In-class Assignment 2

Instructor: Qasim Ali

Develop and Deploy a Machine Learning Application using Docker

Group Name: Group C

Student Name: Bibek Khadka

Objective

The objective of this assignment is to develop a simple machine learning application, containerize it using Docker, and deploy it on GitHub. This exercise will help you understand the principles of containerization, version control, and basic machine learning application development.

Prerequisites

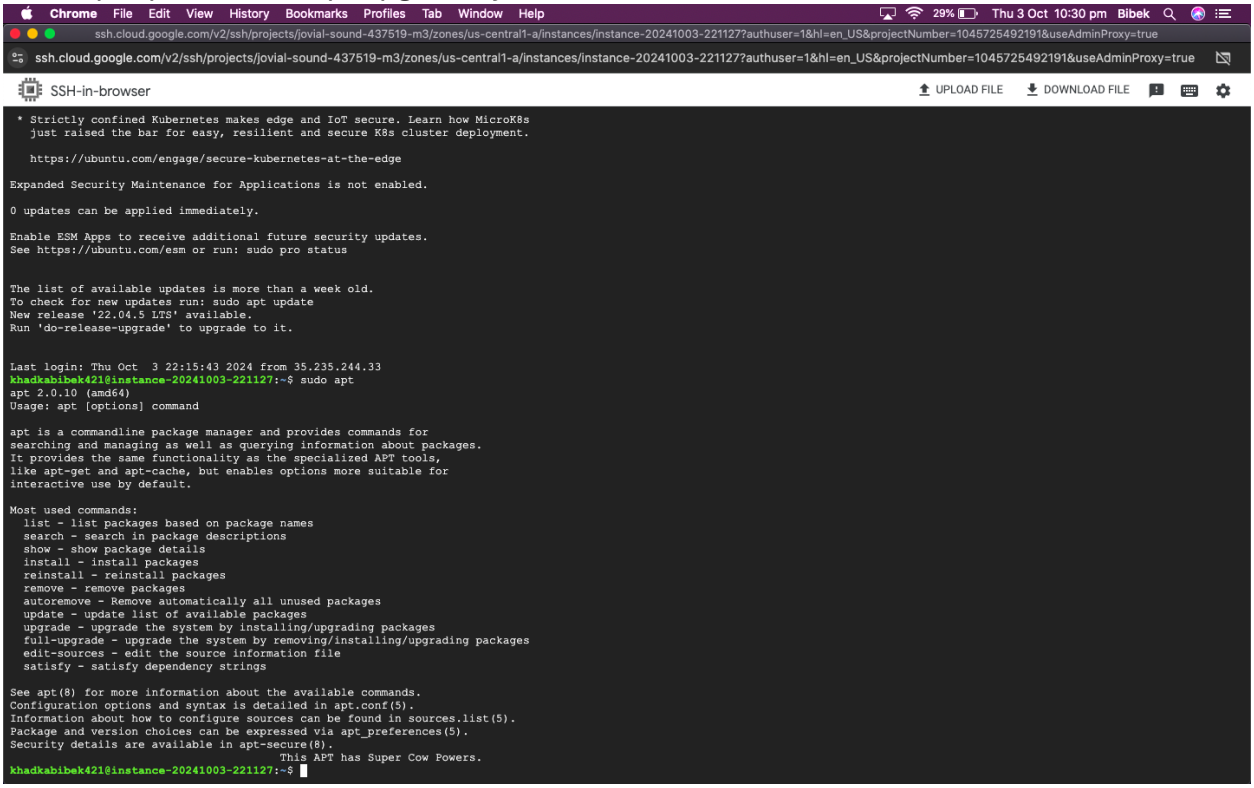
- Basic understanding of Python programming
- Basic understanding of machine learning concepts
- Familiarity with Git and GitHub
- Basic knowledge of Docker

Assignment Steps

Step 1: Set Up the VM

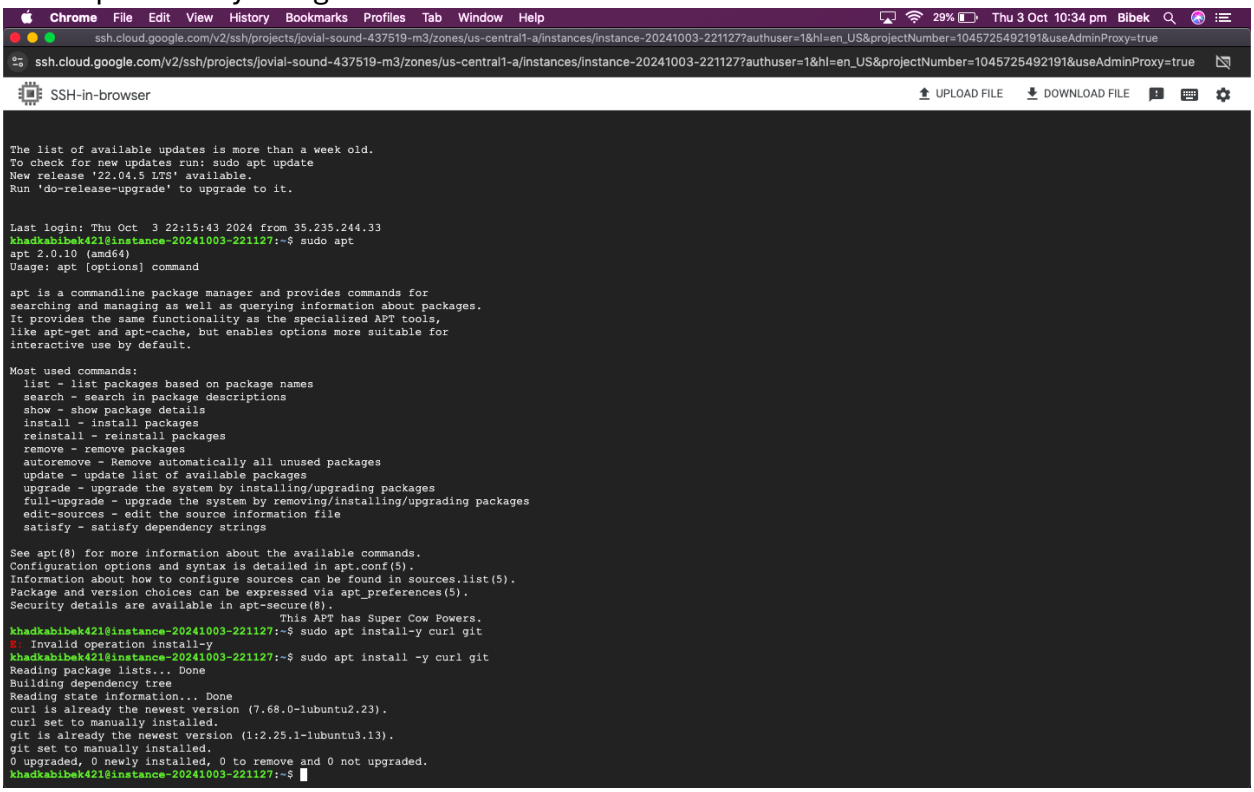
1. **Update the System**
    - Ensure your VM is running an updated version of Ubuntu. Run the following
    - commands:

sudo apt update sudo apt upgrade -y



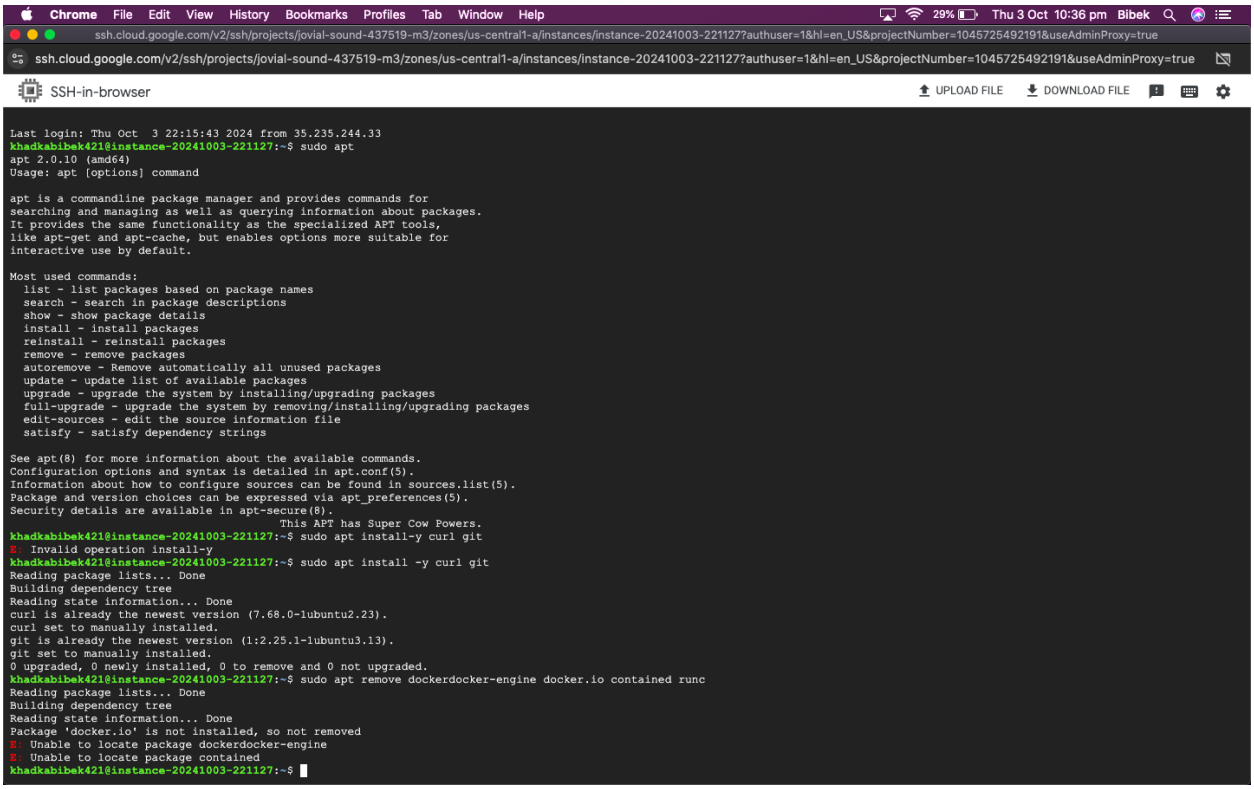
2. **Install Necessary Packages**
    - Install curl and git:

sudo apt install -y curl git



Step 2: Install Docker

1. **Remove Old Versions**
    - Remove any old versions of Docker if present:



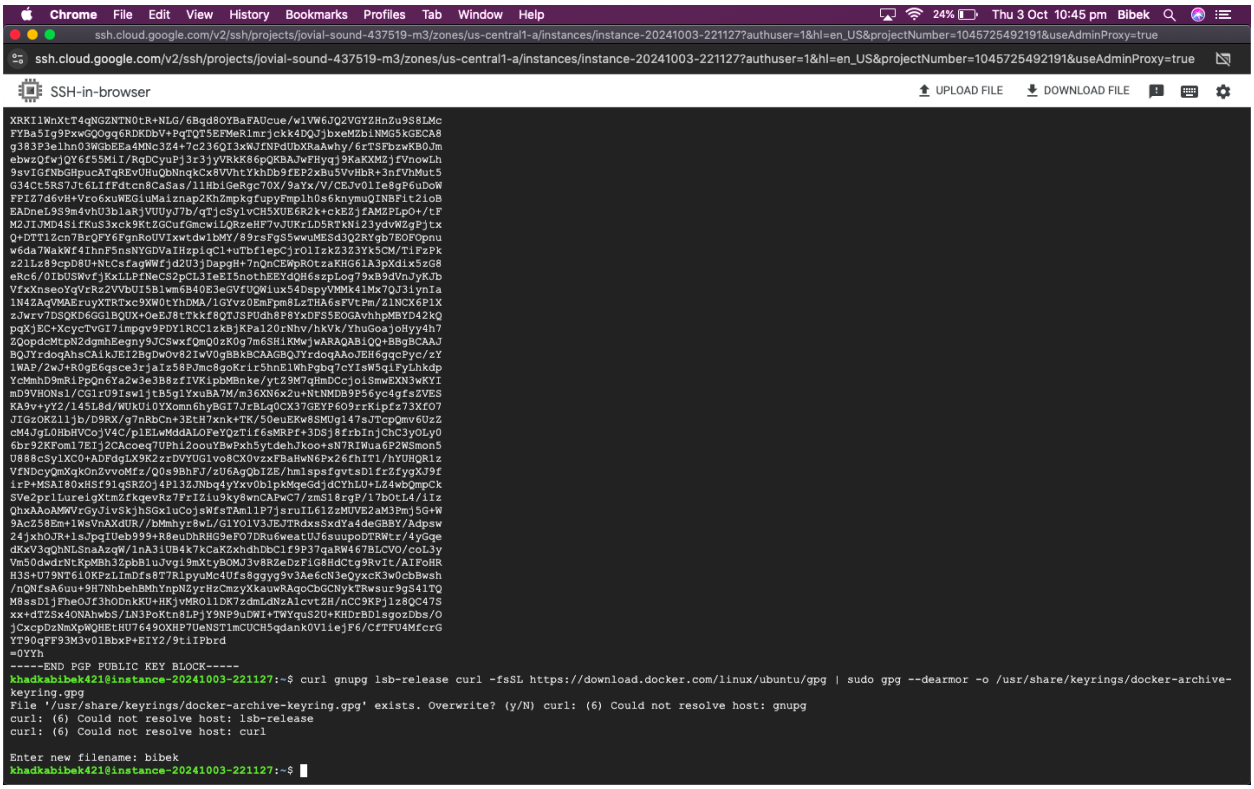
sudo apt remove docker docker-engine docker.io containerd runc

2. **Set Up the Docker Repository**
    - Run the following commands to set up the Docker repository:

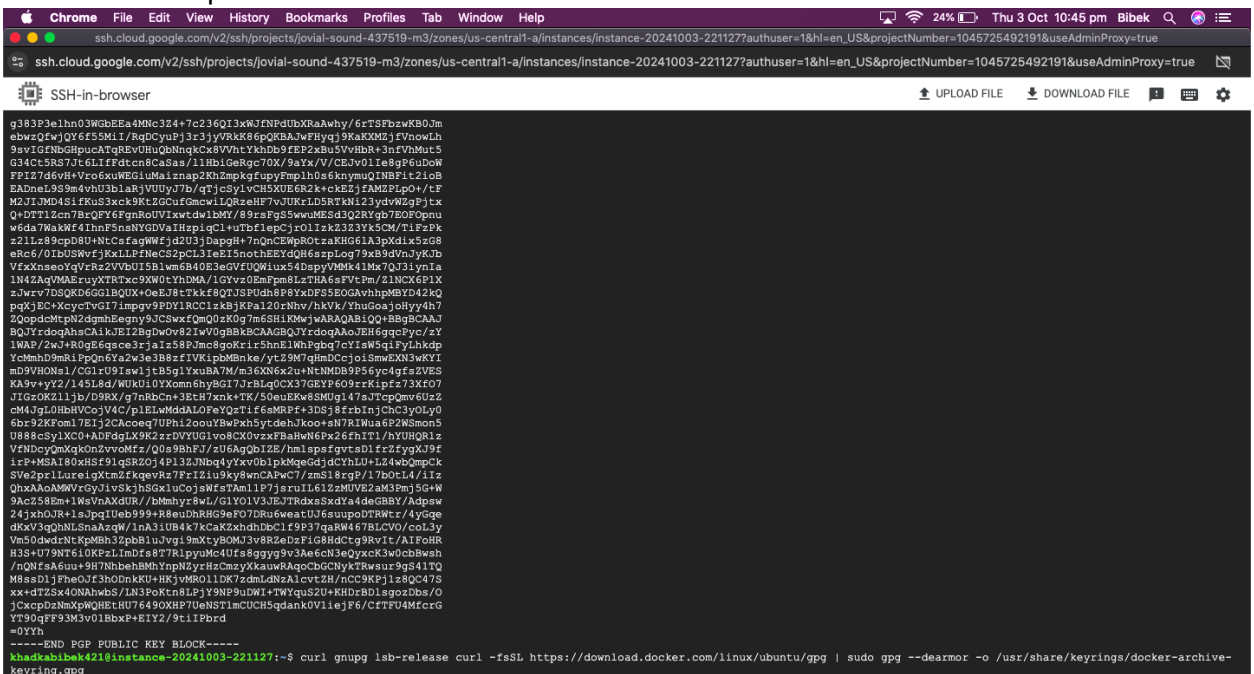
sudo apt update

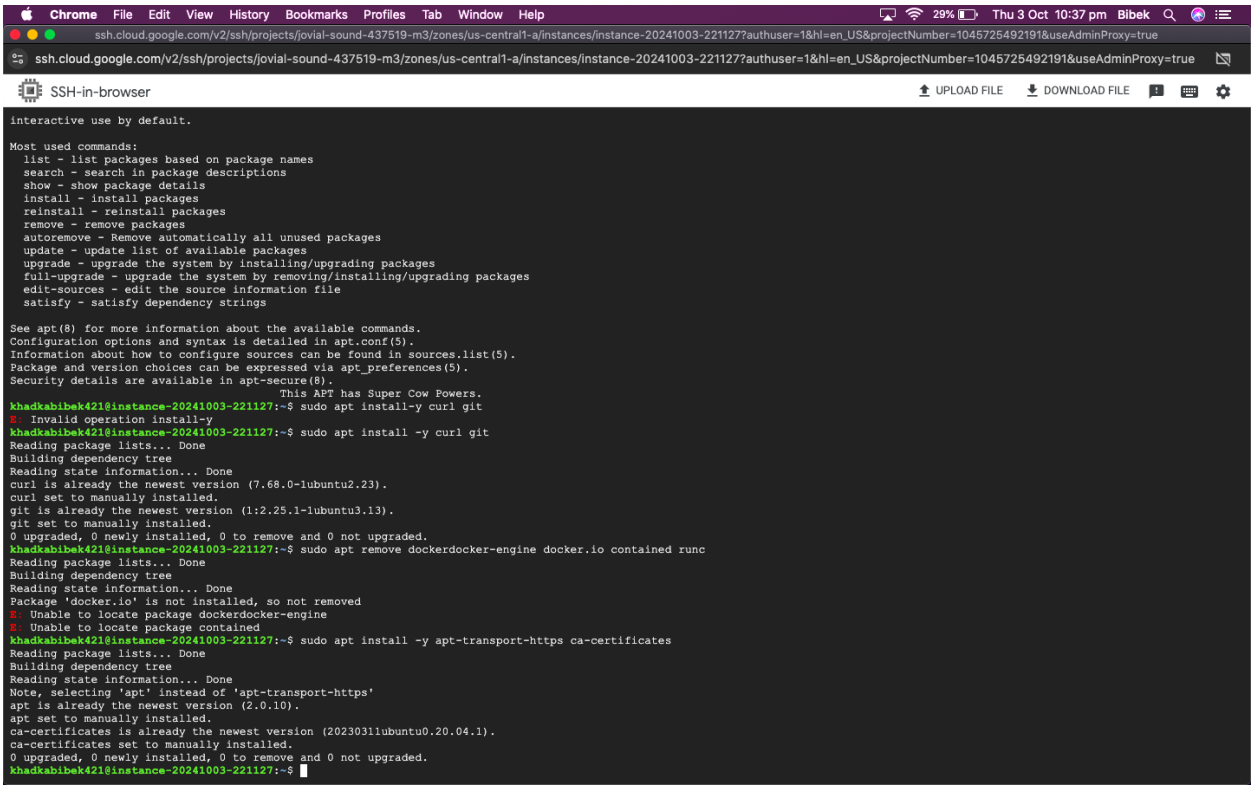
sudo apt install -y apt-transport-https ca-certificates

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



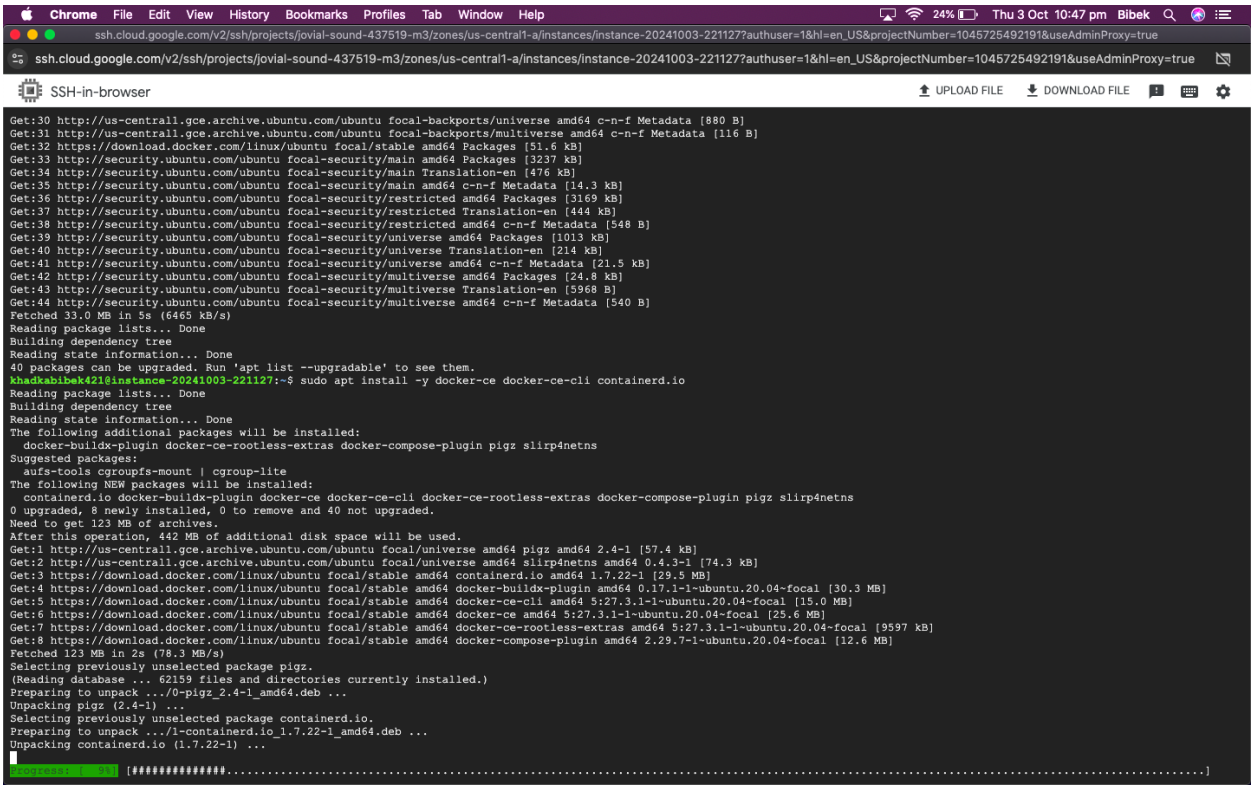
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null





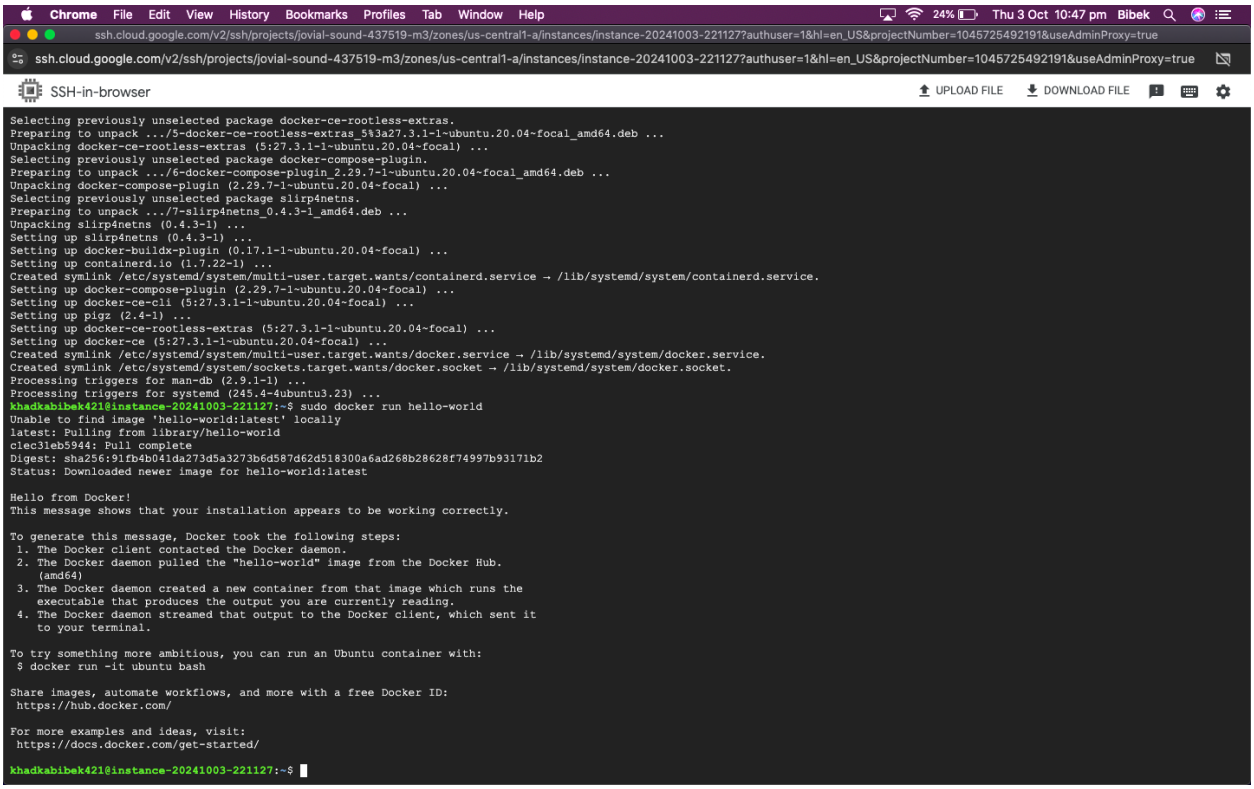
3. **Install Docker Engine**
    - Install Docker Engine using the following command:

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



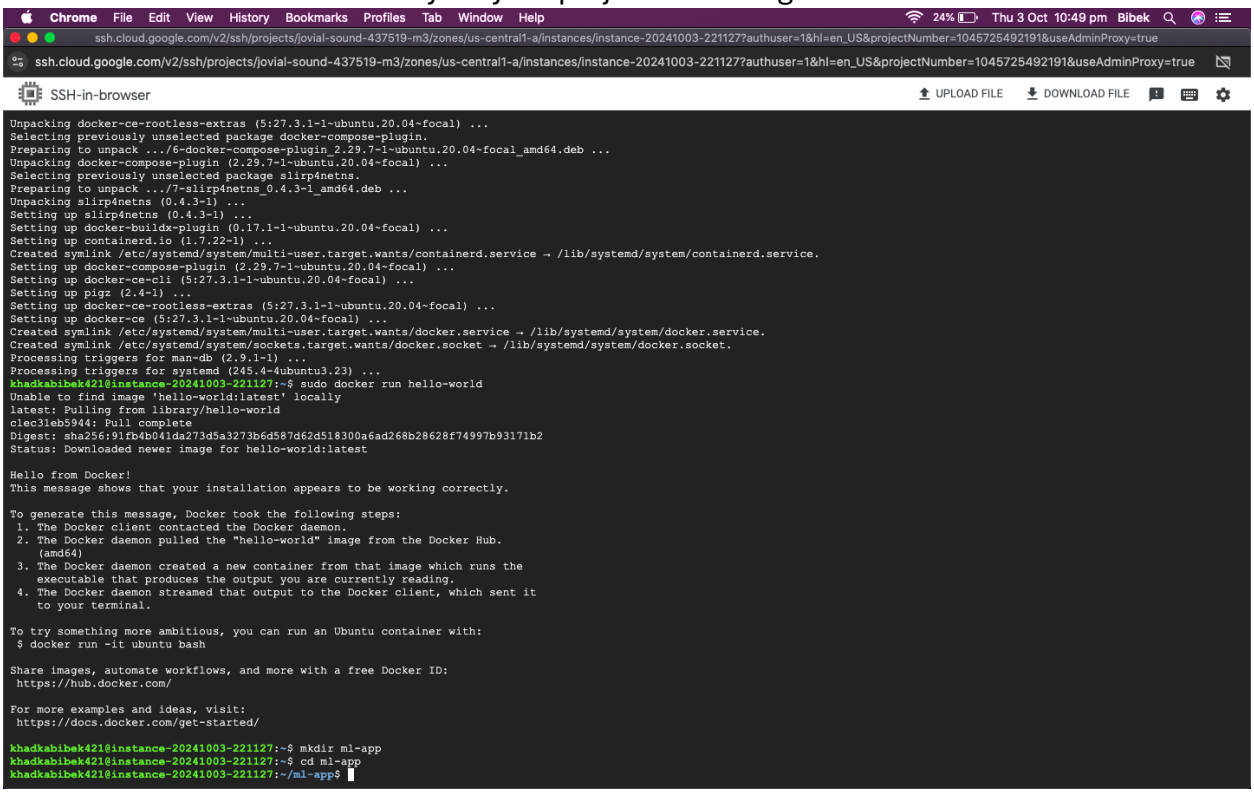
4. **Verify Docker Installation**
    - Verify that Docker is installed correctly by running:

sudo docker run hello-world



Step 3: Create a Dockerfile for the ML Application

1. **Create Project Directory**
    - Create a directory for your project and navigate into it:



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"]
```

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
```

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**)

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
```

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.