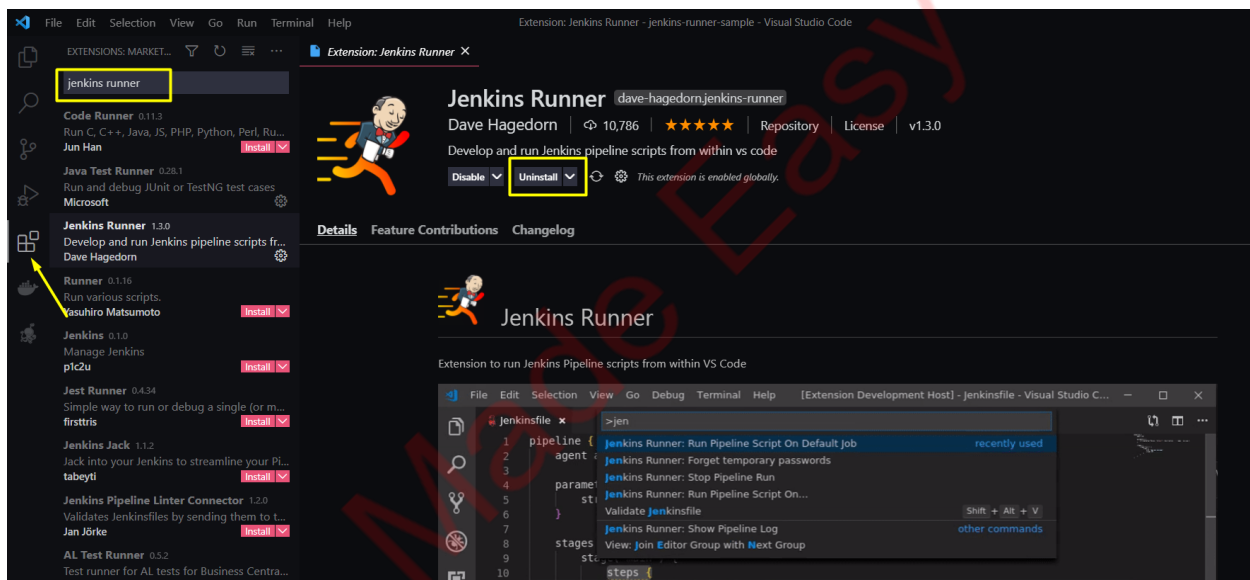


# Running your Jenkins Pipelines from VS Code using Jenkins Runner Plugin

We can test the Jenkins Pipeline directly from the VS code without modifying the pipeline in the Jenkins UI using VS Code's Jenkins Runner Plugin

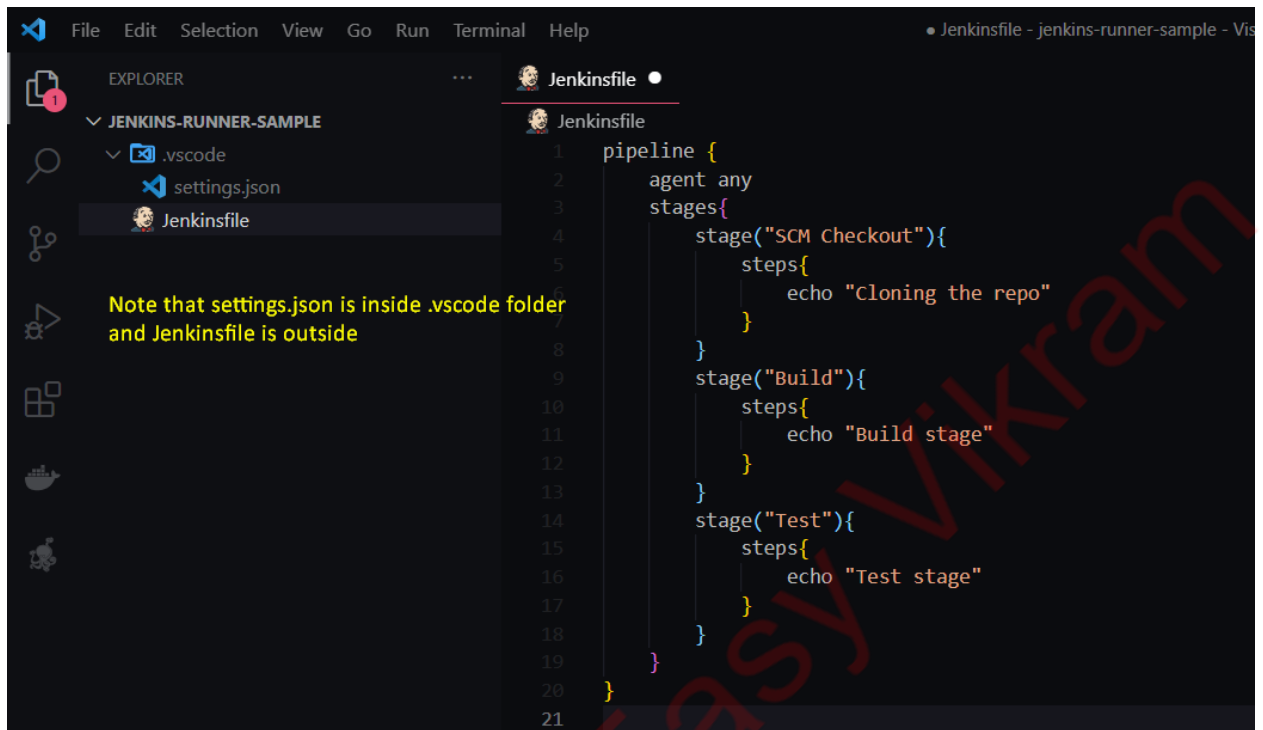
Step1: Download the Jenkins Runner plugin from VS code extensions

- Install Jenkins Runner Plugin



Step2: Create a folder within VS code

- Create a new folder in VS code
- Within this new folder, Create your **Jenkinsfile** and write your pipeline code
- Within this new folder create another folder **.vscode**
- Within **.vscode** folder, create **settings.json** file and edit the contents as shown in the next step



### Step3: Modify settings.json with Jenkins server and Job details

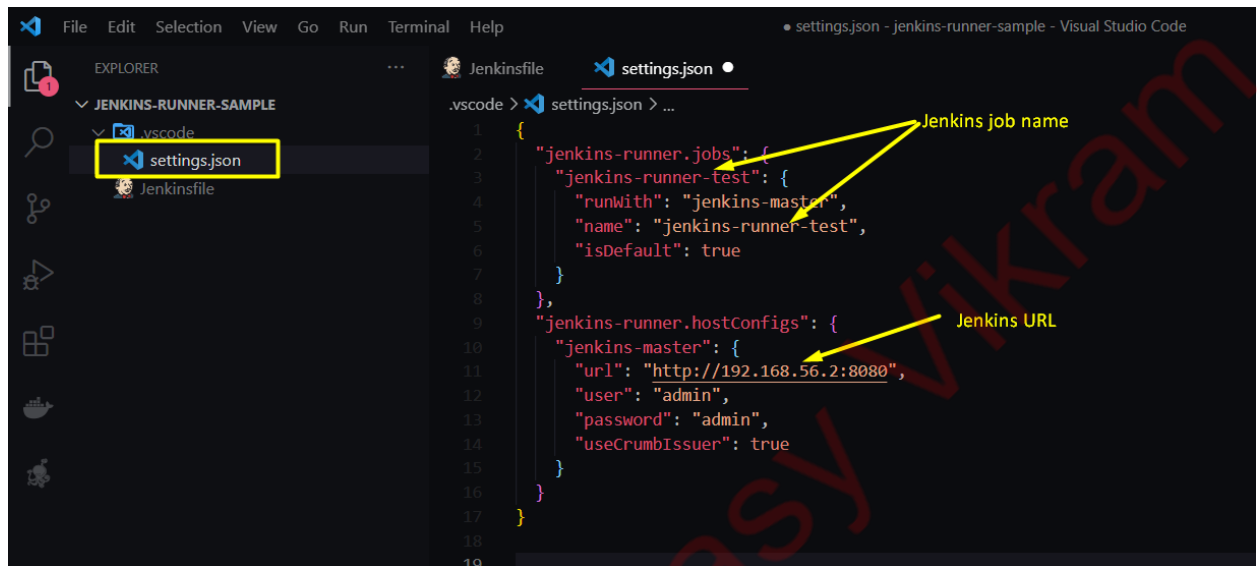
- The settings.json file looks like
 

```

{
  "jenkins-runner.jobs": {
    "jenkins-runner-test": {
      "runWith": "jenkins-master",
      "name": "jenkins-runner-test",
      "isDefault": true
    }
  },
  "jenkins-runner.hostConfigs": {
    "jenkins-master": {
      "url": "http://192.168.56.2:8080",
      "user": "admin",
      "password": "admin",
      "useCrumbIssuer": true
    }
  }
}

```

- Add **Jenkins Server URL, username, password** and **Job name** as required
- Job name to be taken from next step



#### Step4: Create Jenkins pipeline job

- Create a pipeline job and use it in **settings.json** file
- Follow the steps as shown in the screenshots

### Enter an item name

» Required field



#### Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.



#### Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.



#### Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.



#### Folder

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.



#### GitHub Organization

Creates a container organization (or user account) for all repositories matching some defined markers

#### General

[Build Triggers](#)[Advanced Project Options](#)[Pipeline](#)

#### Description

Jenkins runner test

[Plain text] [Preview](#)

☐ Discard old builds

**Pipeline**

Definition

Pipeline script

Script

1

try sample Pipeline...

Save Apply

Dashboard > jenkins-runner-test

Back to Dashboard

Status

Changes

Build Now

Configure

Delete Pipeline

Full Stage View

Rename

Pipeline Syntax

Build History trend ^

## Pipeline jenkins-runner-test

Jenkins runner test

Recent Changes

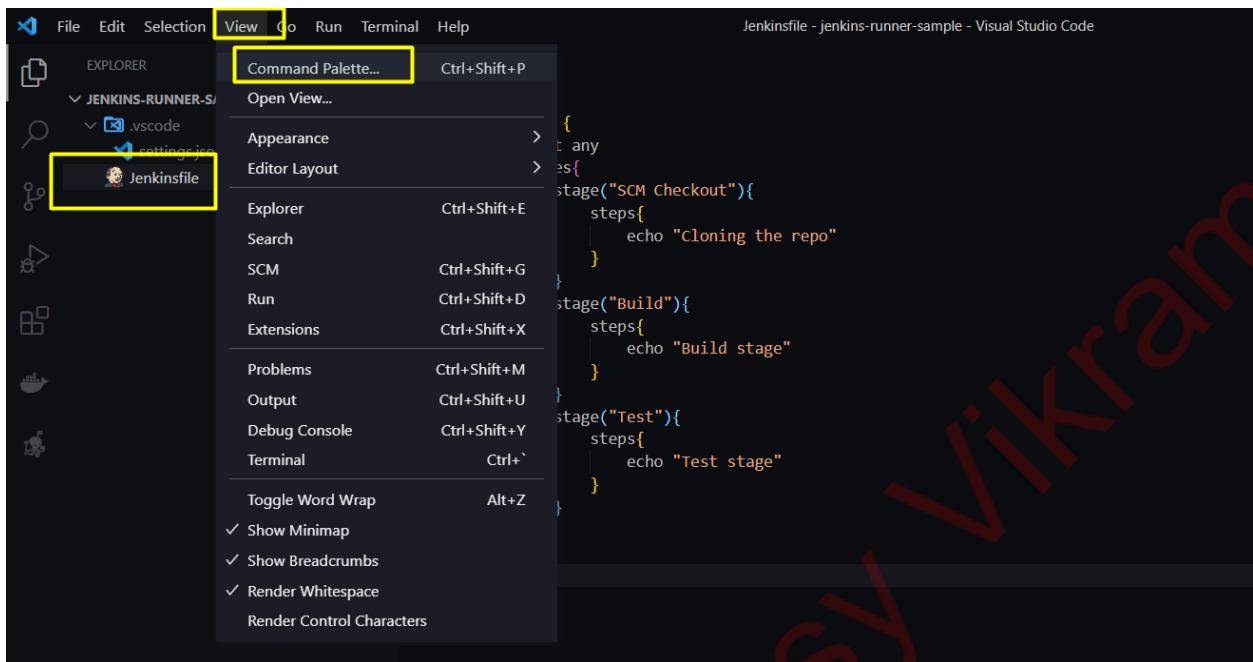
### Stage View

No data available. This Pipeline has not yet run.

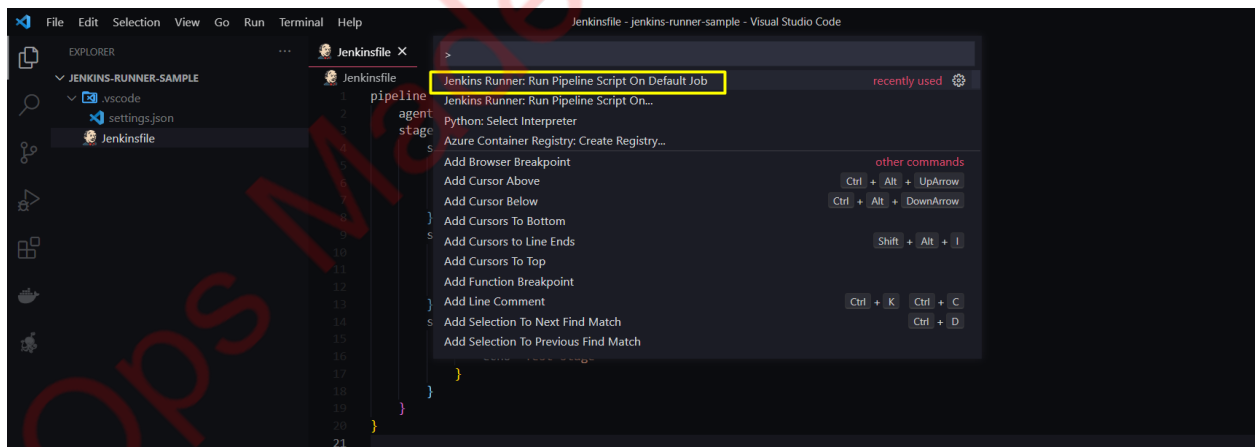
### Permalinks

#### Step5: Run the job from VS code

- Select the **Jenkinsfile** on the left pane, click on **view** on the menu bar and select **Command Palette**



- Search for **Jenkins Runner: Run Pipeline Script on Default Job** option and click on it



- You can view from terminal output the status of job

The screenshot shows the Visual Studio Code interface with a Jenkinsfile open in the editor. The Jenkinsfile defines a pipeline with three stages: 'SCM Checkout', 'Build', and 'Test'. The terminal output shows the pipeline running successfully on a Jenkins agent named 'any'. The output includes the start of the pipeline, the execution of the 'SCM Checkout' stage (echoing 'Cloning the repo'), and the start of the 'Build' stage.

```

Jenkinsfile
1 pipeline {
2   agent any
3   stages{
4     stage("SCM Checkout"){
5       steps{
6         echo "Cloning the repo"
7       }
8     }
9   }
10 }

```

```

running in durability level: PWA_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/jenkins-runner-test
[Pipeline] {
[Pipeline] stage
[Pipeline] { (SCM Checkout)
[Pipeline] echo
Cloning the repo
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] echo
Build stage
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Test)
[Pipeline] echo
Test stage
[Pipeline] }
[Pipeline] // stage
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

- Go to Jenkins UI and verify the same

The screenshot shows the Jenkins UI for the pipeline 'jenkins-runner-test'. The 'Stage View' is highlighted, showing a table of stage times for the 'SCM Checkout', 'Build', and 'Test' stages. The table indicates that the pipeline was last run on April 13 at 21:38 with no changes. The 'Permalinks' section is also visible at the bottom.

Stage	SCM Checkout	Build	Test
Average stage times:	47ms	61ms	41ms
(Average full run time: ~563ms)			
413 Apr 13 21:38 No Changes	47ms	61ms	41ms