Devops Assessment

Submission

Create a GitHub repository and document the step-by-step process of se ng up the project, including images and reports for Docker and Jenkins, in the README file.

Submission Date: 16th July, 2024

1. Automated Tes ng with Jenkins and Docker

Objec ve: U lize Jenkins to automate the tes ng process of a Python Django-based web applica on within a Docker container.

Steps:

- Develop a web applica on with some test cases (e.g., using a tes ng framework like pytest for Python).
- Create a Docker file that includes the applica on and the necessary tes ng tools.
- Set up a Jenkins pipeline to:
- Pull the source code.
- Build the Docker image.
- Run the tests inside the Docker container.
- Report the test results.

ANSWER

Steps

1. Develop a Django Web Application with Test Cases

1. **Set up a Django Project**
    - **Create a new Django project and application.**
    - **Ensure the project structure includes the necessary files (manage.py, settings.py, etc.).**

```
django-admin startproject myproject
cd myproject
django-admin startapp myApp
```

2.Create Views and URLs

- Implement a simple view for printing a message on the screen.

Views.py

from django.shortcuts import render

```
# Create your views here.

def home(request):

    context={}

    return render(request,"myApp/home.html",context)
```

Home.html

```
Hello world this is vishwa here!!
```

myApp/urls.py

```
from . import views

from django.urls import path

urlpatterns = [

  path("",views.home,name="home"),

]
```

Myproject/urls.py

```
from django.contrib import admin
from django.urls import path,include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('',include("myApp.urls")),
]
```

3.Add Test Cases

myApp/tests.py

```
from django.test import TestCase
from django.urls import reverse
class HomePageTests(TestCase):
    def test_home_page_status_code(self):
        response = self.client.get(reverse('home'))
        self.assertEqual(response.status_code, 200)

    def test_home_page_template(self):
        response = self.client.get(reverse('home'))
        self.assertTemplateUsed(response, 'myApp/home.html')
```

pytest.ini

```
[pytest]
DJANGO_SETTINGS_MODULE = myproject.settings
python_files = tests.py test_*.py *_tests.py
testpaths = myApp
```

4. Create a Dockerfile and requirements.txt

1. **Write the Dockerfile**

Dockerfile

```
FROM python:3.9

# Set the working directory
WORKDIR /usr/src/app

# Copy the requirements file into the container
COPY requirements.txt ./

# Install any dependencies
RUN pip install --no-cache-dir -r requirements.txt

# Copy the current directory contents into the container
COPY . .

# Copy the pytest configuration file
COPY pytest.ini ./

# Run the tests
CMD ["pytest"]
```

Requirements.txt

```
Django==3.2.4
pytest==6.2.4
pytest-django==4.4.0
colorama==0.4.4  # Add this if tests need it
```

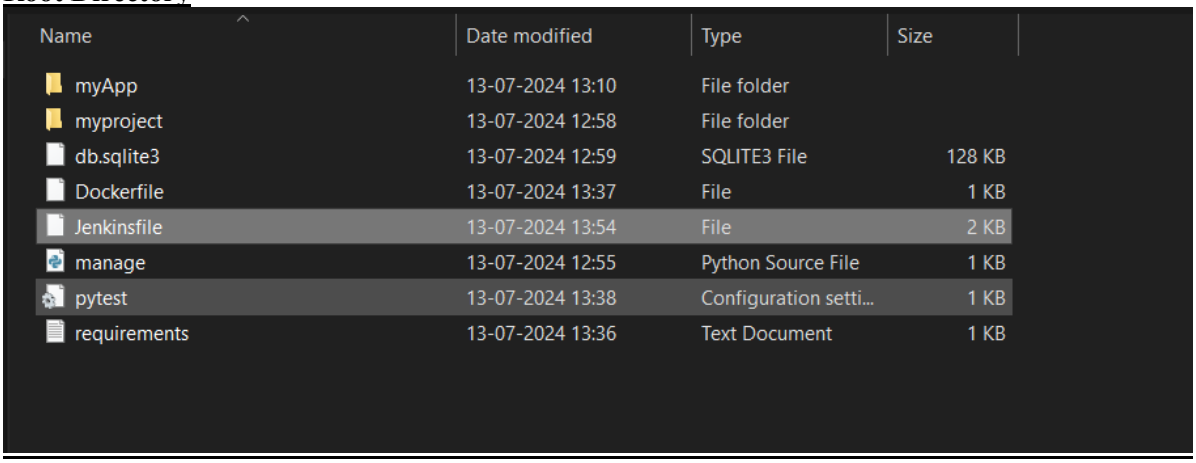
5. Create a Jenkinsfile

```
pipeline {
    agent any

    environment {
        DOCKER_IMAGE = 'my-django-app'
    }

    stages {
        stage('Checkout') {
            steps {
                // Checkout the source code from the Git repository
                git url: 'https://github.com/vishwa21pratap/my-django-project.git', branch: 'main'
            }
        }

        stage('Build Docker Image') {
            steps {
                script {
                    // Build the Docker image
                    docker.build(DOCKER_IMAGE)
                }
```

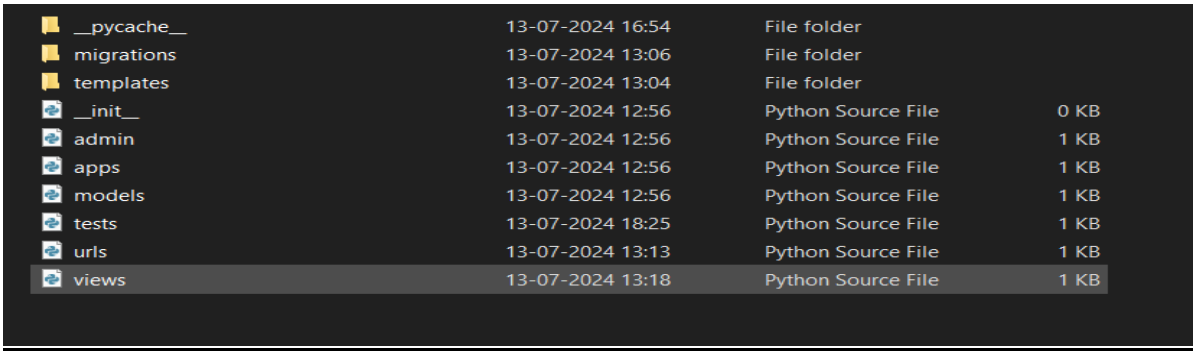
```
            }
        }

        stage('Run Tests') {
            steps {
                script {
                    // Run the tests inside the Docker container
                    docker.image(DOCKER_IMAGE).inside {
                        sh 'pytest'
                    }
                }
            }
        }

        stage('Archive Results') {
            steps {
                // Archive test results
                junit 'test-reports/*.xml'
            }
        }
    }

    post {
        always {
            // Clean up workspace
            cleanWs()
        }
    }
}
```

6. Folders Structure :

- **Root Directory**



- **myApp**



7. Create a GitHub Repository

- Create a new repository on GitHub named my-django-project.
- Clone the repository to your local machine.
- Add the above files to the github repository

8. Install Docker Desktop and navigate to the root directory of your folder.

- **Build Docker Image**

```
C:\Users\VISHWA\Desktop\folde\myproject>docker build -t my-django-app .
[+] Building 2.6s (11/11) FINISHED                                                    docker:desktop-lin
 => [internal] load build definition from Dockerfile                                                  0.
 => => transferring dockerfile: 486B                                                                  0.
 => [internal] load metadata for docker.io/library/python:3.9                                         2.
 => [internal] load .dockerignore                                                                     0.
 => => transferring context: 2B                                                                       0.
 => [1/6] FROM docker.io/library/python:3.9@sha256:47d6f16aa0de11f2748c73e7af8d40eaf44146c6dc059b1d0aa1f917f8c5cc  0.
 => [internal] load build context                                                                     0.
 => => transferring context: 8.17kB                                                                   0.
 => CACHED [2/6] WORKDIR /usr/src/app                                                                 0.
 => CACHED [3/6] COPY requirements.txt ./                                                             0.
 => CACHED [4/6] RUN pip install --no-cache-dir -r requirements.txt                                   0.
 => CACHED [5/6] COPY . .                                                                             0.
 => CACHED [6/6] COPY pytest.ini ./                                                                   0.
 => exporting to image                                                                                0.
 => => exporting layers                                                                               0.
 => => writing image sha256:3fc004aed868423097c721dff87c44df0af894a8bf46e3e239cbf8909d9e73a0          0.
 => => naming to docker.io/library/my-django-app                                                      0.

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/rrf6fn6ao3sj262xx4wmnzfsf

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
```

- **Run the Docker Image**

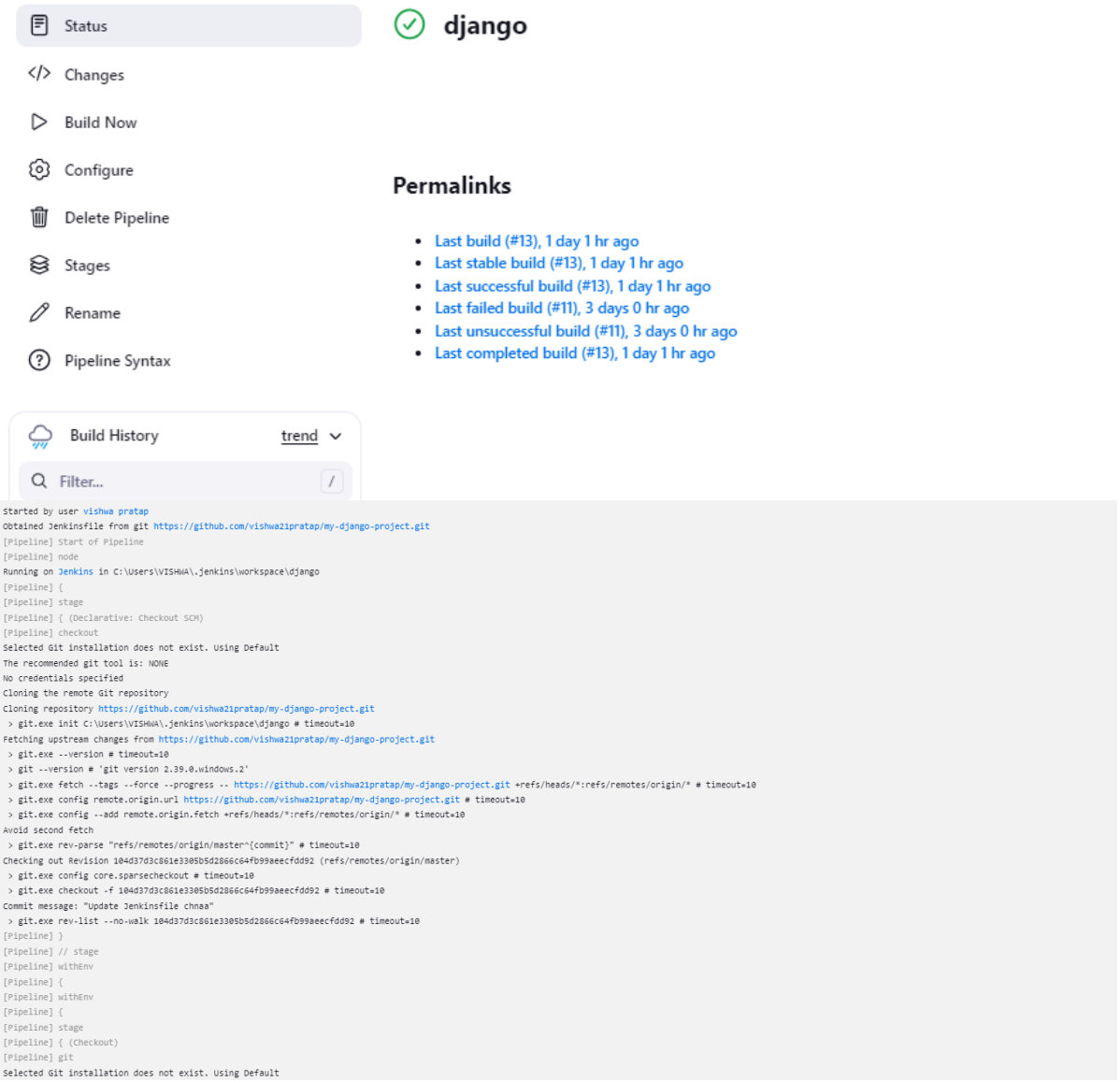
```
C:\Users\VISHWA\Desktop\folde\myproject>docker run -p 8000:8000 my-django-app
============================= test session starts ==============================
platform linux -- Python 3.9.19, pytest-6.2.4, py-1.11.0, pluggy-0.13.1
django: settings: myproject.settings (from ini)
rootdir: /usr/src/app, configfile: pytest.ini, testpaths: myApp
plugins: django-4.4.0
collected 2 items

myApp/tests.py ..                                                        [100%]

============================== 2 passed in 0.65s ===============================

C:\Users\VISHWA\Desktop\folde\myproject>
```

9. Setting up Jenkins

1. Install Jenkins
    - Download and install Jenkins.jar file from here.
    - Start Jenkins and complete the setup wizard.
2. Install Necessary Plugins
    - Docker Pipeline Plugin
    - Git Plugin

Running the Pipeline

1. Create a New Jenkins Pipeline Job
    - Go to Jenkins Dashboard.
    - Click on New Item > Pipeline.
    - Configure the pipeline to use the Jenkinsfile from your GitHub repository.
2. Run the Pipeline
    - Click Build Now to run the pipeline.

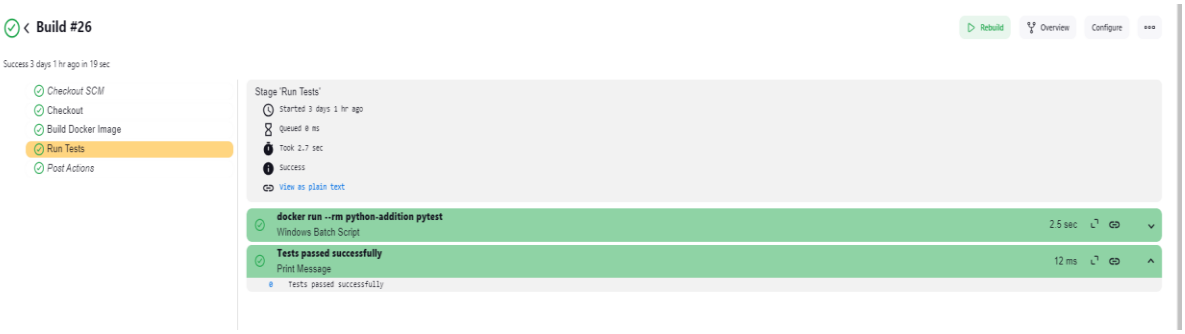


```
[Pipeline] { (Checkout)
[Pipeline] git
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
 > git.exe rev-parse --resolve-git-dir C:\Users\VISHWA\.jenkins\workspace\django\.git # timeout=10
Fetching changes from the remote Git repository
 > git.exe config remote.origin.url https://github.com/vishwa21pratap/my-django-project.git # timeout=10
Fetching upstream changes from https://github.com/vishwa21pratap/my-django-project.git
 > git.exe --version # timeout=10
 > git --version # 'git version 2.39.0.windows.2'
 > git.exe fetch --tags --force --progress -- https://github.com/vishwa21pratap/my-django-project.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision 104d37d3c861e3305b5d2866c64fb99aeecfdd92 (refs/remotes/origin/master)
 > git.exe config core.sparsecheckout # timeout=10
 > git.exe checkout -f 104d37d3c861e3305b5d2866c64fb99aeecfdd92 # timeout=10
 > git.exe branch -a -v --no-abbrev # timeout=10
 > git.exe checkout -b master 104d37d3c861e3305b5d2866c64fb99aeecfdd92 # timeout=10
Commit message: "Update Jenkinsfile chnaa"
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Build Docker Image)
[Pipeline] script
[Pipeline] {
[Pipeline] isUnix
[Pipeline] withEnv
[Pipeline] {
[Pipeline] bat

C:\Users\VISHWA\.jenkins\workspace\django>docker build -t "my-django-app" -f Dockerfile .
#0 building with "desktop-linux" instance using docker driver

#1 [internal] load build definition from Dockerfile
#1 transferring dockerfile: 622B 0.0s done
#1 DONE 0.1s

#2 [internal] load metadata for docker.io/library/python:3.8-slim
#2 ...

#3 [auth] library/python:pull token for registry-1.docker.io
#3 DONE 0.0s

#2 [internal] load metadata for docker.io/library/python:3.8-slim
```

```
#10 1.350   Apply all migrations: admin, auth, contenttypes, sessions
#10 1.393 Running migrations:
#10 1.393   No migrations to apply.
#10 DONE 1.7s

#11 [6/6] RUN python manage.py collectstatic --noinput
#11 1.372
#11 1.372 128 static files copied to '/usr/src/app/static'.
#11 DONE 1.6s

#12 exporting to image
#12 exporting layers
#12 exporting layers 2.2s done
#12 writing image sha256:49f4d53e716db8226fcdb2ffb9fd1266fd90eb47d0d6bd6508000607324ab320e done
#12 naming to docker.io/library/my-django-app done
#12 DONE 2.2s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/8ykqtjl59y11a9dwp7xlswfma
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] cleanWs
[WS-CLEANUP] Deleting project workspace...
[WS-CLEANUP] Deferred wipeout is used...
[WS-CLEANUP] done
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Reporting Test Results

- Check the Console Output
  - Verify that each stage of the pipeline runs successfully.
- View Test Results
  - Go to the Test Results section in Jenkins to view the detailed report.


Build #26
Rebuild
Overview
Configure
Success 3 days 1 hr ago in 19 sec
Checkout SCM
Checkout
Build Docker Image
Run Tests
Post Actions
Stage 'Run Tests'
Started 3 days 1 hr ago
Queued 0 ms
Took 2.7 sec
Success
View as plain text
docker run --rm python-addition pytest
Windows Batch Script
2.5 sec
Tests passed successfully
Print Message
12 ms
Tests passed successfully