### **Creation of Pipeline in Jenkins:**

#### **Step 1: Create Project Directory and Navigate Into It**

* mkdir my\_python\_project && cd my\_python\_project
  + Creates a new directory named my\_python\_project.
  + Changes the working directory to my\_python\_project.

#### **Step 2: Initialize a Git Repository**

* git init
  + Initializes a new Git repository in the project directory.

#### **Step 3: Create pyproject.toml for Project Configuration**

* cat << EOF > pyproject.toml ... EOF
  + Creates a pyproject.toml file with project configurations.
  + Specifies the build system (hatchling) and the project's metadata.
  + Defines a CLI script (myapp) that points to my\_python\_project.main:main.

#### **Step 4: Create Source Code Directory and Main Python File**

* mkdir -p src/my\_python\_project
  + Creates a directory structure (src/my\_python\_project).
* cat << EOF > src/my\_python\_project/main.py ... EOF
  + Creates main.py with a simple function that prints a message.
  + **Mistake:** \_name\_ should be corrected to \_\_name\_\_, and \_main\_ to \_\_main\_\_.

#### **Step 5: Create a Basic Test File**

* mkdir tests
  + Creates a tests directory for unit tests.
* cat << EOF > tests/test\_main.py ... EOF
  + Creates a test file using unittest framework.
  + Captures output from main() and compares it with expected output.
  + **Mistake:** \_name\_ should be \_\_name\_\_, \_main\_ should be \_\_main\_\_.
  + sys.\_stdout\_ should be corrected to sys.stdout.

#### **Step 6: Create a Dockerfile**

* cat << EOF > Dockerfile ... EOF
  + Defines a Docker image using python:3.9-slim as the base image.
  + Copies the built .whl file into the container.
  + Installs the wheel and sets myapp as the default command.

#### **Step 7: Create a .gitignore File**

* cat << EOF > .gitignore ... EOF
  + Specifies files and directories to ignore in Git.
  + Ignores \_\_pycache\_\_, compiled Python files, dist/, build/, .egg-info/, and venv/.

#### **Step 8: Build the Initial Wheel File**

* pip install build
  + Installs the build package to create Python wheels.
* python -m build --wheel
  + Builds a wheel (.whl) package of the project.

#### **Step 9: Commit Everything to Git**

* git add .
  + Stages all new and modified files for commit.
* git commit -m "Initial project setup with Python code and Dockerfile"
  + Commits the changes with a descriptive message.

#### **Step 10: Create a Jenkinsfile for CI/CD Pipeline**

* cat << EOF > Jenkinsfile ... EOF
  1. Defines a Jenkins pipeline for automated build, test, and deployment.
* **Stages in the Jenkinsfile:**
  1. **Checkout:** Clones the repository from a local Git URL.
  2. **Build Wheel:** Installs build package and generates a .whl file.
  3. **Test:** Installs pytest and runs tests.
  4. **Build Docker Image:** Builds a Docker image for the application.
  5. **Deploy:** Stops and removes any existing container, then runs the new container.
* Uses post block to handle success or failure messages.

#### **Step 11: Add Jenkinsfile to Git and Commit**

* git add Jenkinsfile
  + Stages the Jenkinsfile.
* git commit -m "Add Jenkinsfile for CI/CD pipeline"
  + Commits the Jenkinsfile with a message.

#### **Step 12: Display Project Structure**

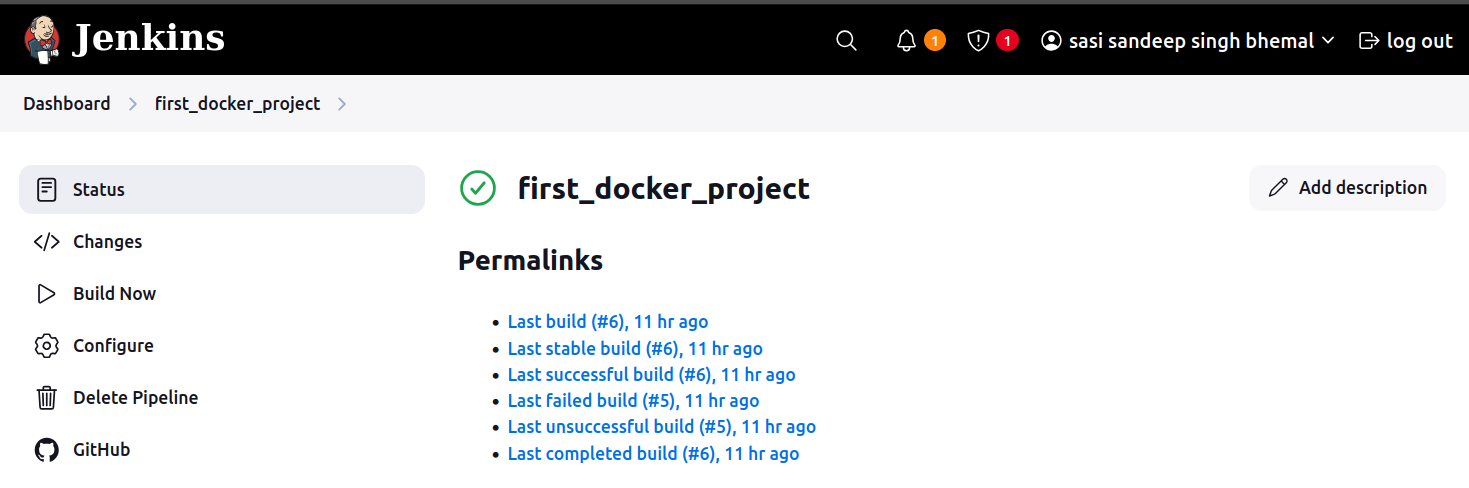
* echo "Project structure created:"
  + Prints a message.
* ls -R
  + Lists all files and directories recursively.

### **Additional Jenkins Pipeline for Repository Cloning**

* **Stage 1: Clone Repository**
  + cleanWs()
    - Cleans the Jenkins workspace before starting.
  + sh 'echo "Current directory: $PWD"'
    - Prints the current working directory.
  + git clone https://github.com/jineshranawatcode/my\_python\_project.git
    - Clones the repository from GitHub.
  + ls -la my\_python\_project
    - Lists all files inside the cloned directory.
* **Stage 2: Verify Clone**
  + cd my\_python\_project
    - Changes directory to the cloned repository.
  + ls -la
    - Lists files inside the repository.
  + git status
    - Checks the current Git status.
* **Post Actions:**
  + success → Prints **"Repository cloned successfully to workspace!"**.
  + failure → Prints **"Failed to clone repository."**.

### 

### **OUTPUT:**



### **Key Takeaways:**

* **Project Initialization:** Set up Python project structure, pyproject.toml, and Dockerfile.
* **Testing:** Used unittest and pytest for test automation.
* **Git & GitHub:** Version control with git init, add, commit, and clone.
* **CI/CD with Jenkins:** Automated build, test, and deployment using Jenkinsfile.
* **Docker:** Containerized the application with a Dockerfile.
* **Jenkins Pipeline for Git Cloning:** Ensured correct workspace setup for CI/CD.