Directory Structure Validation for Jenkins automation L1

In any software development or project environment, maintaining a standardized directory structure is crucial for efficient organization, easy access to files, and smooth collaboration. A well-defined directory structure helps in reducing configuration errors, streamlining workflows, and ensuring that necessary resources are always readily available.

You are responsible for setting up a new project environment where a specific directory structure must be maintained. The project resides in the path C:\.....i\Desktop\test, and the expected directory layout includes:

Under the test folder in the desktop you will be provide with these folders

- logs
- config
- data
- data\output

The directory structure needs to be validated to ensure the setup complies with the project's organization standards.

Requirements

Initiating Directory Validation:

- A Jenkins pipeline is configured to run the directory validation.
- The pipeline reads the EXPECTED_STRUCTURE variable, which defines the necessary subdirectories required in the project.
- The **Target Directory** is specified as C: .....\ Desktop\test, where the validation will be performed.

Scanning the Target Directory:

- The pipeline uses a script to scan the C:\..... \Desktop\test directory.
- It retrieves all subdirectories within the target directory using the dir /s /b /ad command.

Comparing Expected vs. Actual Structure:

- The pipeline compares the expected directory structure (provided in the EXPECTED_STRUCTURE variable) with the actual structure in the target directory.
- Any discrepancies (missing or extra directories) are highlighted and reported.

Validation Report Generation:

- The pipeline generates a validation report detailing any missing directories or extra directories present in the project setup.

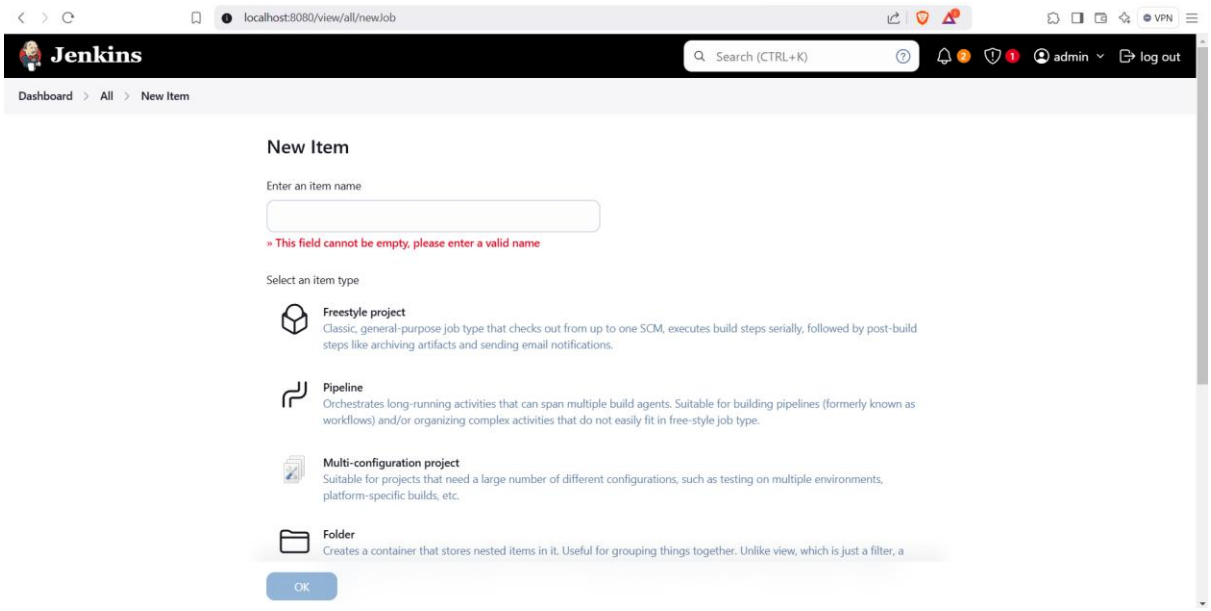
Post-validation Actions:

- If validation passes, the pipeline outputs: Directory validation completed!

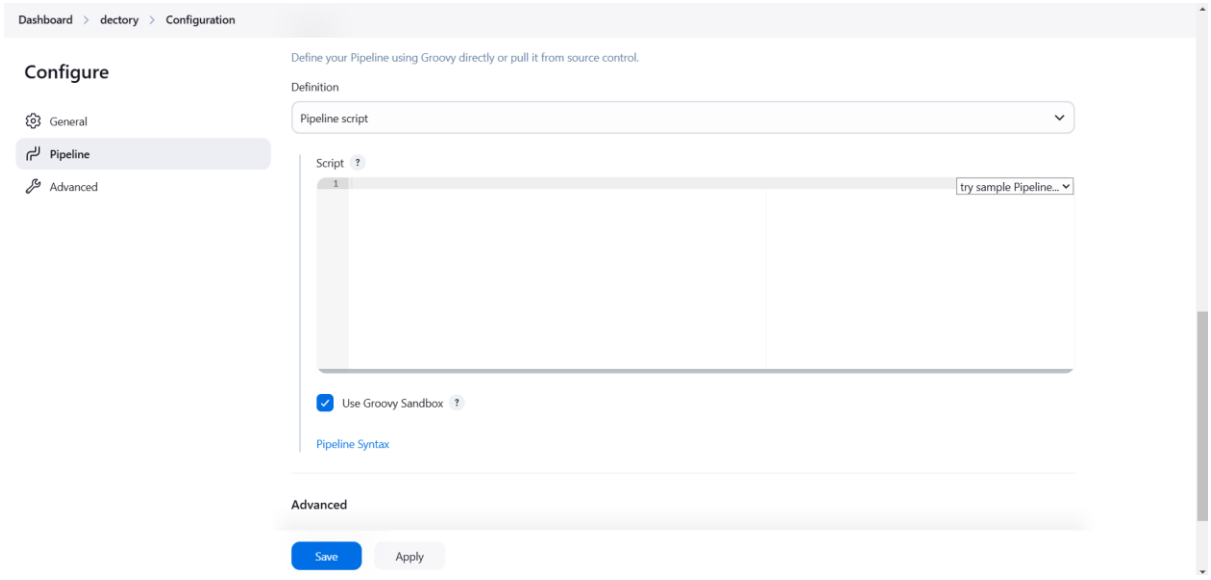
- If validation fails, the pipeline provides a detailed report of issues but does not stop the build, ensuring project setup can proceed with minimal interruption.

As Devops engineer you are suppose complete the pipeline code

- Log in to the devops environment
- Use the usename and password provide in the desktop in the environment



- Create the new pipeline give the New item a name.and select the pipeline option and click ok
- The next page you can see the pipeline script



- Your are suppose to read out the requirement from the document and create the pipeline script .
- Click on save
- Start the build check if the build is successful
- After the build is successful you are suppose use the use the python script to run the test of the activities done .
- To run the test case you can follow the below steps

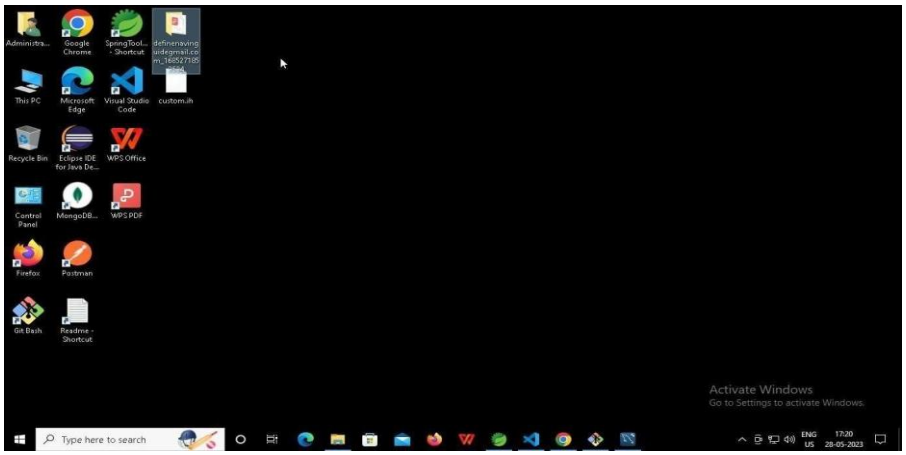
To run and push the code to git

- To run the testcase
- Use the command pytest filename.py
- Screenshot to push the application to github

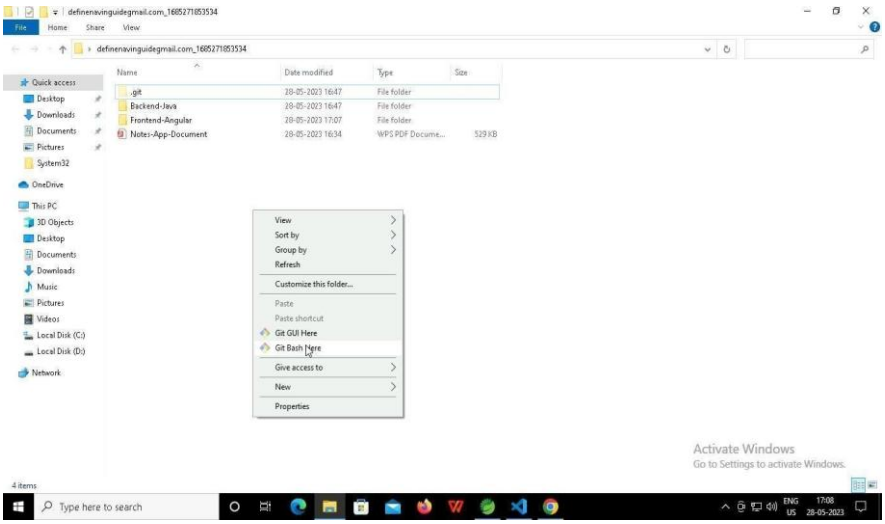
-----x-----

You can run test cases as many numbers of times and at any stage of Development, to check howmany test cases are passed/failed and accordingly refactor your code.

1. Make sure before final submission you commit all changes to git.
   For that open theproject folder available on desktop

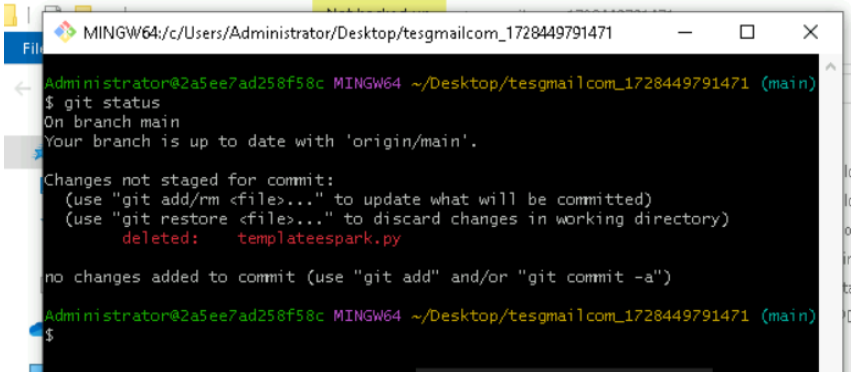


   a. Right click in folder and open Git Bash



b. In Git bash terminal, run following commands

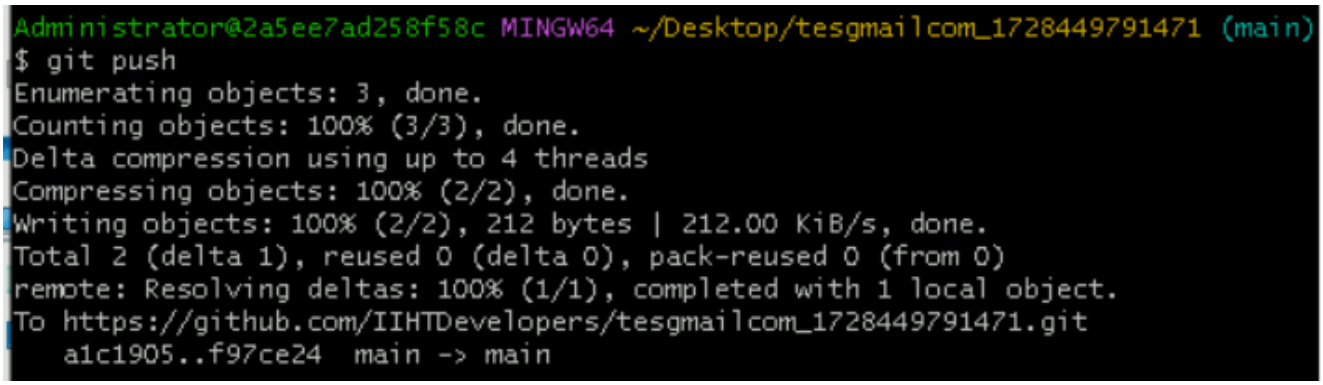
c. git status



d. git add .

e. git commit -m “First commit”
(You can provide any message every time you commit)

f. git push


Administrator@2a5ee7ad258f58c MINGW64 ~/Desktop/tesgmailcom_1728449791471 (main)
$ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 212 bytes | 212.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/IIHTDevelopers/tesgmailcom_1728449791471.git
a1c1905..f97ce24 main -> main