Usecase on Automating Apache HTTP Server Deployment and Testing using docker

Introduction:

ShopEase, a mid-sized e-commerce company, is preparing for its annual seasonal sale. With a dramatic increase in website traffic expected, the company must ensure its web servers can scale seamlessly and operate reliably. The DevOps team is tasked with automating the deployment process for Apache HTTP servers and verifying their functionality through an efficient, repeatable pipeline. This automation is essential to support the anticipated surge and to guarantee zero downtime during the critical sales period.

Background:

In previous seasonal sales, the lack of a standardized deployment process led to inconsistencies across servers, resulting in avoidable outages and loss of revenue. To resolve this, the DevOps team decided to leverage Docker containers to create a consistent, portable server environment. Using automated testing scripts, the team aims to ensure all servers are operational before being deployed to handle live traffic.

This use case illustrates how the automation of web server deployment and testing transforms the reliability and efficiency of ShopEase's infrastructure, providing a robust solution to meet high traffic demands during peak periods. By integrating Docker, Python-based tests, and version control via GitHub, the team ensures a smooth and collaborative development process.

Objective:

To automate the deployment of an Apache HTTP Server using Docker containers and verify its functionality through Python-based test scripts.

Pre-requisites:

1. **Docker** installed on your machine.
2. **Python 3.x** installed along with docker and requests libraries.
3. **Git** installed and configured.
4. Access to a GitHub repository for version control.

Experiment Steps:

Set Up Docker

1. **Install Docker:**
    - **Windows:**
        - Download Docker Desktop from the Docker website.
        - Install Docker Desktop and enable WSL2 integration.
        - Verify installation with:

```
docker --version
```

- **Linux:**
- sudo apt-get update
- sudo apt-get install docker.io
- sudo systemctl start docker
- sudo systemctl enable docker

2. **Pull Apache HTTPD Image:**

docker pull httpd

3. **Run a Docker Container:**
4. docker run -d --name httpd-container -p 8080:80 httpd

   docker ps is used to check the container is running in the environment

   - Verify the container by visiting http://localhost:8080.

Part 2: Customize and Test the Deployment

1. **Create a Custom Docker Image:**
   - Create a file named Dockerfile with the following content:
   - FROM httpdCOPY ./index.html /usr/local/apache2/htdocs/
   - Place a custom index.html file in the same directory.
   - Build the Docker image:docker build -t custom-apache-image .
   - Run the new container: docker run -d --name custom-httpd -p 8080:80 custom-apache-image

---

Expected Outcome:

1. Successful deployment of an Apache HTTP Server in a Docker container.
2. Verification of server functionality through automated tests.
3. Code and results are version-controlled and shared on GitHub.

Task List to be completed

1. Check if docker is installed in the virtual machine ?
2. Create docker images in the virtual environment ?
3. Create webserver using apache server and create a sample website
4. Check to-do website host it in the webserver the website must be written in HTML /CSS JAVA SCRIPT

5. Check whether the application is hosted in the correct ports
6. Launch the application
7. Run the testcase to check if the deployment in successful

Run the code and submit the code

To Run the code using the exe file inside the folder .

1. Run test.exe to execute the testcase
2. Run the push.exe to save the created file in the repository