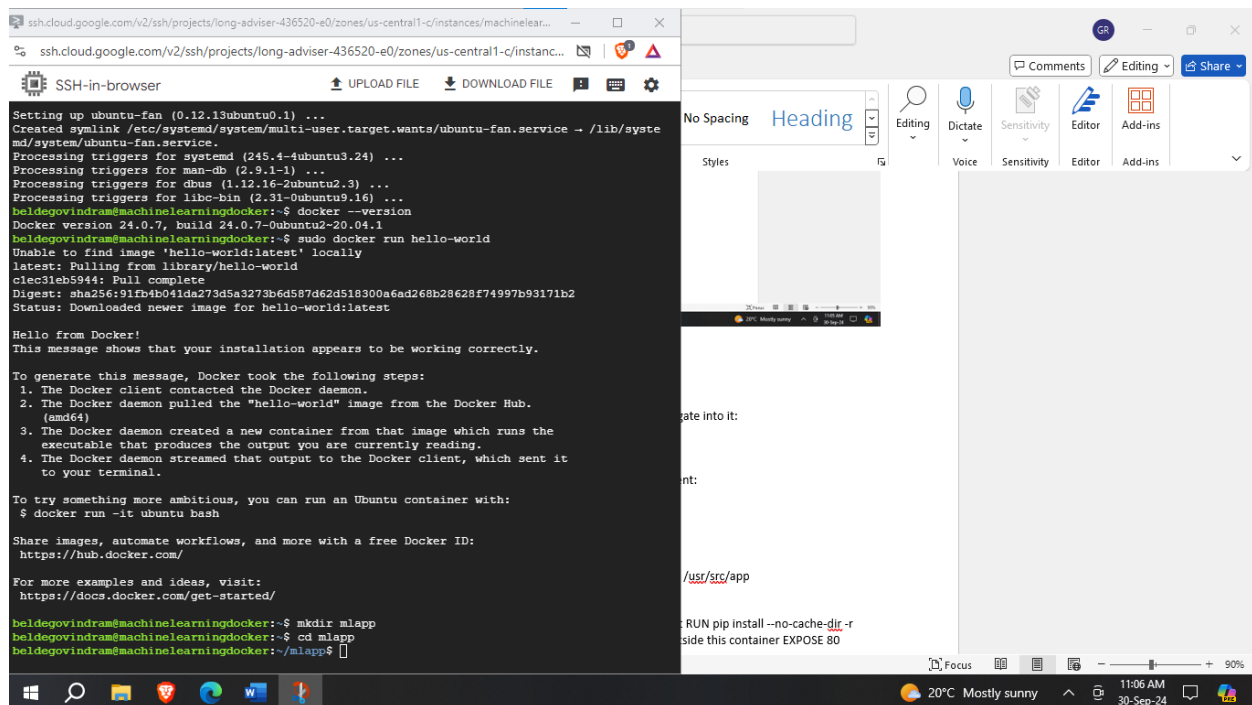
WE can see that docker is installed correctly.

Step 3: Create a Dockerfile for the ML Application

1. Create Project Directory

- Create a directory for your project and navigate into it:

```
mkdir ml-app cd ml-app
```



2. Create a Dockerfile

- Create a **Dockerfile** with the following content:

Use an official Python runtime as a parent image

FROM python:3.9-slim

Set the working directory

WORKDIR /usr/src/app

Copy the current directory contents into the container at /usr/src/app

*****YOU NEED TO WRITE COMMAND HERE*****

Install any needed packages specified in requirements.txt RUN pip install --no-cache-dir -r requirements.txt # Make port 80 available to the world outside this container EXPOSE 80

Run app.py when the container launches CMD ["python", "app.py"]

ssh.cloud.google.com/v2/projects/long-adviser-436520-e0/zones/us-central1-c/instances/machinelear...

ssh.cloud.google.com/v2/ssh/projects/long-adviser-436520-e0/zones/us-central1-c/instanc...

SSH-in-browser

UPLOAD FILE DOWNLOAD FILE

```
GNU nano 4.8 Dockerfile Modified
# Use an official Python runtime as a parent image
FROM python:3.9-slim

# Set the working directory in the container
WORKDIR /usr/src/app

# Copy the current directory contents into the container
COPY . .

# Install any needed packages specified in requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 80

# Run app.py when the container launches
CMD ["python", "app.py"]
```

Get Help Exit Write Out Read File Where Is Replace Cut Text Paste Text Justify To Spell Cur Pos Go To Line

Focus 20°C Mostly sunny 11:14 AM 30-Sep-24

ssh.cloud.google.com/v2/projects/long-adviser-436520-e0/zones/us-central1-c/instances/machinelear...

ssh.cloud.google.com/v2/ssh/projects/long-adviser-436520-e0/zones/us-central1-c/instanc...

SSH-in-browser

UPLOAD FILE DOWNLOAD FILE

```
beldegovindram@machinelearningdocker:~$ mkdir mlapp
beldegovindram@machinelearningdocker:~$ cd mlapp
beldegovindram@machinelearningdocker:~/mlapp$ FROM python:3.9-slim
FROM: command not found
beldegovindram@machinelearningdocker:~/mlapp$ touch Dockerfile
beldegovindram@machinelearningdocker:~/mlapp$ pip freeze > requirements.txt
Command 'pip' not found, but can be installed with:
apt install python3-pip
Please ask your administrator.
beldegovindram@machinelearningdocker:~/mlapp$ apt install python3-pip
E: Could not open lock file /var/lib/dpkg/lock-frontent - open (13: Permission denied)
E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontent), are you root?
beldegovindram@machinelearningdocker:~/mlapp$ #use an official python runtime as a parent i
mage
beldegovindram@machinelearningdocker:~/mlapp$ FROM PYTHON:3.9-SLIM
FROM: command not found
beldegovindram@machinelearningdocker:~/mlapp$ nano Dockerfile
beldegovindram@machinelearningdocker:~/mlapp$ cat Dockerfile
# Use an official Python runtime as a parent image
FROM python:3.9-slim

# Set the working directory in the container
WORKDIR /usr/src/app

# Copy the current directory contents into the container
COPY . .

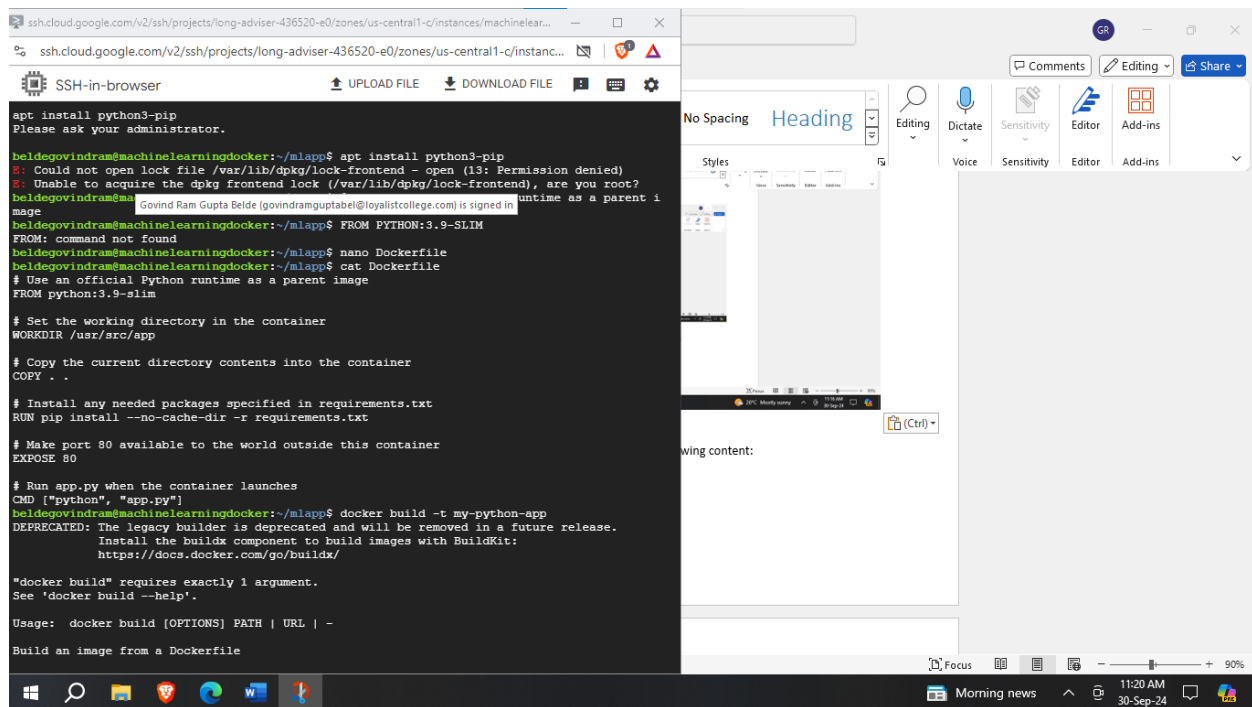
# Install any needed packages specified in requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 80

# Run app.py when the container launches
CMD ["python", "app.py"]
beldegovindram@machinelearningdocker:~/mlapp$
```

Govind Ram Gupta Belde (govindramguptabel@loyalistcollege.com) is signed in

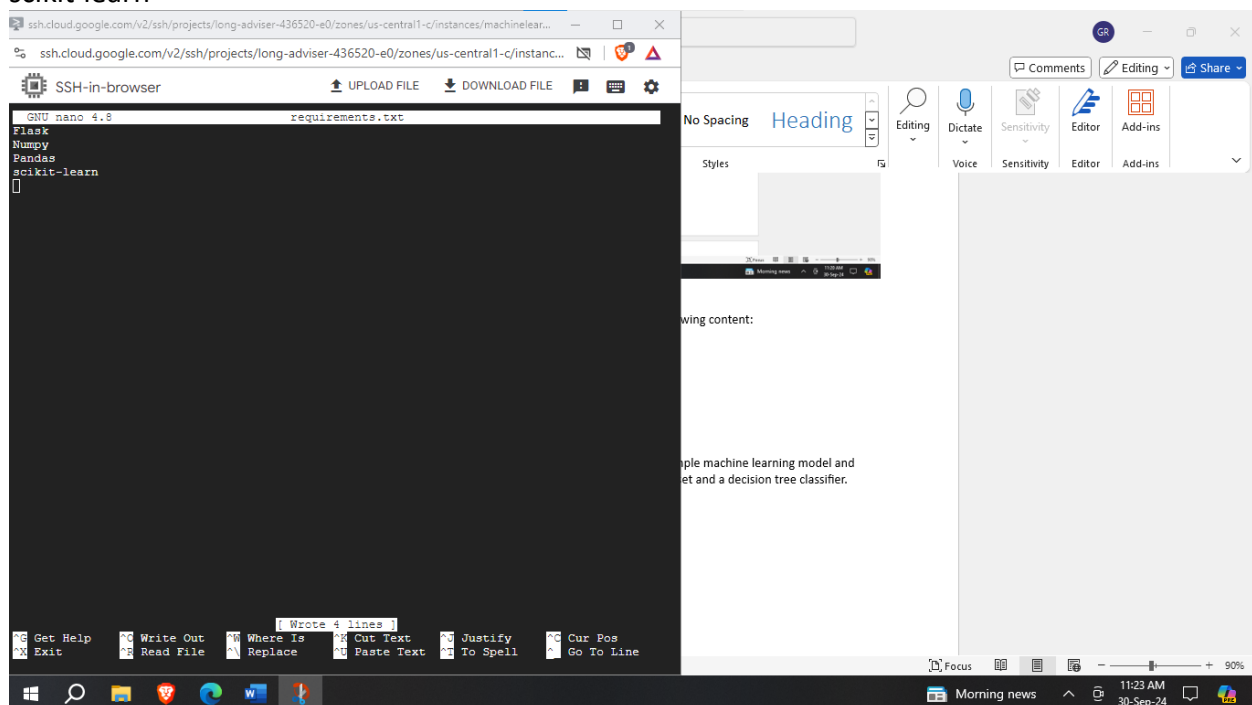
Focus 20°C Mostly sunny 11:16 AM 30-Sep-24

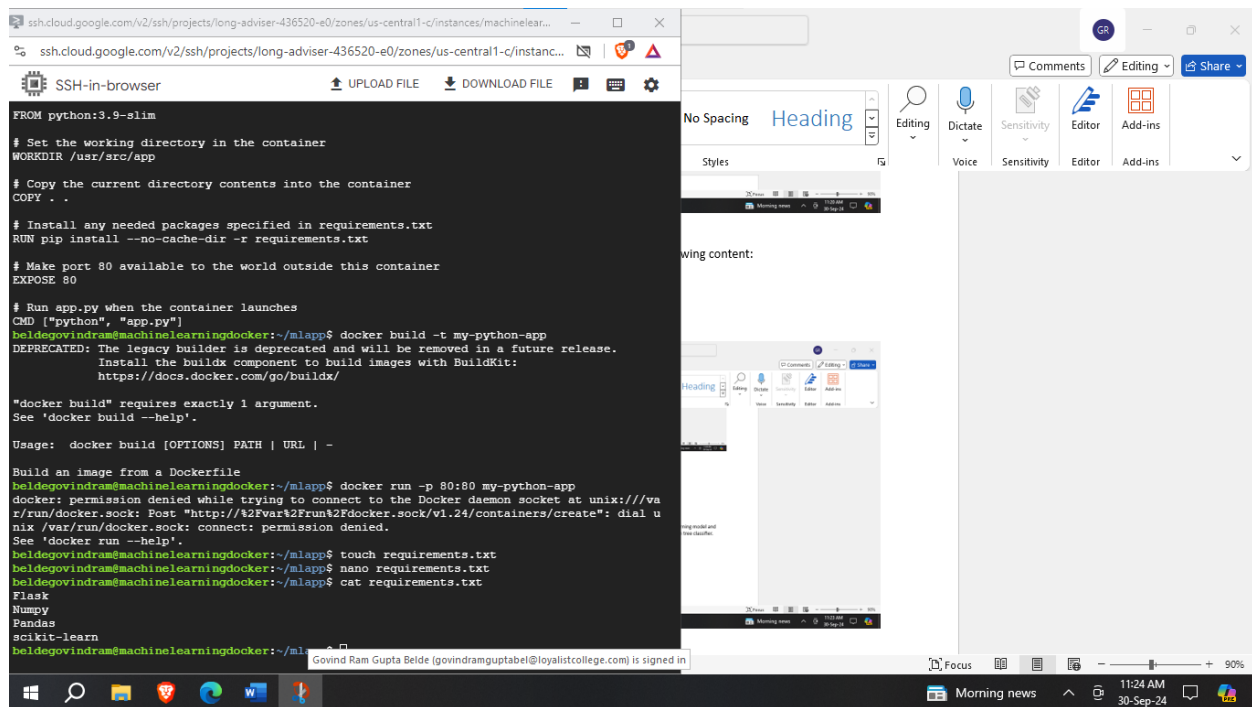


3. Create requirements.txt File

- Create a **requirements.txt** file with the following content:

Flask
Numpy
Pandas
scikit-learn





Step 4: Develop the Machine Learning Application

1. Create a Simple ML Model

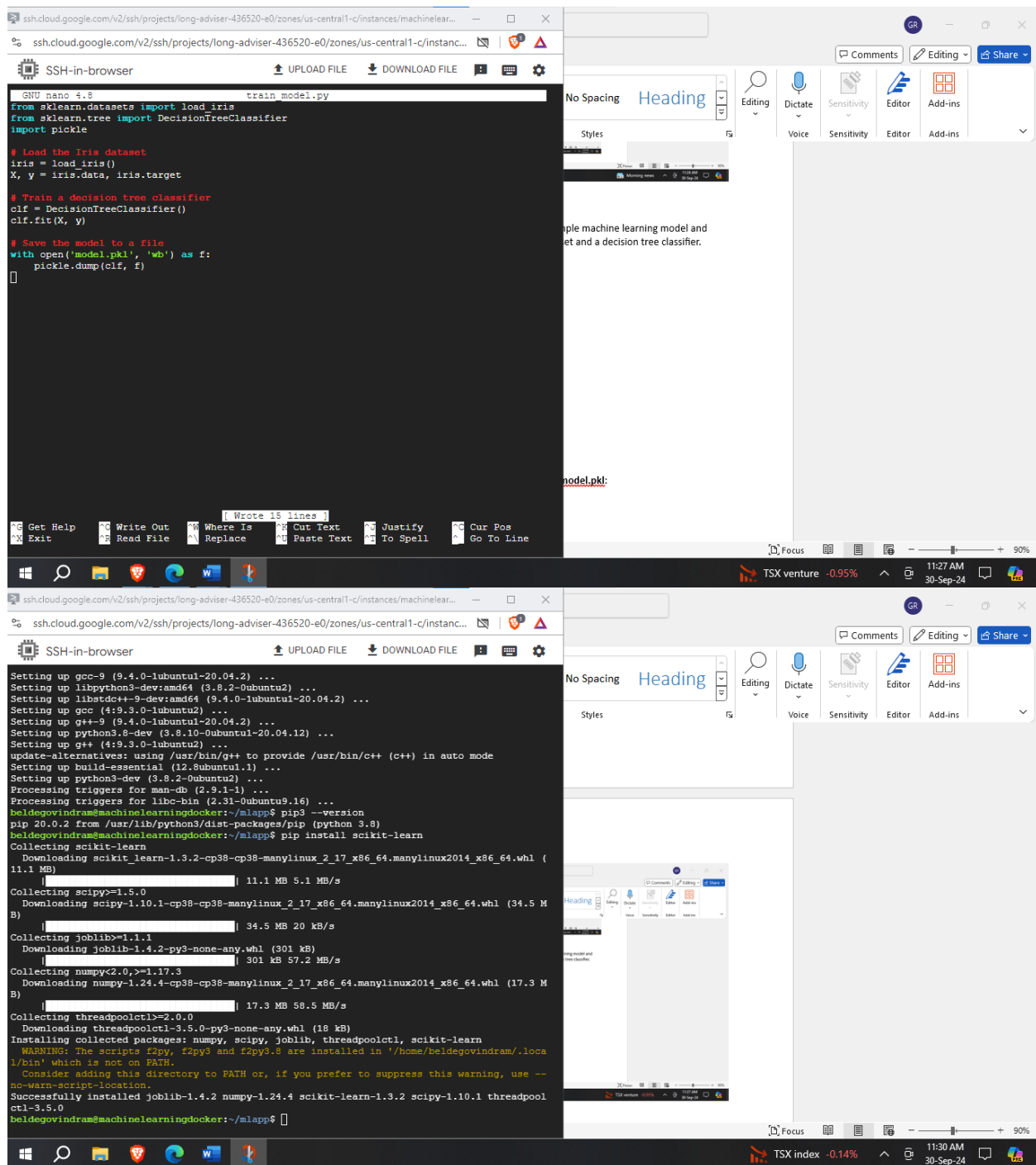
- Create a script **train_model.py** to train a simple machine learning model and save it. For simplicity, we'll use the Iris dataset and a decision tree classifier.

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
import pickle
```

```
# Load the Iris dataset
iris = load_iris()
X, y = iris.data, iris.target
```

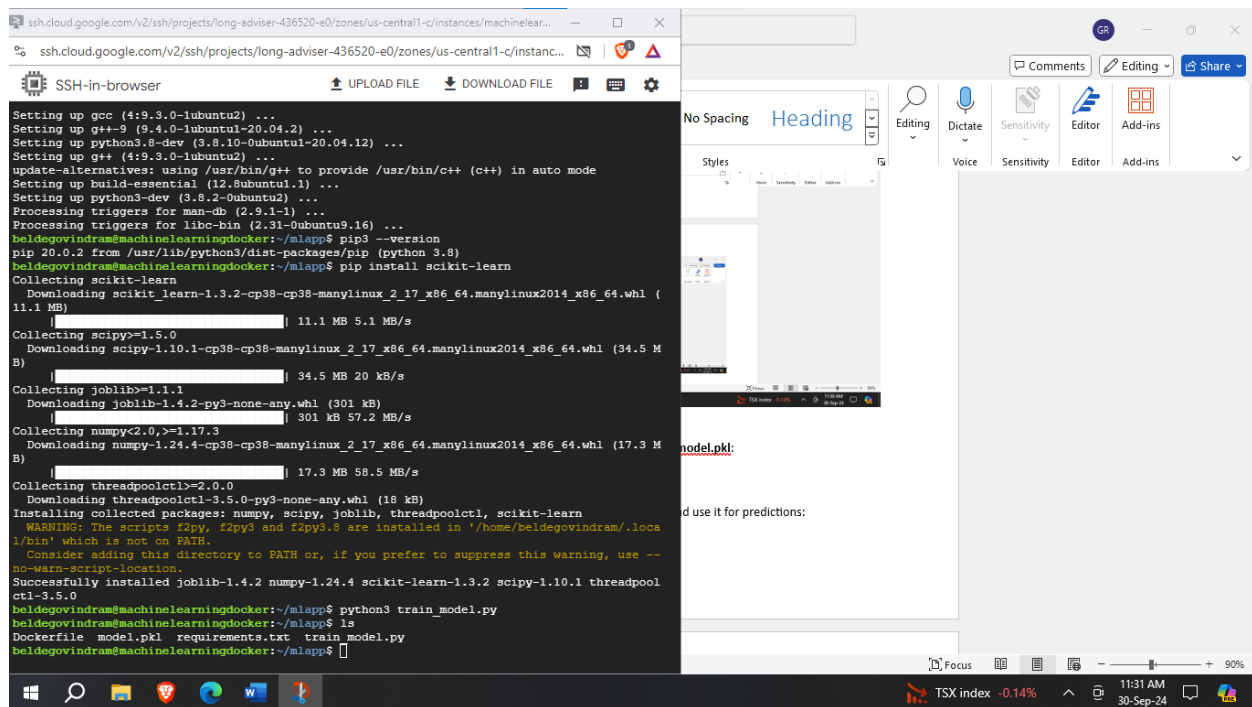
```
# Train a decision tree classifier
clf = DecisionTreeClassifier()
clf.fit(X, y)
```

```
# Save the model to a file
with open('model.pkl', 'wb') as f:
    pickle.dump(clf, f)
```



2. Run the Model Training Script

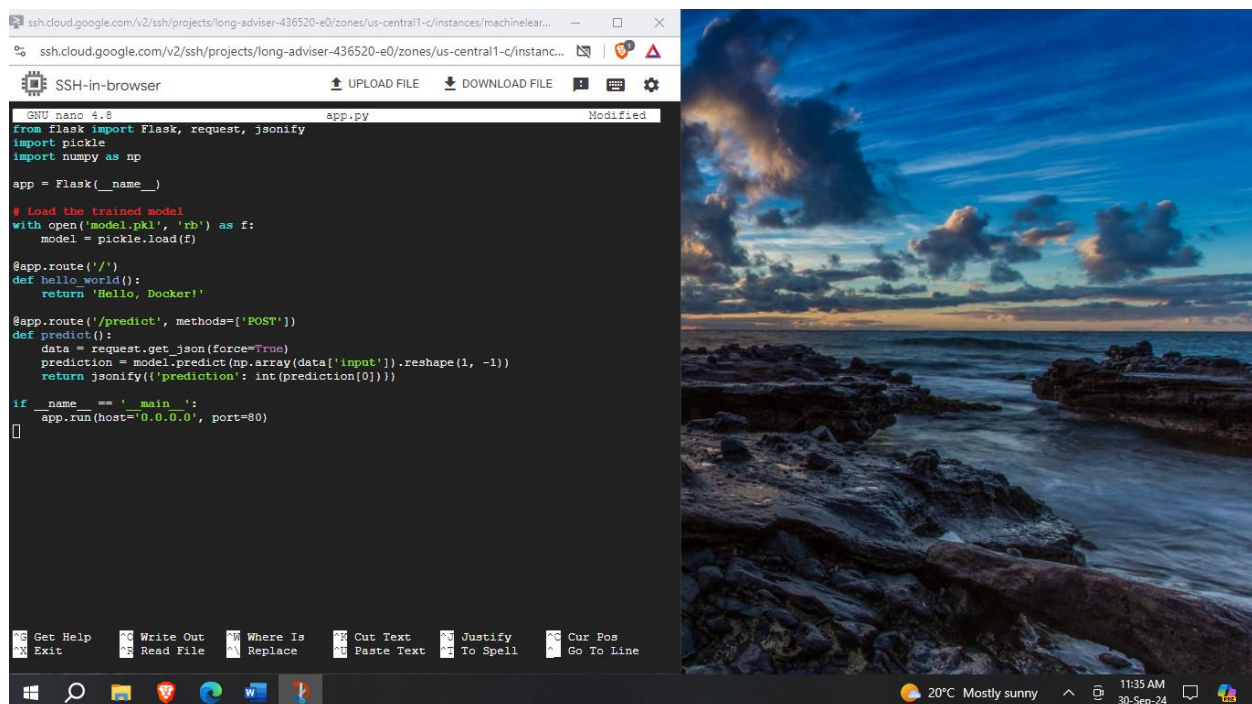
- Run the **train_model.py** script to generate **model.pkl**:
`python train_model.py`



Successfully trained the model

3. Integrate the Model into the Flask App

- Update **app.py** to load the trained model and use it for predictions:



from flask import Flask, request, jsonify

```

import pickle
import numpy as np

app = Flask(__name__)

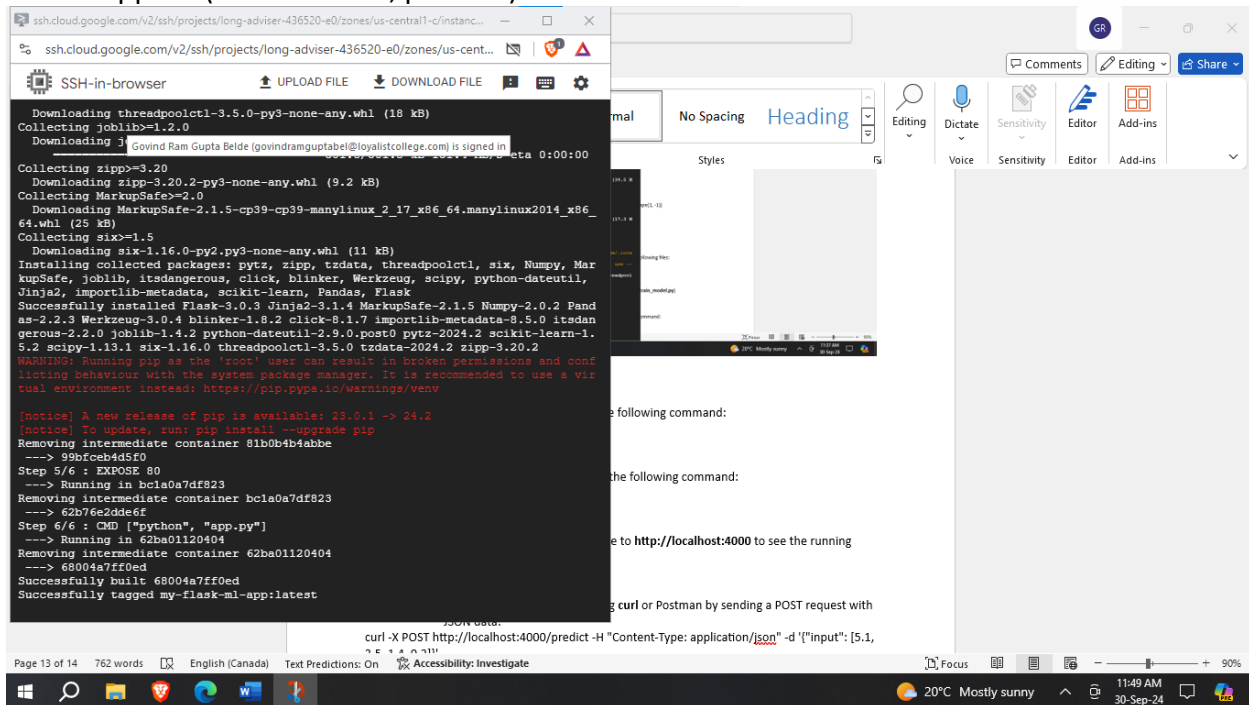
# Load the trained model
with open('model.pkl', 'rb') as f:
    model = pickle.load(f)

@app.route('/')
def hello_world():
    return 'Hello, Docker!'

@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json(force=True)
    prediction = model.predict(np.array(data['input']).reshape(1, -1))
    return jsonify({'prediction': int(prediction[0])})

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=80)

```



4. Update the Project Directory

- Ensure your project directory contains the following files:
 - **Dockerfile**
 - **requirements.txt**

- **train_model.py**
- **app.py**
- **model.pkl** (generated after running **train_model.py**)

The screenshot shows a terminal window with the following content:

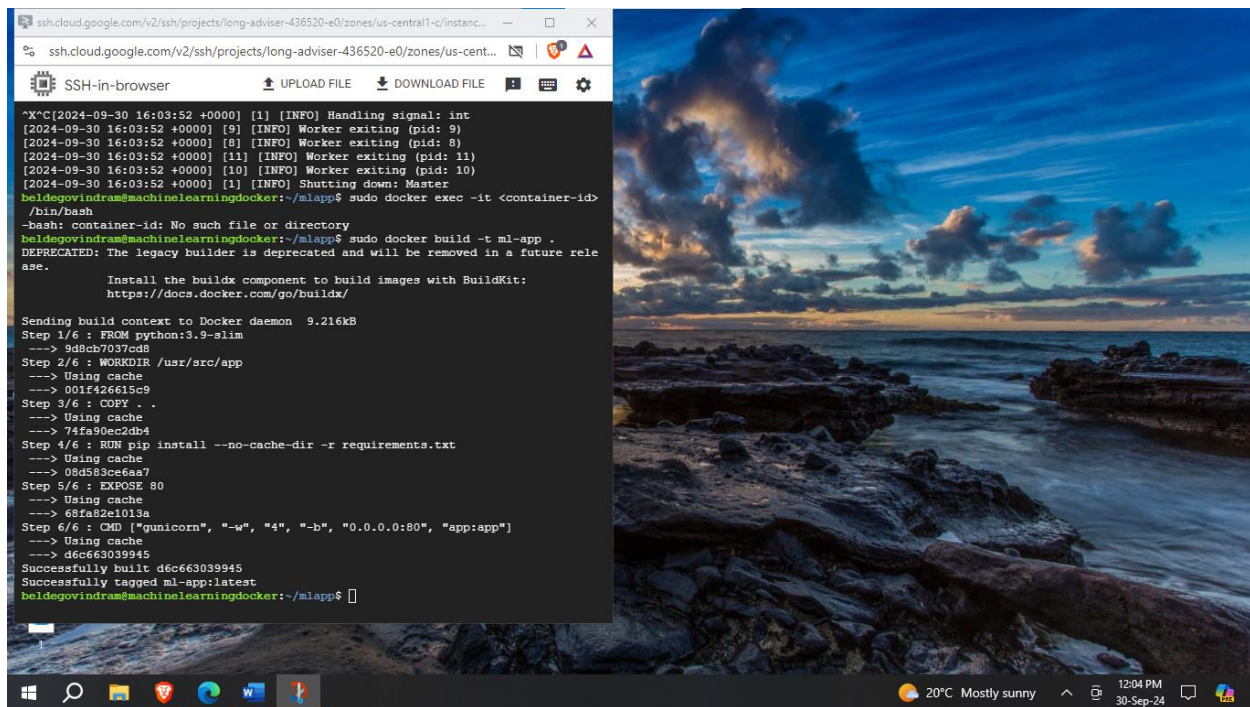
```
Setting up build-essential (12.8ubuntu1.1) ...
Setting up python3-dev (3.8.2-0ubuntu2) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.16) ...
beldegovindram@machinelearningdocker:~/mlapp$ pip3 --version
pip 20.0.2 from /usr/lib/python3/dist-packages/pip (python 3.8)
beldegovindram@machinelearningdocker:~/mlapp$ pip install scikit-learn
Collecting scikit-learn
  Downloading scikit_learn-1.3.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.1 MB)
    |#####| 11.1 MB 5.1 MB/s
Collecting scipy>=1.5.0
  Downloading scipy-1.10.1-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (34.5 MB)
    |#####| 34.5 MB 20 kB/s
Collecting joblib>=1.1.1
  Downloading joblib-1.4.2-py3-none-any.whl (301 kB)
    |#####| 301 kB 57.2 MB/s
Collecting numpy<2.0,>=1.17.3
  Downloading numpy-1.24.4-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3 MB)
    |#####| 17.3 MB 58.5 MB/s
Collecting threadpoolctl>=2.0.0
  Downloading threadpoolctl-3.5.0-py3-none-any.whl (18 kB)
Installing collected packages: numpy, scipy, joblib, threadpoolctl, scikit-learn
WARNING: The scripts f2py, f2py3 and f2py3.8 are installed in '/home/beldegovindram/.local/bin' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed joblib-1.4.2 numpy-1.24.4 scikit-learn-1.3.2 scipy-1.10.1 threadpoolctl-3.5.0
beldegovindram@machinelearningdocker:~/mlapp$ python3 train_model.py
beldegovindram@machinelearningdocker:~/mlapp$ ls
Dockerfile  model.pkl  requirements.txt  train_model.py
beldegovindram@machinelearningdocker:~/mlapp$ touch app.py
beldegovindram@machinelearningdocker:~/mlapp$ nano app.py
beldegovindram@machinelearningdocker:~/mlapp$ python3 train_model.py
beldegovindram@machinelearningdocker:~/mlapp$ ls
Dockerfile  app.py  model.pkl  requirements.txt  train_model.py
beldegovindram@machinelearningdocker:~/mlapp$ doc[
```

Step 5: Build and Run the Docker Container

1. Build the Docker Image

- Build the Docker image with the following command:

`sudo docker build -t ml-app .`



2. Run the Docker Container

- Run the Docker container with the following command:

```
sudo docker run -p 4000:80 ml-app
```


ssh.cloud.google.com/v2/ssh/projects/long-adviser-436520-e0/zones/us-central1-c/instanc... — □ ×

ssh.cloud.google.com/v2/ssh/projects/long-adviser-436520-e0/zones/us-cent... | 🔒 🔔 🔧

SSH-in-browser UPLOAD FILE DOWNLOAD FILE

```
beldegovindram@machinelearningdoker:~/mlapp$ sudo docker build -t ml-app .
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/

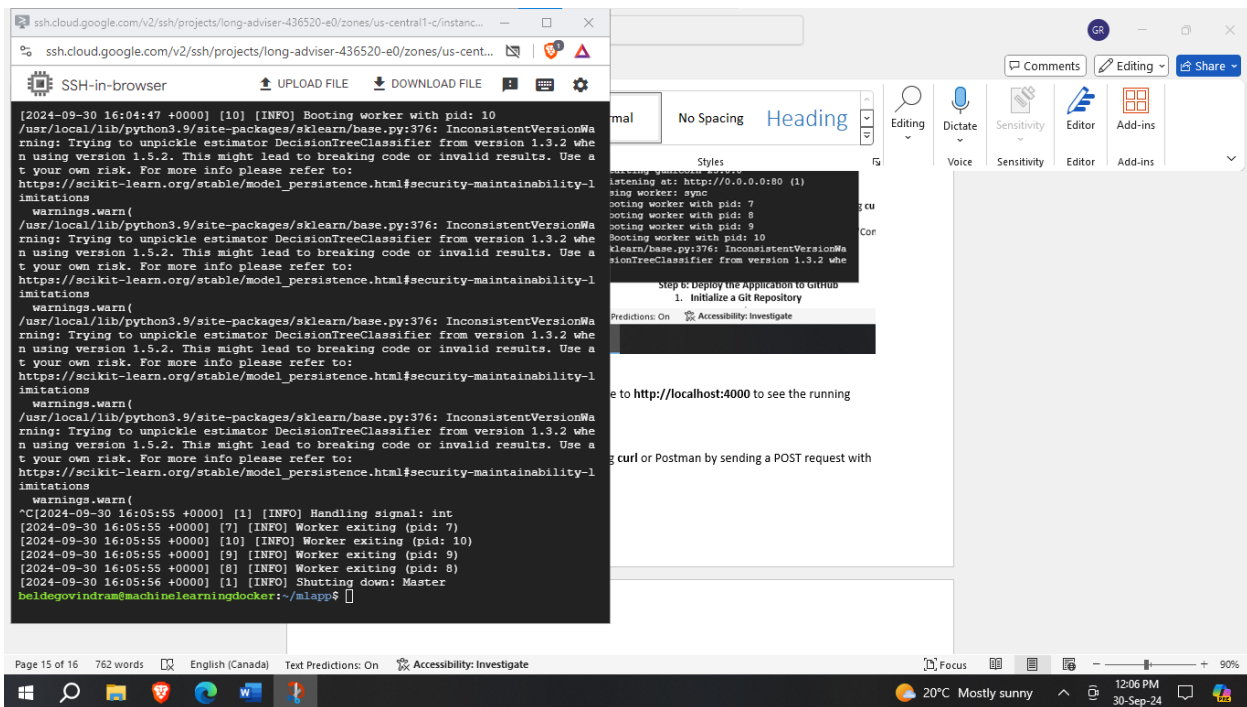
Sending build context to Docker daemon  9.216kB
Step 1/6 : FROM python:3.9-slim
--> 9d8cb7037cd8
Step 2/6 : WORKDIR /usr/src/app
--> Using cache
--> 001f426615c9
Step 3/6 : COPY . .
--> Using cache
--> 74fa90ec2db4
Step 4/6 : RUN pip install --no-cache-dir -r requirements.txt
--> Using cache
--> 08d583ce6aa7
Step 5/6 : EXPOSE 80
--> Using cache
--> 68fa82e1013a
Step 6/6 : CMD ["gunicorn", "-w", "4", "-b", "0.0.0.0:80", "app:app"]
--> Using cache
--> d6c663039945
Successfully built d6c663039945
Successfully tagged ml-app:latest
beldegovindram@machinelearningdoker:~/mlapp$ sudo docker run -p 4000:80 ml-app
[2024-09-30 16:04:47 +0000] [1] [INFO] Starting gunicorn 23.0.0
[2024-09-30 16:04:47 +0000] [1] [INFO] Listening at: http://0.0.0.0:80 (1)
[2024-09-30 16:04:47 +0000] [1] [INFO] Using worker: sync
[2024-09-30 16:04:47 +0000] [7] [INFO] Booting worker with pid: 7
[2024-09-30 16:04:47 +0000] [8] [INFO] Booting worker with pid: 8
[2024-09-30 16:04:47 +0000] [9] [INFO] Booting worker with pid: 9
[2024-09-30 16:04:47 +0000] [10] [INFO] Booting worker with pid: 10
/usr/local/lib/python3.9/site-packages/sklearn/base.py:376: InconsistentVersionWarning: Trying to unpickle estimator DecisionTreeClassifier from version 1.3.2 whe
```

Step 6: Deploy the Application to GitHub
1. Initialize a Git Repository

Page 14 of 15 762 words English (Canada) Text Predictions: On Accessibility: Investigate

3. Access the Application

- Open your browser and navigate to **http://localhost:4000** to see the running application.



4. Test the ML Endpoint

- Test the **/predict** endpoint using **curl** or Postman by sending a POST request with JSON data:

`curl -X POST http://localhost:4000/predict -H "Content-Type: application/json" -d '{"input": [5.1, 3.5, 1.4, 0.2]}'`

Step 6: Deploy the Application to GitHub

1. Initialize a Git Repository

- Initialize a Git repository in your project directory:

`git init`

2. Add All Files and Commit

- Add all files to the repository and commit:

`git add .`

`git commit -m "Initial commit"`

3. Create a New Repository on GitHub

- Create a new repository on GitHub and follow the instructions to push your local repository to GitHub:

`git remote add origin https://github.com/yourusername/your-repository.git`

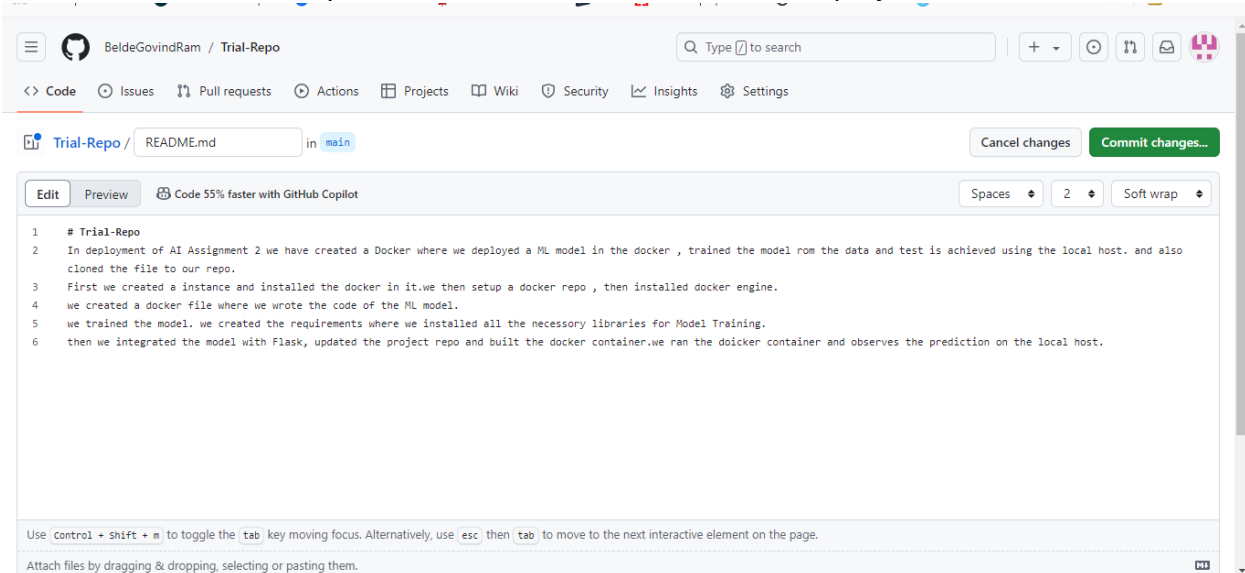
`git branch -M main`

`git push -u origin main`

Step 7: Document the Process

1. Create a README.md File

- Document the process in a **README.md** file in your repository. Include the following:
 - Overview of the project
 - Instructions to build and run the Docker container
 - Instructions to test the ML endpoint
 - Any other relevant information about the project



Submission

- Take screenshots of every step you perform and paste in the submission word/pdf file.
- Submit the GitHub repository link of your project.
- Ensure your repository is public and the README.md file is well-documented.