Clone a Docker repository, Configure the target environment with python and Run an Automation script

- Rohith Neeraje

Content:

Create Docker repository

Create the automation Script

Run and verify the automation script

We have already created the docker image. Now, to get the docker image and container into a repository, we have to do the following steps.

Create a Docker Repository

Step 1:

Log in to your Docker account on the web browser.

Step 2:

First, find out the image name by typing in,

docker images

```
C:\Users\rohit>docker images
REPOSITORY       TAG       IMAGE ID       CREATED             SIZE
threejs-editor   latest    1d2da9017fcb   56 minutes ago      44.5MB
<none>           <none>    5999ba596bd1   About an hour ago   121MB
<none>           <none>    2a2785e899e3   About an hour ago   121MB
myapp            latest    2897313c55f1   23 hours ago        187MB
```

Step 3:

Tag the required image using the following command.

docker tag threejs-editor rohithneeraje/editor

(you can include versioning in this step: docker tag threejs-editor rohithneeraje/editor:v1.0)

Check docker images to see if the image has been created or not.

```
C:\Users\rohit>docker tag threejs-editor rohithneeraje/editor

C:\Users\rohit>docker images
REPOSITORY             TAG       IMAGE ID       CREATED             SIZE
threejs-editor         latest    1d2da9017fcb   58 minutes ago      44.5MB
rohithneeraje/editor   latest    1d2da9017fcb   58 minutes ago      44.5MB
<none>                 <none>    5999ba596bd1   About an hour ago   121MB
<none>                 <none>    2a2785e899e3   About an hour ago   121MB
myapp                  latest    2897313c55f1   23 hours ago        187MB
```

Step 4:

Log into docker on the command line with the command.

docker login

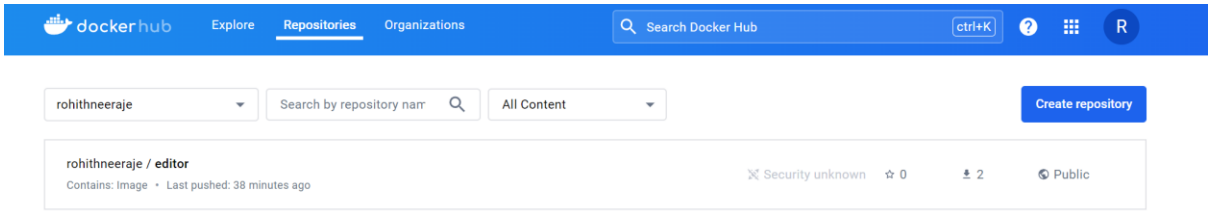
Step 5:

Push the docker image into its new repository using the following command.

docker push rohithneeraje/editor:latest

```
C:\Users\rohit>docker push rohithneeraje/editor:latest
The push refers to repository [docker.io/rohithneeraje/editor]
760ecb1a353c: Pushed
5f70bf18a086: Pushed
13c52683b537: Mounted from library/nginx
337b7d64083b: Mounted from library/nginx
cdd311f34c29: Mounted from library/nginx
3e8ad8bcb0ac: Mounted from library/nginx
74b4ff8dbbd1: Mounted from library/nginx
c018a48a857c: Mounted from library/nginx
0f73163669d4: Mounted from library/nginx
aedc3bda2944: Mounted from library/nginx
latest: digest: sha256:c04ad7f29b1022dfb0c92407cb4d24104f1679a5bc97ce7fa3105db716263194 size: 2405
```

You can also check the docker hub on the web browser to see it getting reflected in it.



We have successfully created a docker repository.

The next step is to create an automation script. For this purpose, I use python.

Creation of Automation script in Python

For this, we need to create a python script to check whether or not docker is installed, if not install it, pull the Docker image from the repo, start the docker container and verify whether or not the application is running.

Step 1:

We create the following python file in the same directory as threejs/editor, and name it as deploy_threejs.py

Step 2:

Check whether docker is installed or not:

```python
def check_docker_installed():
    """Check if Docker is installed on the target machine."""
    try:
```

```
        subprocess.run(["docker", "--version"], check=True,
stdout=subprocess.PIPE)
        print("Docker is already installed.")
    except subprocess.CalledProcessError:
        print("Docker is not installed.")
        install_docker()
```

Install docker if not installed.

```
def install_docker():
    """Install Docker on the target machine."""
    print("Installing Docker...")
    subprocess.run(["curl", "-fsSL", "https://get.docker.com", "-o", "get-
docker.sh"], check=True)
    subprocess.run(["sudo", "sh", "get-docker.sh"], check=True)
    print("Docker installed successfully.")
```

Step 3:

Pull the docker image.

```
def pull_docker_image(image_name):
    """Pull the Docker image for the three.js editor application."""
    print(f"Pulling Docker image: {image_name}")
    try:
        subprocess.run(["docker", "pull", image_name], check=True)
        print("Docker image pulled successfully.")
    except subprocess.CalledProcessError:
        print("Failed to pull Docker image. Make sure the image exists.")
```

Step 4:

Start the container.

```
def start_docker_container(image_name, container_name, port_mapping):
    """Start the Docker container with the specified configuration."""
    print("Starting Docker container...")
    try:
        subprocess.run(["docker", "run", "-d", "-p", port_mapping, "--name",
container_name, image_name], check=True)
        print("Docker container started successfully.")
    except subprocess.CalledProcessError:
        print("Failed to start Docker container.")
```

Step 5:

Verify whether the application has started or not.

```
def verify_application_accessibility(container_name, port):
    """Verify that the application is running and accessible."""
    print("Verifying application accessibility...")
```

```
    try:
        response = requests.get(f"http://localhost:{port}")
        if response.status_code == 200:
            print("Application is running and accessible.")
        else:
            print("Application is not accessible.")
    except requests.RequestException:
        print("Failed to connect to the application.")
```

Step 6:

The main of the program.

```
if __name__ == "__main__":
    # Configuration
    IMAGE_NAME = "rohithneeraje/editor:latest"
    CONTAINER_NAME = "threejs-editor"
    PORT_MAPPING = "8080:8080"

    # Check if Docker is installed
    check_docker_installed()

    # Pull Docker image
    pull_docker_image(IMAGE_NAME)

    # Start Docker container
    start_docker_container(IMAGE_NAME, CONTAINER_NAME, PORT_MAPPING)

    # Verify application accessibility
    verify_application_accessibility(CONTAINER_NAME, 8080)
```

We also need to import subprocesses and requests at the beginning of the file.

```
import subprocess
import requests
```

The file should totally look like this:

```
import subprocess
import requests

def check_docker_installed():
    """Check if Docker is installed on the target machine."""
    try:
        subprocess.run(["docker", "--version"], check=True,
stdout=subprocess.PIPE)
        print("Docker is already installed.")
    except subprocess.CalledProcessError:
```

```
        print("Docker is not installed.")
        install_docker()

def install_docker():
    """Install Docker on the target machine."""
    print("Installing Docker...")
    subprocess.run(["curl", "-fsSL", "https://get.docker.com", "-o", "get-
docker.sh"], check=True)
    subprocess.run(["sudo", "sh", "get-docker.sh"], check=True)
    print("Docker installed successfully.")

def pull_docker_image(image_name):
    """Pull the Docker image for the three.js editor application."""
    print(f"Pulling Docker image: {image_name}")
    try:
        subprocess.run(["docker", "pull", image_name], check=True)
        print("Docker image pulled successfully.")
    except subprocess.CalledProcessError:
        print("Failed to pull Docker image. Make sure the image exists.")

def start_docker_container(image_name, container_name, port_mapping):
    """Start the Docker container with the specified configuration."""
    print("Starting Docker container...")
    try:
        subprocess.run(["docker", "run", "-d", "-p", port_mapping, "--name",
container_name, image_name], check=True)
        print("Docker container started successfully.")
    except subprocess.CalledProcessError:
        print("Failed to start Docker container.")

def verify_application_accessibility(container_name, port):
    """Verify that the application is running and accessible."""
    print("Verifying application accessibility...")
    try:
        response = requests.get(f"http://localhost:{port}")
        if response.status_code == 200:
            print("Application is running and accessible.")
        else:
            print("Application is not accessible.")
    except requests.RequestException:
        print("Failed to connect to the application.")

if __name__ == "__main__":
    # Configuration
    IMAGE_NAME = "rohithneeraje/editor:latest"
    CONTAINER_NAME = "threejs-editor"
    PORT_MAPPING = "8080:8080"
```

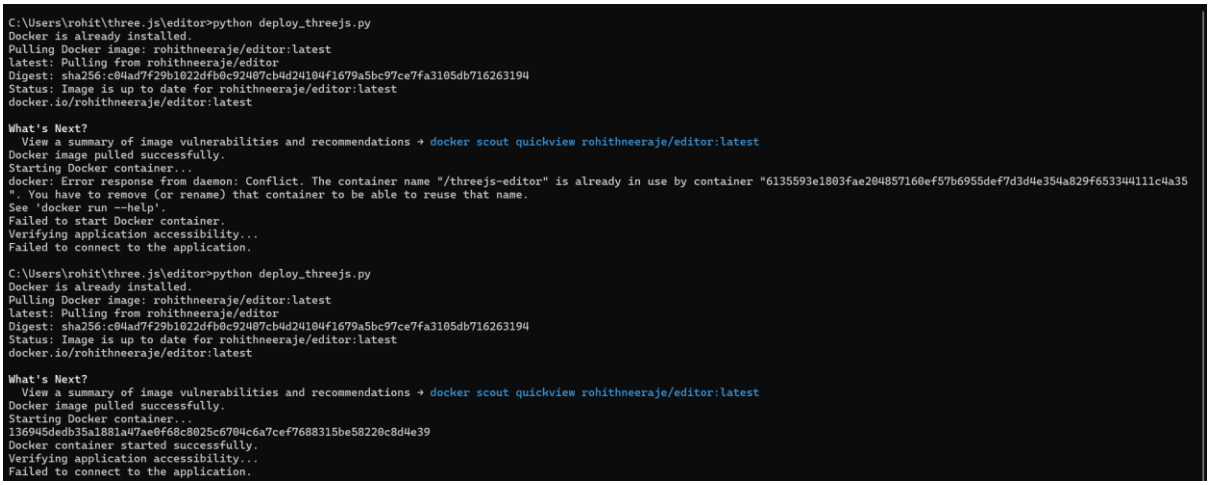
```python
# Check if Docker is installed
check_docker_installed()

# Pull Docker image
pull_docker_image(IMAGE_NAME)

# Start Docker container
start_docker_container(IMAGE_NAME, CONTAINER_NAME, PORT_MAPPING)

# Verify application accessibility
verify_application_accessibility(CONTAINER_NAME, 8080)
```

We have now created the automation script. Next we have to run it.

Running the automation script

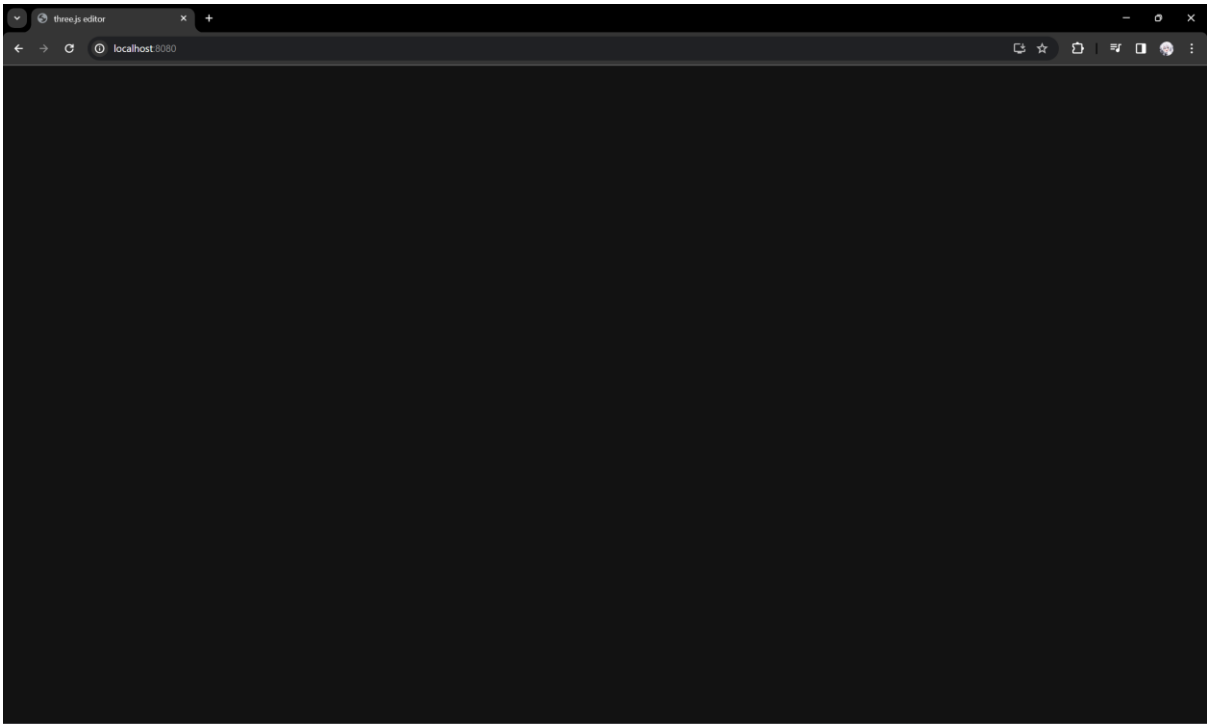
To run the script, we use the command

python deploy_threejs.py



It should look similar to this.

As we can see, it’s the same as the index.html file present in the editor directory.



Hence, we have successfully created a docker repository, Configured the target environment with an automation script, and also can verify whether the app is running or not.