Tutorial and Lab

Week 7

Learning Objectives

By the end of this tutorial, you will be able to understand the basic steps of creating and running Docker images and containers and pushing them to the Docker Hub.

Part 1 - Docker

Please perform these steps:

1. Download the docker desktop (from this link) and install it using your administrator credentials
2. Create a personal account on the Docker Hub (from this link)
3. Check the docker version installed on your machine using `docker --version` command
4. Pull the docker images using this command: `docker pull ibrahimradwan/prediction2022`
5. Run the pulled docker using: 'docker run [provide image Id here]'
6. Create a Docker image for the project you developed in weeks 4 and 5 and share it on your GitLab repository.
7. Build the docker image
8. Push this Docker image into your profile in the Docker Hub.
9. Can you pull this Docker image from your profile and run an instance on your local machine?

[Hint: Please revisit the notes of the week's lecture for the use and explanation of the Docker commands that you may need in these exercises.]

Part2 - Using Docker with simple projects

Please perform the following steps:

1. Create a Project Directory*

First, create a directory on your local machine to hold the project files.

```
mkdir docker-python-app
```

```
cd docker-python-app
```

2. Create a Python Script

In this directory, create a simple Python script, such as app.py.

```
touch app.py
```

Open app.py in a text editor and add the following code to

```
print("Hello, Docker!")
```

3. Create a Dockerfile

Now, create a Dockerfile that Docker will use to build the image.

```
touch Dockerfile
```

Open the Dockerfile in a text editor and add the following content:

- line 1: Use an official Python runtime as a parent image

  ```
  FROM python:3.9-slim
  ```

- line 2: Set the working directory in the container

  ```
  WORKDIR /usr/src/app
  ```

- line 3: Copy the current directory contents into the container

  ```
  COPY . .
  ```

- line 4: Run the Python application

  ```
  CMD [ "python", "./app.py" ]
  ```

4. Build the Docker Image

Now that you have the Dockerfile build the Docker image. Run this command inside the project directory:

```
docker build -t python-hello-docker .
```

5. Run the Docker Container

Once the image is built, you can run it as a container.

```
docker run python-hello-docker
```

This should print:

```
Hello, Docker!
```

6. Check the Docker Images

To see the list of images on your machine, use:

```
docker images
```

Part3 - Using Docker with advanced projects

Please perform the following steps:

1. **Create a Project Directory**

   same as the previous exercise

2. **Create a Flask Application**

- Create a simple Flask web app in `app.py` in this directory.
- Open `app.py` in a text editor and add the following Python code:

  ```
  touch app.py
  ```

```
from flask import Flask, jsonify

app = Flask(__name__)

@app.route('/')
def home():
    return jsonify(message="Hello from Dockerized Flask app!")

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

3. **Create `requirements.txt` for Dependencies.**

   `touch requirements.txt`

   Then, add Flask as a dependency to this file as follows:

   `Flask==2.3.2`

4. **Create a Dockerfile**

Open the `Dockerfile` in a text editor and add the following content:

```
# Step 1: Use an official Python runtime as a parent image
FROM python:3.9-slim

# Step 2: Set the working directory in the container
WORKDIR /usr/src/app

# Step 3: Copy the requirements.txt file into the container
COPY requirements.txt ./

# Step 4: Install the required Python packages
RUN pip install --no-cache-dir -r requirements.txt

# Step 5: Copy the current directory contents into the container
COPY . .

# Step 6: Expose the application port
EXPOSE 7000

# Step 7: Define the command to run the Flask app
CMD [ "python", "./app.py" ]
```

5. **Build the Docker Image**

`docker build -t flask-docker-app .`

6. **Run the Docker Container**

   `docker run -p 7000:7000 flask-docker-app`

7. **Access the application**

   Check on the addresses /IPs that the bash provides you with and open them in your browser on the addresses /IPs that the bash provides you with and open them in your browser, to see the app running. You should see something like this:

```
{
    "message": "Hello from Dockerized Flask app!"
}
```
