1. **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.

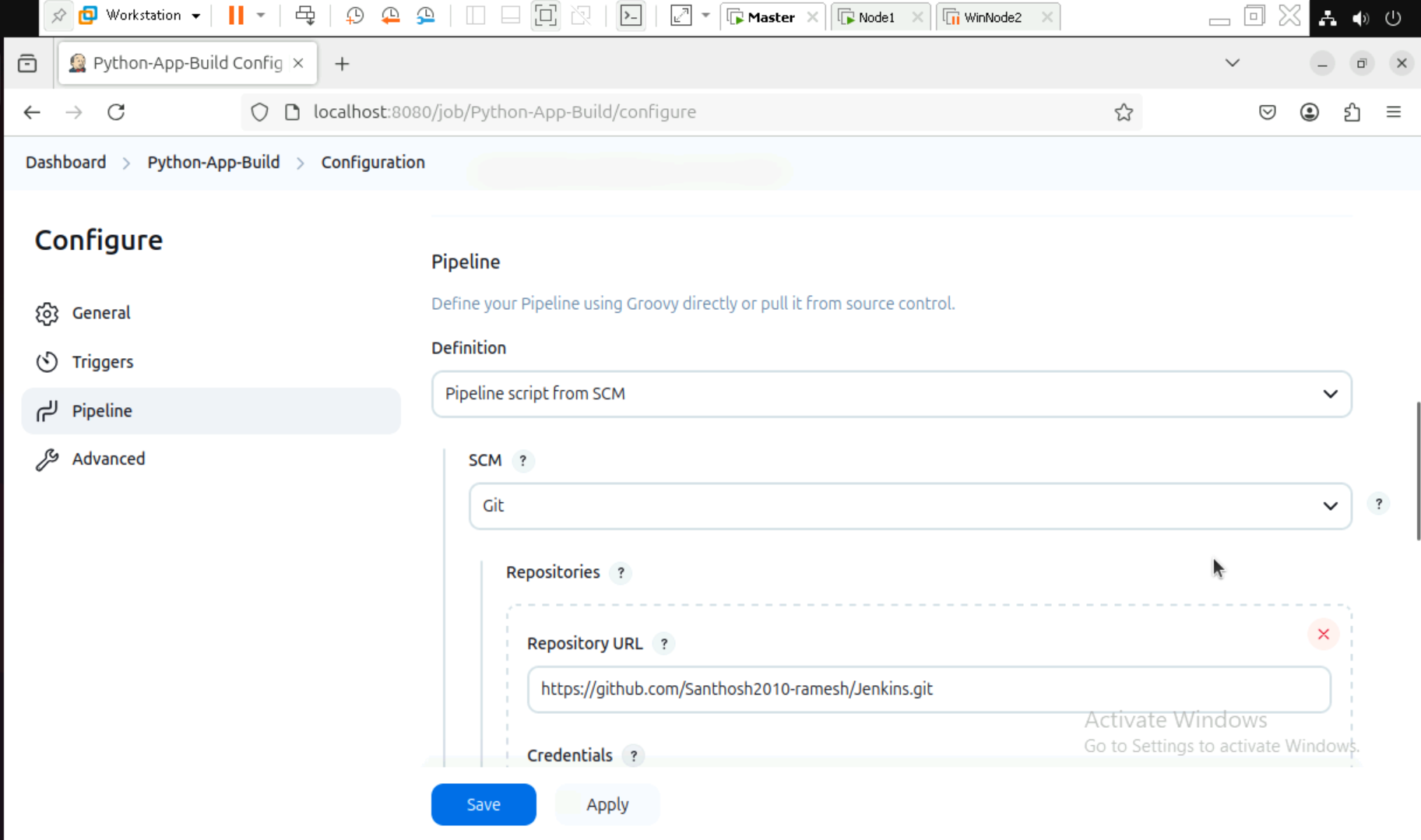
**Step 2: Create a New Jenkins Job**

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.



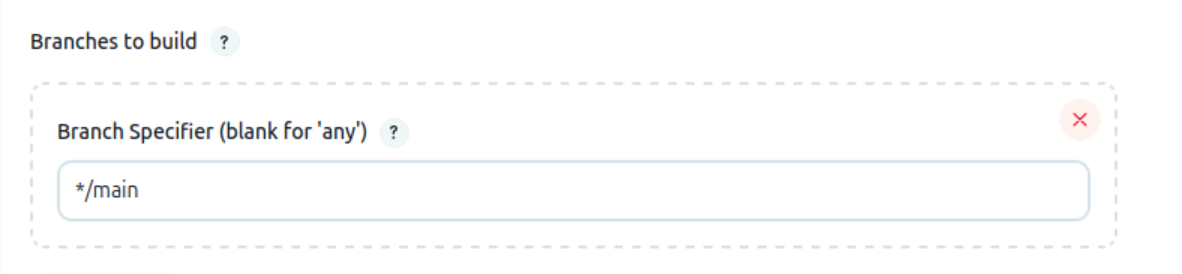
**Step 3: Configure Jenkins to Pull Code from GitHub**

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.



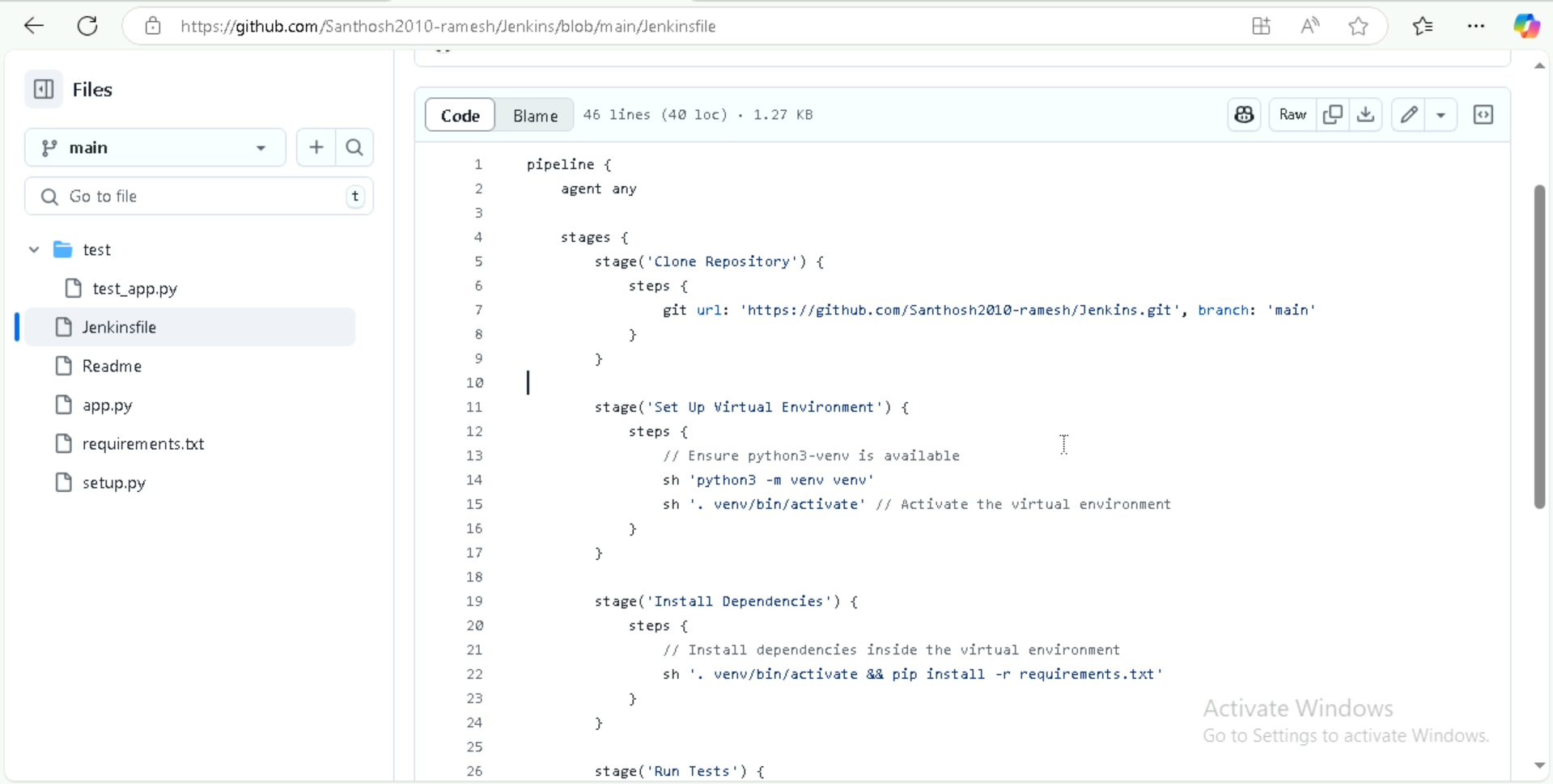
**Step 4: Install Dependencies and Run Tests**

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

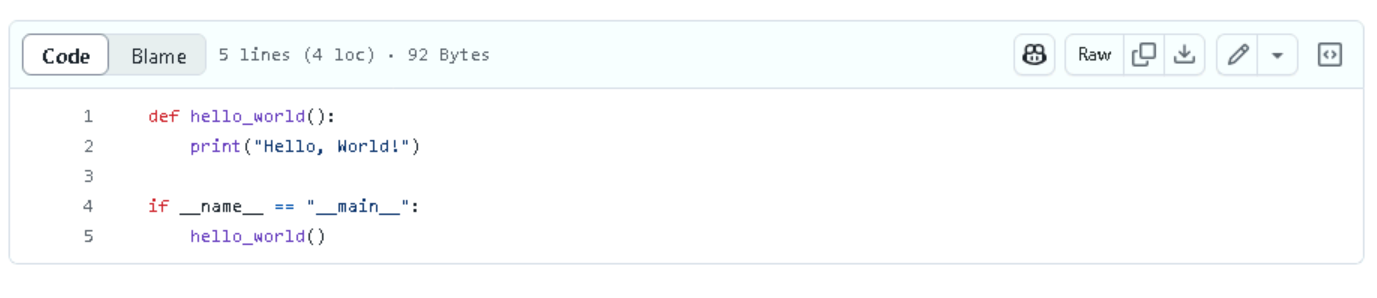


**Step 5: Pipeline with Python Scripts.**

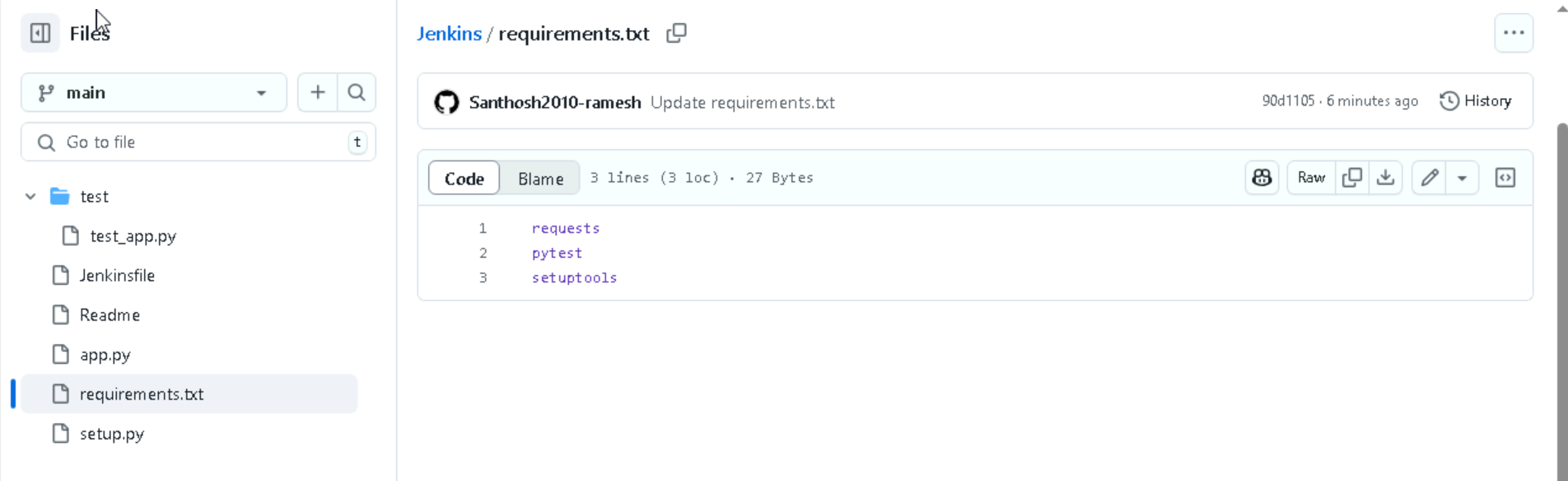
**Step 1: Building the Declarative: Checkout SCM with the Script.**

****

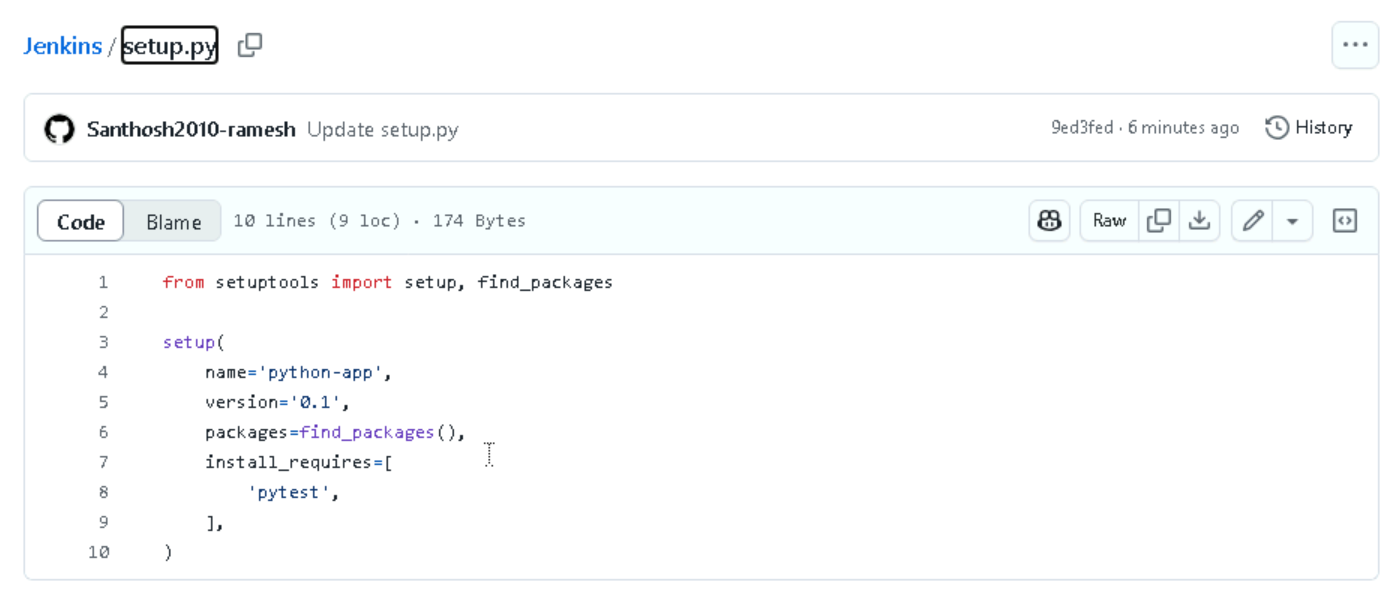
**Step 2: Sample Python file.**

****

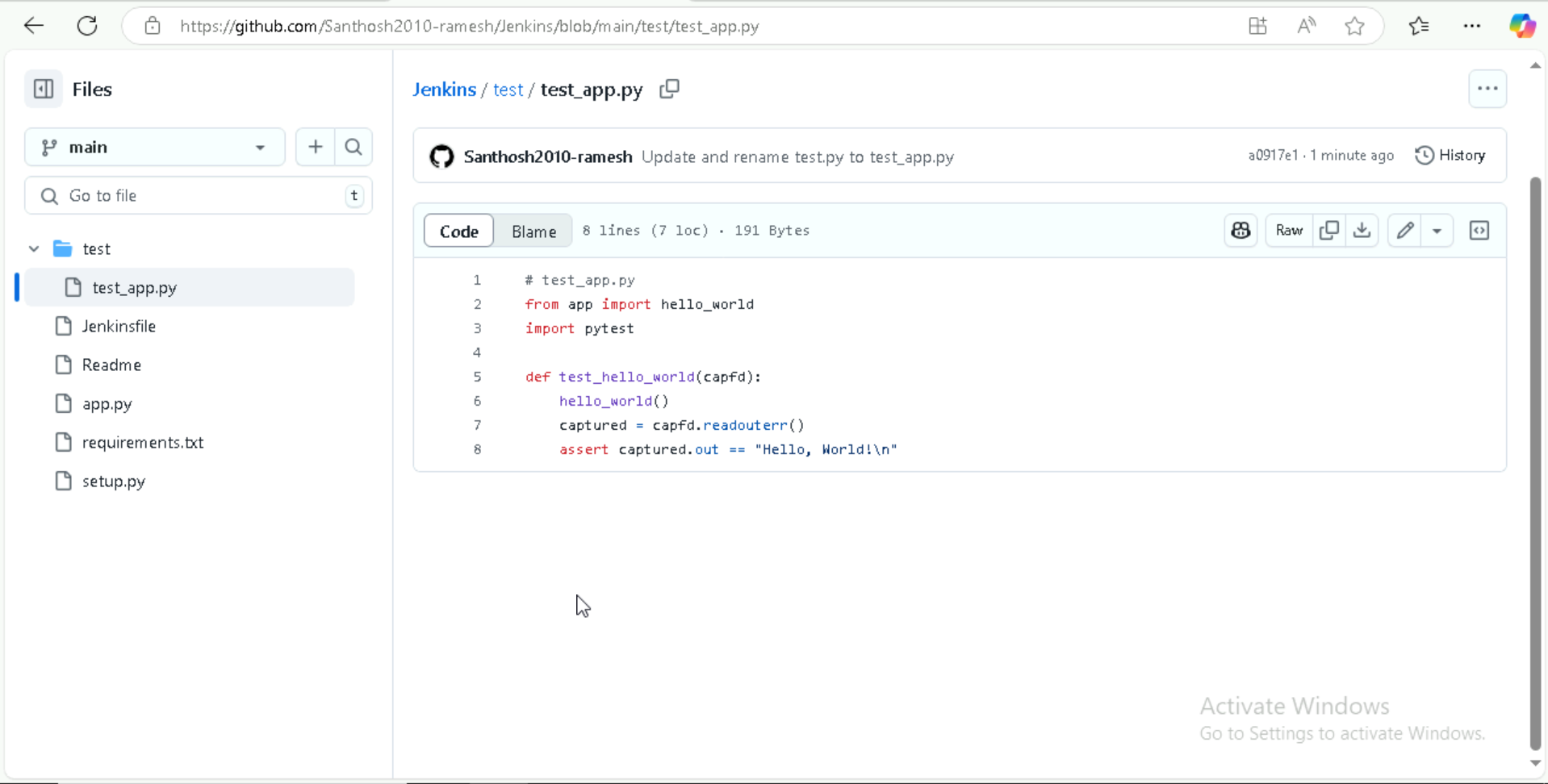
**Step 3: Creating Requirements.txt file.**

****

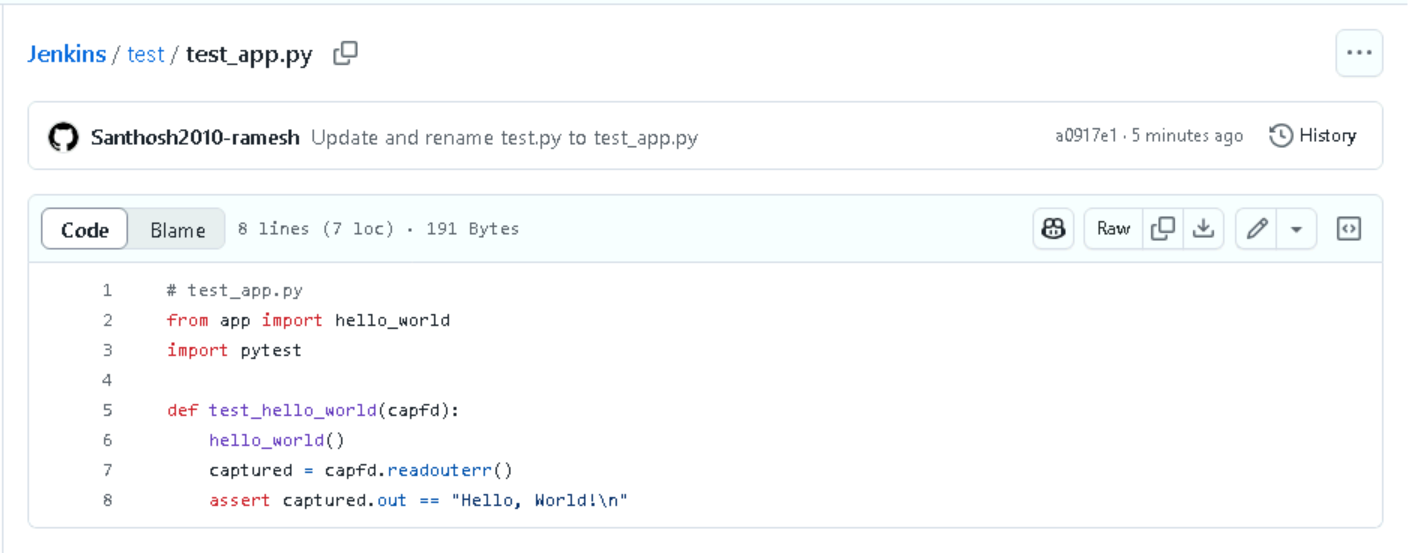
**Step 4: Creating a Setup.py file.**

****

**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.**

****

**Step 7: Run the Jenkins Job**

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

