JENKINS ASSIGNMENT

1.Build and Deploy a Simple App using Jenkins

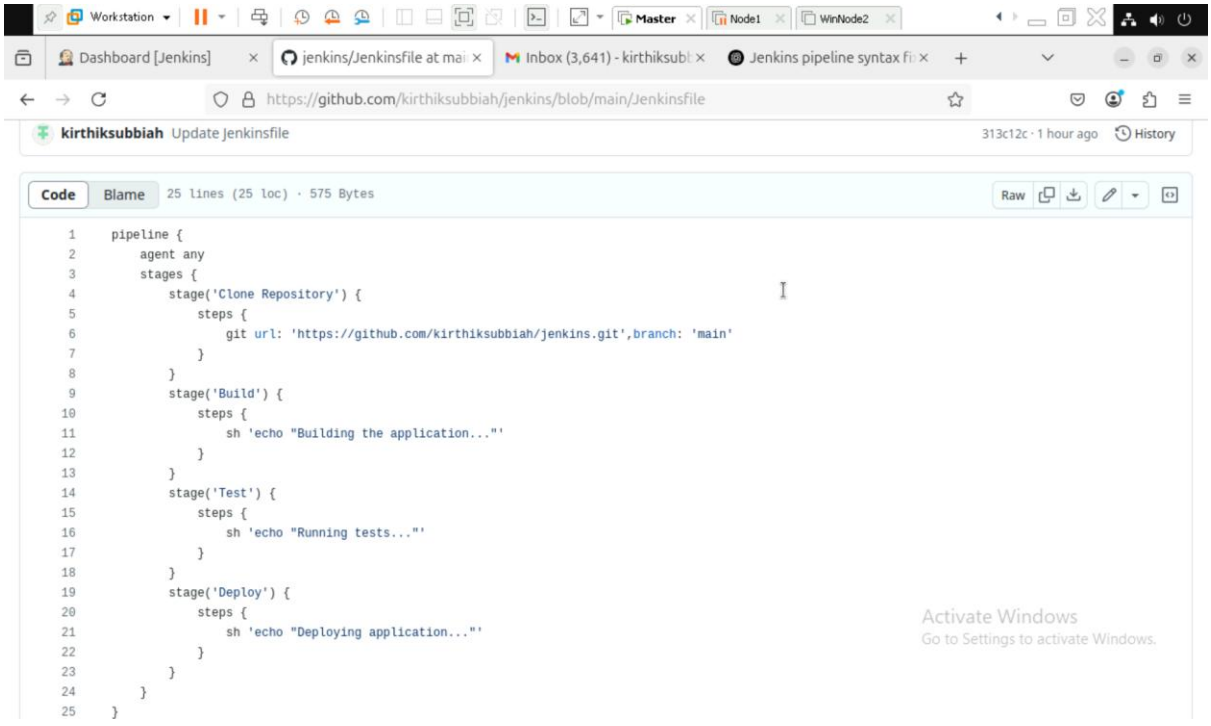
Step 1: Install Required Plugins

1. Open **Jenkins Dashboard → Manage Jenkins → Manage Plugins**.
2. Install the following plugins:

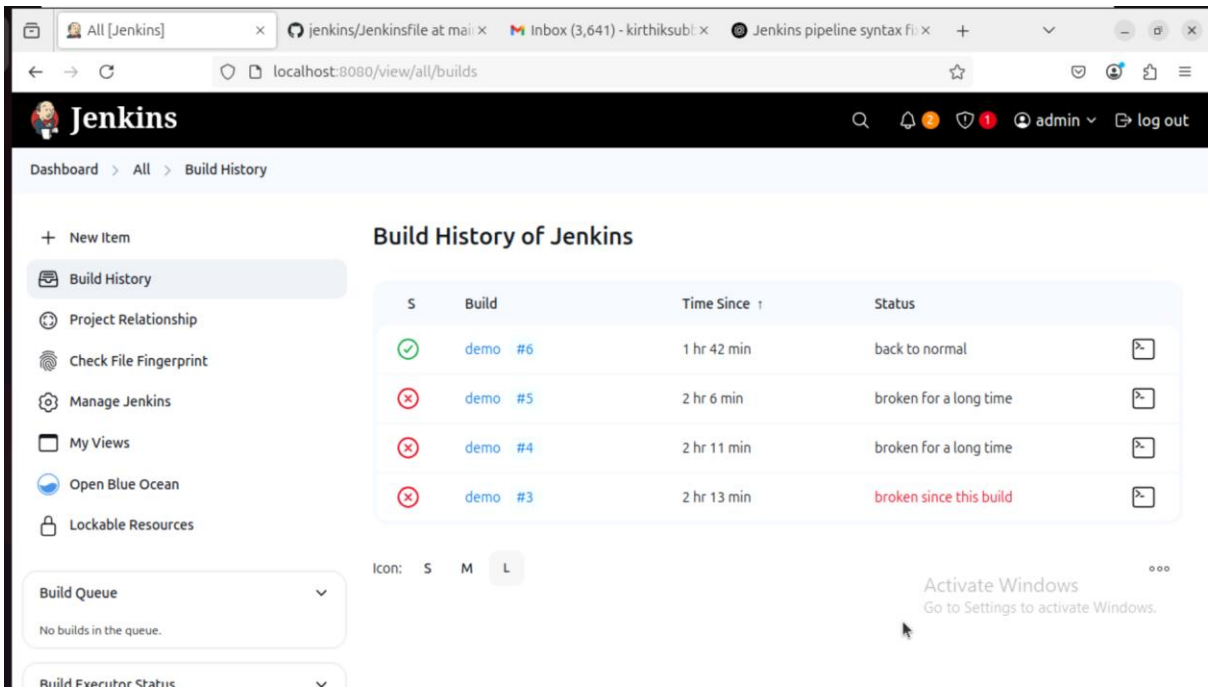
- **Pipeline**
- **Git Plugin**
- **Docker Pipeline Plugin** (if using Docker)

Step 2: Create a New Jenkins Pipeline Job

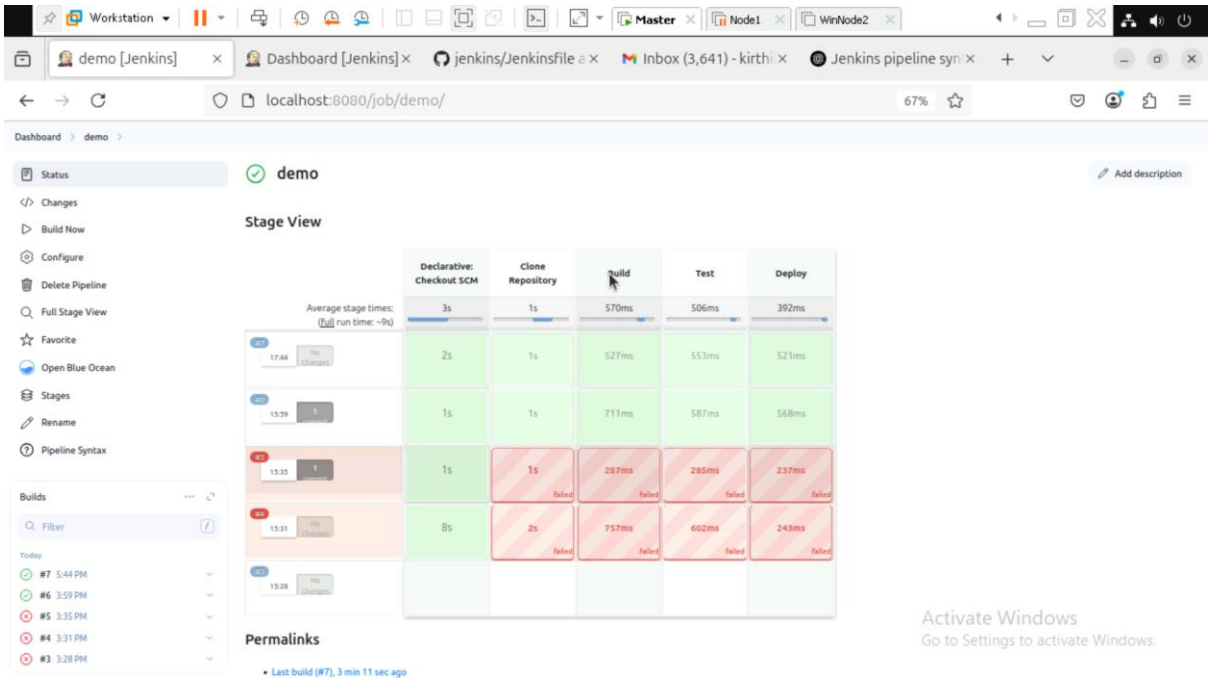
1. In Jenkins, click **New Item → Pipeline →** Give it a name → Click **OK**.
2. Scroll down to the **Pipeline** section and select **Pipeline script from SCM**.
3. In the **SCM** field, select **Git**, then enter your **GitHub repository URL**.
4. In the **Branch Specifier**, enter `*/main` or `*/master`.
5. In the **Script Path**, enter `Jenkinsfile` (which we will create in the next step).



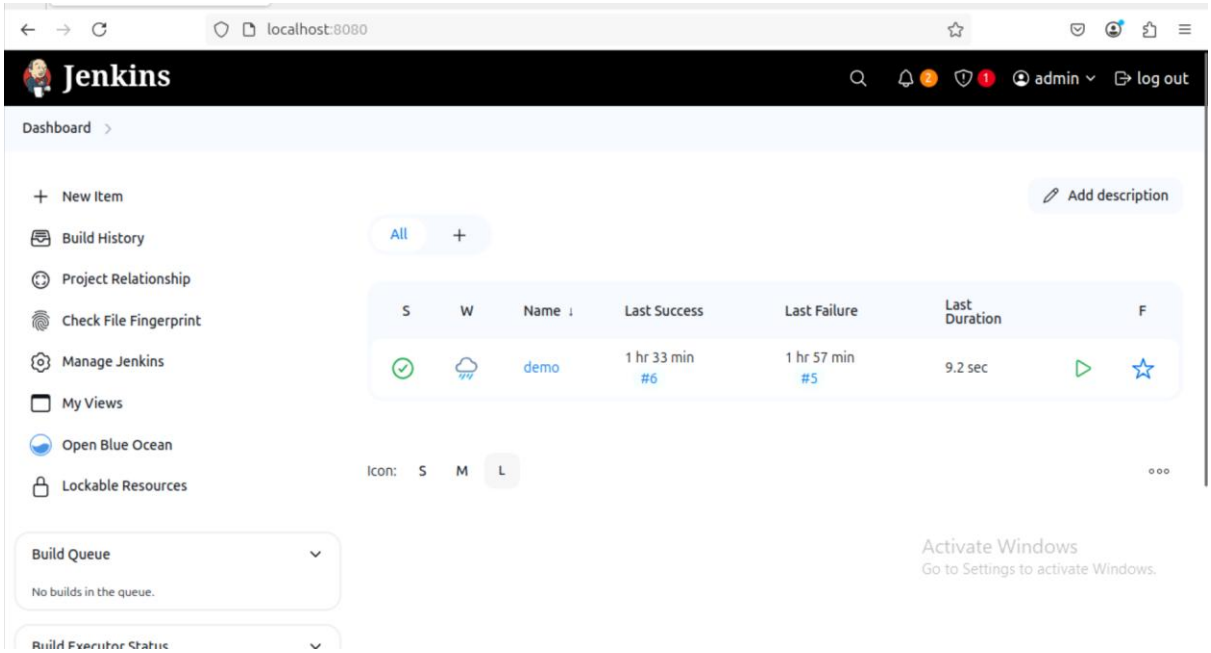
Build History:



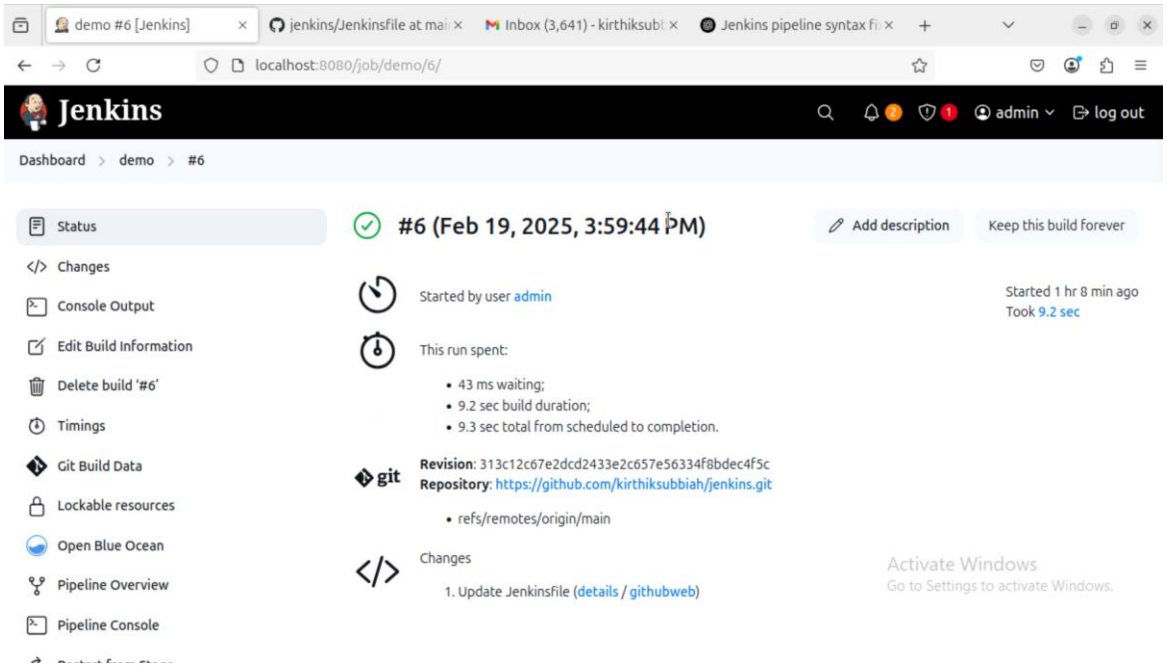
Status of Each stages of the pipeline:



Workspace:



Status of the Pipeline:



Step 3: Create a Jenkinsfile (CI/CD Pipeline)

In your GitHub repository, create a new file named **Jenkinsfile** and add the following code:

```
groovy
CopyEdit
pipeline {
    agent any
    stages {
        stage('Clone Repository') {
            steps {
                git 'https://github.com/your-username/your-repo.git'
            }
        }
        stage('Build') {
            steps {
                sh 'echo "Building the application..."'
            }
        }
        stage('Test') {
            steps {
                sh 'echo "Running tests..."'
            }
        }
        stage('Deploy') {
            steps {
                sh 'echo "Deploying application..."'
            }
        }
    }
}
```

Explanation:

- **Stage 1: Clone Repository** → Pulls the latest code from GitHub.
- **Stage 2: Build** → Builds the application (placeholder command).
- **Stage 3: Test** → Runs tests (placeholder command).
- **Stage 4: Deploy** → Deploys the app (placeholder command).

Step 4: Build and Run the Pipeline

1. Go back to **Jenkins Dashboard** → Click on your job.
2. Click **Build Now**.
3. Click **Console Output** to see the progress.

If everything is successful, you will see all **stages completed** in the output.

Step 5: Deploy with Docker (Optional)

If you want to deploy your application as a **Docker container**, modify your **Jenkinsfile** like this:

```
groovy
CopyEdit
pipeline {
    agent any
    environment {
        IMAGE_NAME = 'your-dockerhub-username/my-app'
    }
    stages {
        stage('Clone Repository') {
            steps {
                git 'https://github.com/your-username/your-repo.git'
            }
        }
        stage('Build Docker Image') {
            steps {
                sh 'docker build -t $IMAGE_NAME .'
            }
        }
        stage('Push Docker Image') {
            steps {
                withDockerRegistry([credentialsId: 'docker-hub-
credentials']) {
                    sh 'docker push $IMAGE_NAME'
                }
            }
        }
        stage('Deploy') {
            steps {
                sh 'docker run -d -p 8080:80 $IMAGE_NAME'
            }
        }
    }
}
```

Explanation:

- **Builds a Docker image** of your application.
- **Pushes it to Docker Hub** (requires credentials setup in Jenkins).
- **Deploys the container** on a local machine or server.

- You have successfully set up **Jenkins CI/CD** to **build, test, and deploy** a simple app!
- If using **Docker**, your app is now running on **http://localhost:8080**.

2. Automating a Python Application Build using Jenkins.

Step 1: Install Jenkins and Required Plugins

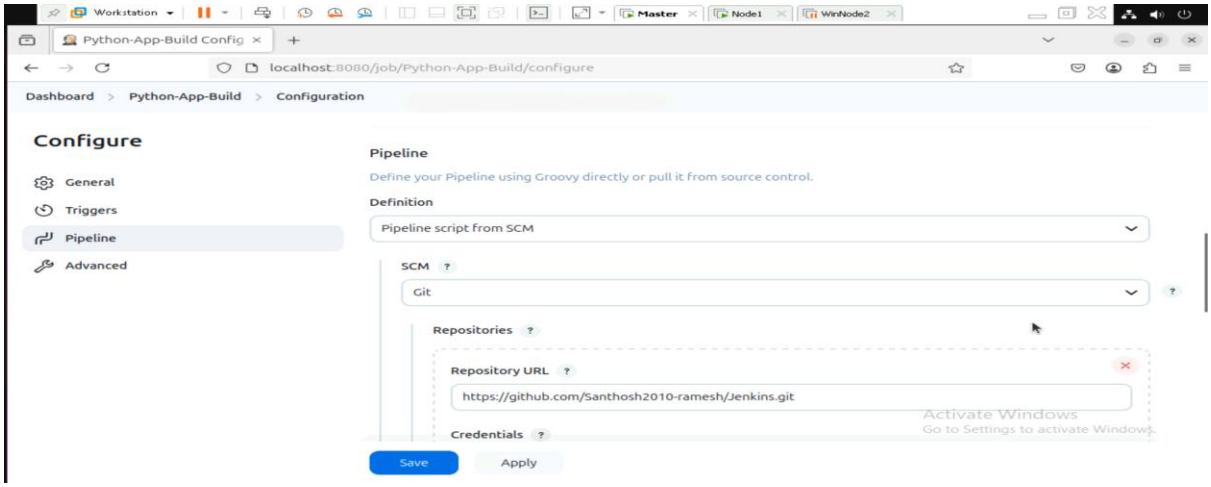
1. Start the Jenkins service and log in to the Jenkins dashboard.
2. Navigate to Manage Jenkins → Plugins.
3. Install the following plugins:
    - Pipeline Plugin (for defining Jenkins Pipelines)
    - Git Plugin (for pulling code from GitHub)
    - Build Tools Plugin (for executing builds)
4. Restart Jenkins to apply the changes.

Step 2: Create a New Jenkins Job

1. Open **Jenkins Dashboard**.
2. Click on **New Item** → Enter a project name.
3. Choose **Freestyle Project** or **Pipeline Project**.
4. Click **OK** to create the project.

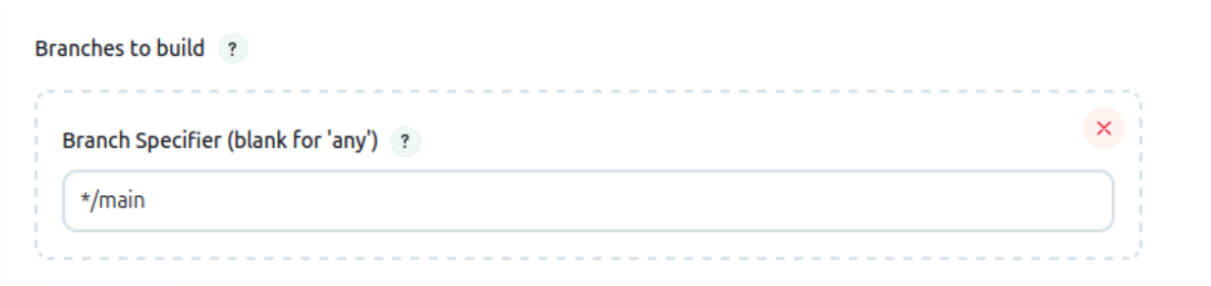
Step 3: Configure Jenkins to Pull Code from GitHub

1. Open the newly created Jenkins job.
2. Navigate to **Source Code Management**.
3. Select **Git** and enter the repository URL:
    - Example: https://github.com/your-repo/python-app.git
4. Configure GitHub credentials if authentication is required.



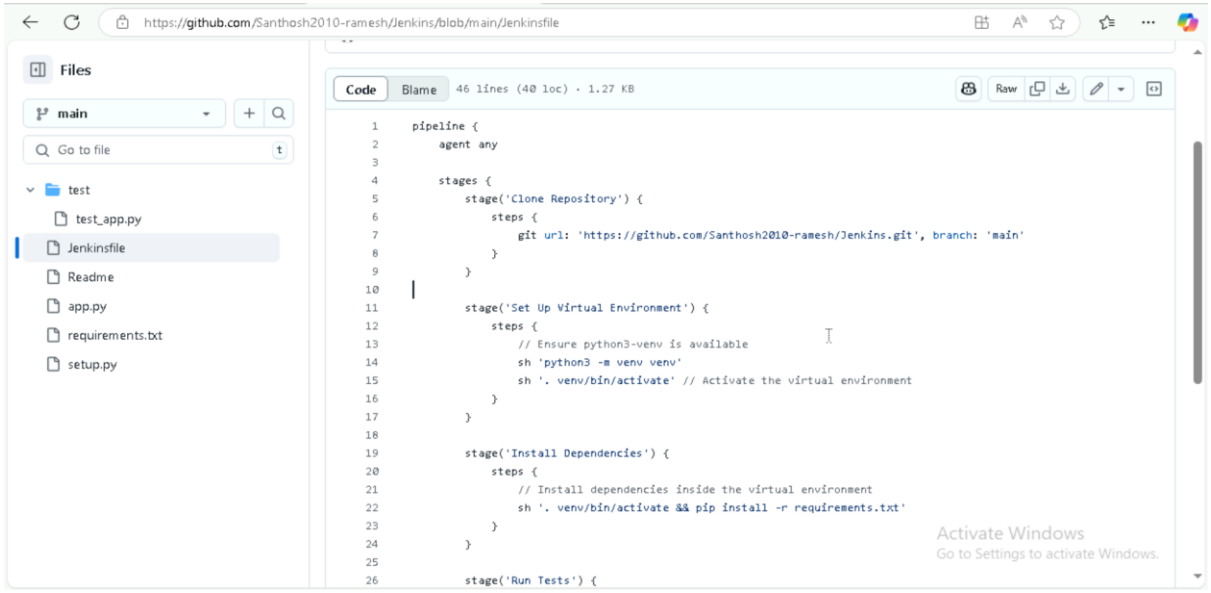
Step 4: Install Dependencies and Run Tests

Step 1: Adding the desired branch here we use Main branch.



Step 5: Pipeline with Python Scripts.

Step 1: Building the Declarative: Checkout SCM with the Script.

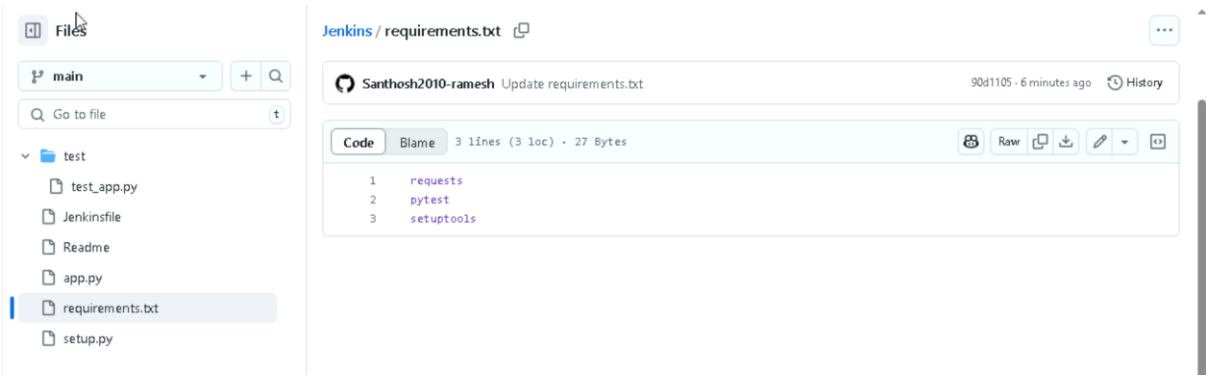


Step 2: Sample Python file.

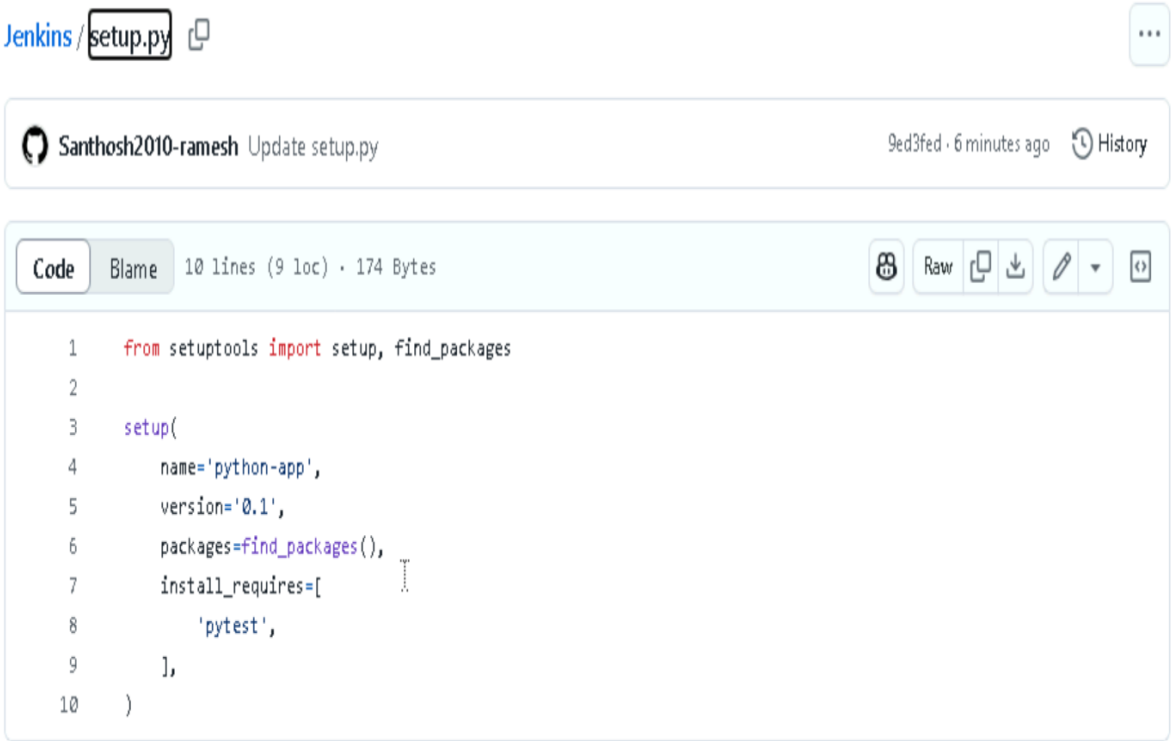
Code Blame 5 lines (4 loc) · 92 Bytes Raw

```
1  def hello_world():
2      print("Hello, World!")
3
4  if __name__ == "__main__":
5      hello_world()
```

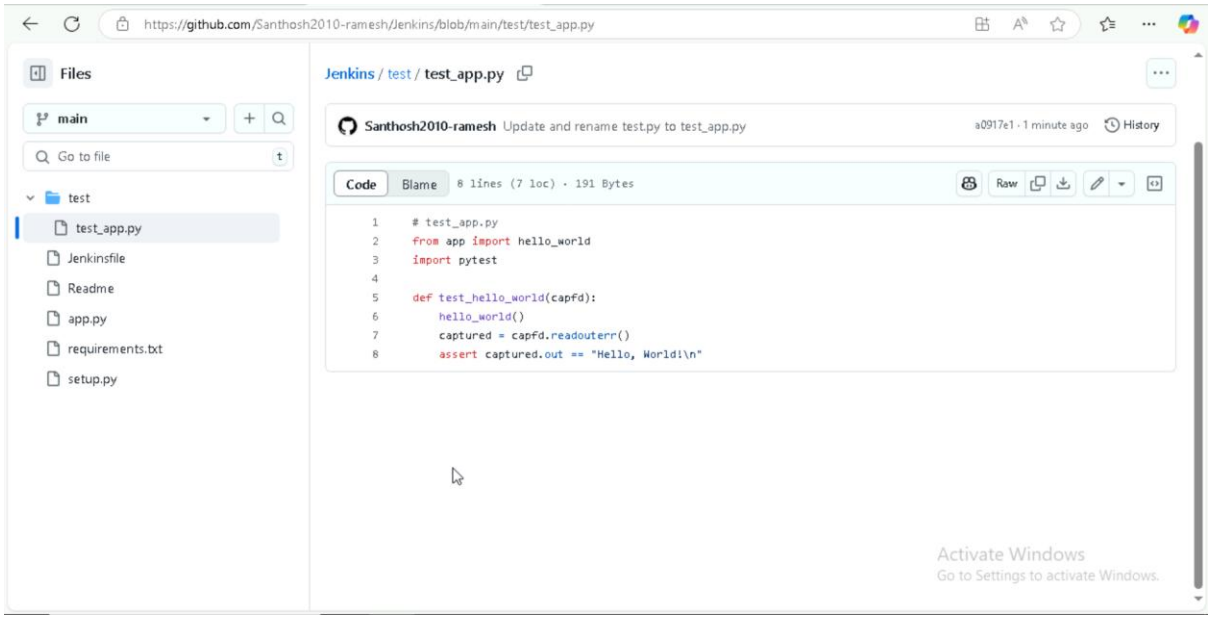
Step 3: Creating Requirements.txt file.



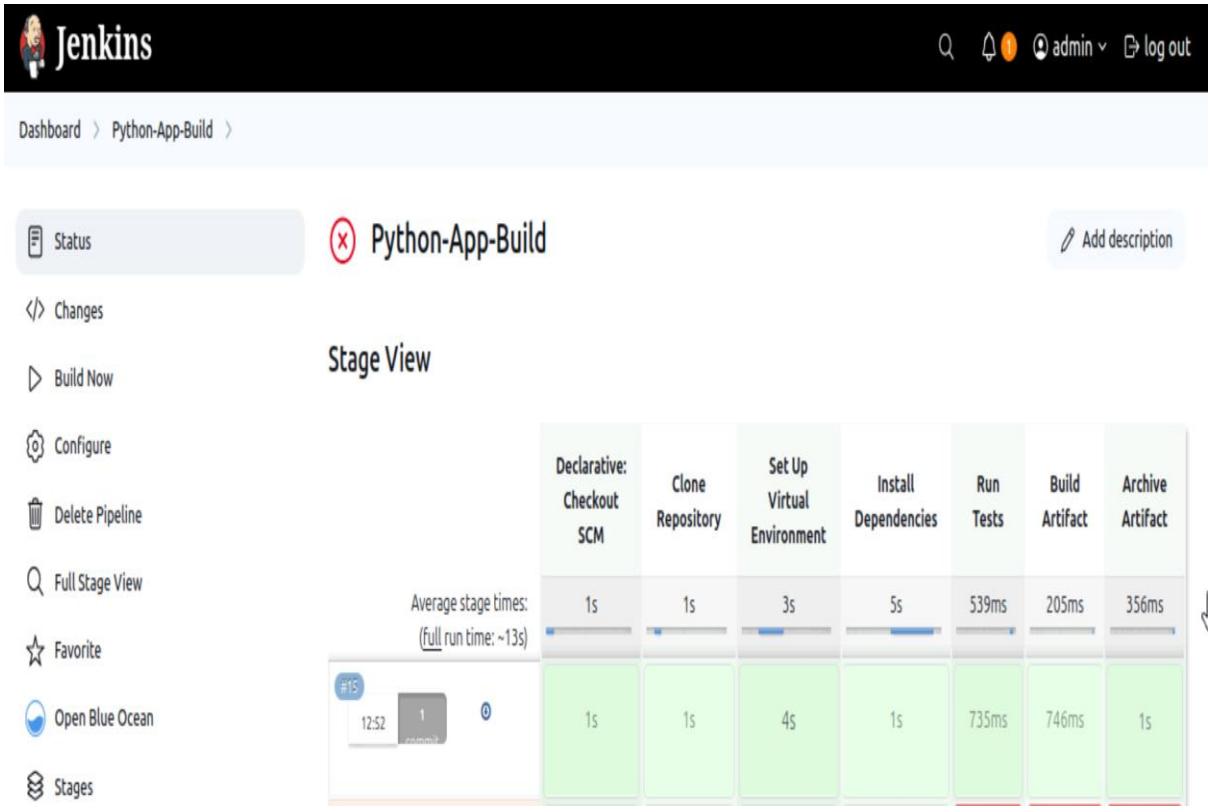
Step 4: Creating a Setup.py file.



Step 5: Creating a sample test.py for checkrun.



Step 6: Build and Archive Artifacts with Declarative: Checkout SCM, Clone Repository, set up Virtual Environment, Install Dependencies, Run Tests, Build Artifact, Archive Artifact.



Jenkins / test / test_app.py

Santhosh2010-ramesh Update and rename test.py to test_app.py a0917e1 · 5 minutes ago History

Code Blame 8 lines (7 loc) · 191 Bytes Raw

```
1  # test_app.py
2  from app import hello_world
3  import pytest
4
5  def test_hello_world(capfd):
6      hello_world()
7      captured = capfd.readouterr()
8      assert captured.out == "Hello, World!\n"
```

Step 7: Run the Jenkins Job

1. Click **Build Now** to trigger a build.
2. Monitor logs in **Console Output**.