Deploying HTML File with Docker

Training Material

Table of Contents

Chapter 1. Prerequisites

Software Requirements

- **Jenkins**: Installed and running on your server or local machine.
- **Docker**: Installed on the same machine where Jenkins is running.
- **Git**: Installed if you are using Git for version control.

Setting Up Jenkins

1. Install Jenkins by following the official installation guide.
2. Access Jenkins via http://localhost:8080 (or your server's address).
3. Complete the setup wizard and install the recommended plugins.

Setting Up Docker

1. Install Docker by following the official installation guide.
2. Ensure Docker is running and you can run Docker commands from the terminal.

Chapter 2. Creating the HTML Project on GitHub

Example HTML Project Structure

Create a simple HTML project directory on your local machine and push it to GitHub. The structure might look like this:

css
Copy code
html-project/
├── index.html
└── Dockerfile

Creating a Dockerfile

In the html-project directory, create a Dockerfile with the following content:

dockerfile
Copy code
Use a lightweight web server
FROM nginx:alpine

Copy the HTML file into the container
COPY index.html /usr/share/nginx/html/index.html

Expose port 80
EXPOSE 80

Push to GitHub

Initialize a Git repository in your project directory:

git init
Add your files:
git add .

Commit the changes:

git commit -m "Initial commit"

1. Create a new repository on GitHub.
2. Link your local repository to GitHub and push the changes:

git remote add origin https://github.com/your-username/html-project.git
git push -u origin main

Chapter 3. Configuring Jenkins

Installing Necessary Plugins

1. Go to the Jenkins Dashboard.
2. Navigate to **Manage Jenkins** > **Manage Plugins**.
3. Search for and install the following plugins:

    - **Pipeline**
    - **Git** (if it's not already installed)
    - **Docker Pipeline** (if you plan to use Docker commands in your pipeline)

Creating a New Pipeline Job

1. In the Jenkins dashboard, click on **New Item**.
2. Enter a name for your pipeline (e.g., DeployHTMLToDocker).
3. Select **Pipeline** and click **OK**.

Chapter 4. Writing the Jenkins Pipeline Script

Sample Pipeline Script

In the configuration page of your new pipeline job, scroll down to the **Pipeline** section. Here's a sample Jenkinsfile you can use:

Groovy code:

```
pipeline {
    agent any

    environment {
        DOCKER_IMAGE = 'html-nginx:latest'
    }
```

```
    stages {
        stage('Checkout Code') {
            steps {
                // Clone the GitHub repository
                git 'https://github.com/your-username/html-project.git'
            }
        }

        stage('Build Docker Image') {
            steps {
                script {
                    // Build the Docker image
                    sh 'docker build -t $DOCKER_IMAGE .'
                }
            }
        }

        stage('Run Docker Container') {
            steps {
                script {
                    // Stop and remove any existing container
                    sh 'docker rm -f html-nginx || true'
                    // Run the new container
                    sh 'docker run -d -p 80:80 --name html-nginx $DOCKER_IMAGE'
                }
            }
        }
    }

    post {
        always {
            // Optionally clean up Docker images
            sh 'docker rmi $DOCKER_IMAGE || true'
        }
    }
}
```

Steps Breakdown

1. **Checkout Code**: Clones the GitHub repository containing the HTML file.
2. **Build Docker Image**: Builds the Docker image using the Dockerfile.
3. **Run Docker Container**: Stops any existing container with the same name and starts a new one.

Chapter 5. Running the Pipeline

Triggering the Build

1. Once you have saved the pipeline configuration, click on **Build Now**.

2. Monitor the build process in the **Build History**.

Accessing the Deployed Application

- After a successful build, you can access the HTML application in your browser at http://localhost or http://your-server-ip.

Chapter 6:Deploy a html file with Docker Desktop:

To deploy an HTML file in Docker Desktop for Windows, you can use Docker to package the HTML and its dependencies into a standardized unit:

- Create a directory for the HTML files
- Create an HTML page, such as index.html
- Create a directory named Dockerfile
- Open the Dockerfile in a text editor and add the following code:
- FROM nginx:alpine
- Build the Docker image for the HTML server by running the command docker build -t html-server-image:v1
- Run the Docker container
- Test the port by browsing to http://localhost:8080

Example of my html file named “index.html”. I placed in my local storage path “F:\Devops\karnataka devops course\devops project\index.html”.

Here’s a step-by-step guide to deploy your HTML file located at F:\Devops\karnataka devops course\devops project\index.html using Docker Desktop on Windows.

Step 1: Prepare Your Project Directory

1. Create a Project Directory:
2. Open PowerShell or Command Prompt.
3. Navigate to a desired location where you want to create a new project directory:

mkdir F:\Devops\karnataka-devops-course\devops-project-dockercd F:\Devops\karnataka-devops-course\devops-project-docker

4. Copy Your HTML File
5. Copy your existing index.html file into the new project directory. You can do this via File Explorer or use the command line:

copy "F:\Devops\karnataka devops course\devops project\index.html" .

Step 2: Create the Dockerfile

1. Create a Dockerfile:
2. In the same directory, create a file named Dockerfile (no file extension). You can use a text editor or run:

echo > Dockerfile

3. Open the Dockerfile with a text editor (like Notepad or VSCode) and add the following content: in Dockerfile

```
# Use the official Nginx image as a base
FROM nginx:alpine

# Copy the HTML file to the Nginx web server directory
COPY index.html /usr/share/nginx/html/

# Expose port 80
EXPOSE 80
```

Step 3: Build the Docker Image

1. Open Command Prompt or PowerShell: Make sure you are still in your project directory (F:\Devops\karnataka-devops-course\devops-project-docker).
2. Build the Image: Run the following command to build your Docker image:

```
docker build -t my-html-app .
```

This command will create an image named my-html-app based on your Dockerfile.

Step 4: Run the Container

1. Run the Container: Use the following command to create and start a container from your image:

```
docker run -d -p 8080:80 --name my-html-container my-html-app
```

The -d flag runs the container in detached mode, and -p 8080:80 maps port 8080 on your host to port 80 in the container.

Step 5: Access Your HTML Page

1. Open a Web Browser:
2. Open your web browser and navigate to:

```
http://localhost:8080
```

You should see your HTML page served by Nginx.

Step 6: Manage the Container

1. List Running Containers:

```
docker ps
```

2. Stop the Container:

```
docker stop my-html-container
```

3. Remove the Container:

```
docker rm my-html-container
```

4. Remove the Image (if needed):

```
rmi my-html-app
```

Summary

1. Create a project directory.
2. Copy your index.html file into that directory.
3. Create a Dockerfile to use Nginx for serving the HTML file.
4. Build the Docker image with docker build.
5. Run a container from the image using docker run.
6. Access your HTML page via a web browser.

Chapter 7. Conclusion

You have successfully set up Jenkins to automate the deployment of an HTML file from a GitHub repository to a Docker container. This process enables quick updates and deployments of your web application with each commit to your repository.

Further Considerations

- **Environment Variables**: Consider using environment variables for sensitive information.
- **Testing**: Implement automated tests to ensure your HTML behaves as expected before deployment.
- **Cleanup**: Regularly clean up unused Docker images and containers to free up disk space.