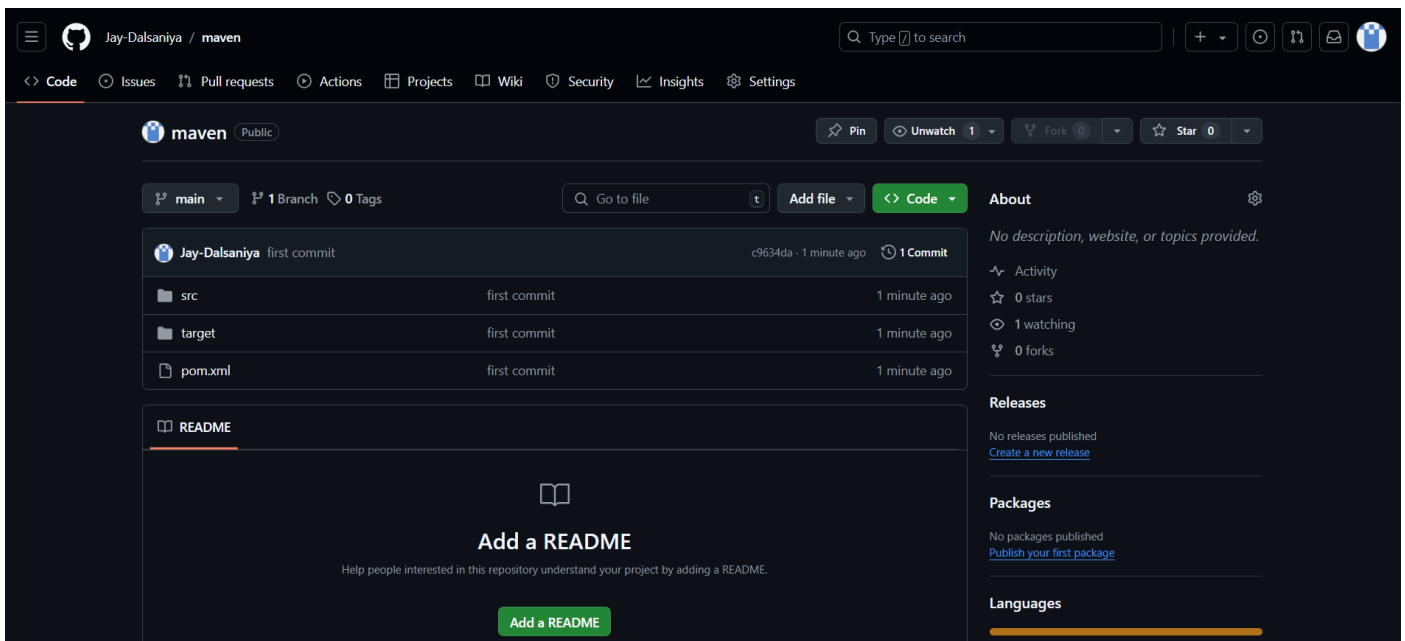


3. Create Jenkins job pipeline it from GitHub repo to run using build

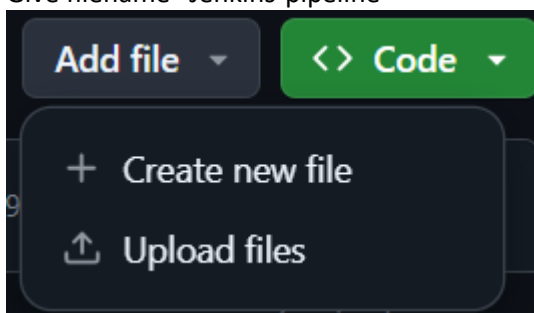
➤ Push the maven application on GitHub

- Go to 2nd practical maven application folder
- Open CMD
- `git init`
- `git add README.md`
- `git commit -m "first commit"`
- `git branch -M main`
- `git remote add origin https://github.com/Jay-Dalsaniya/maven.git`
- `git push -u origin main`

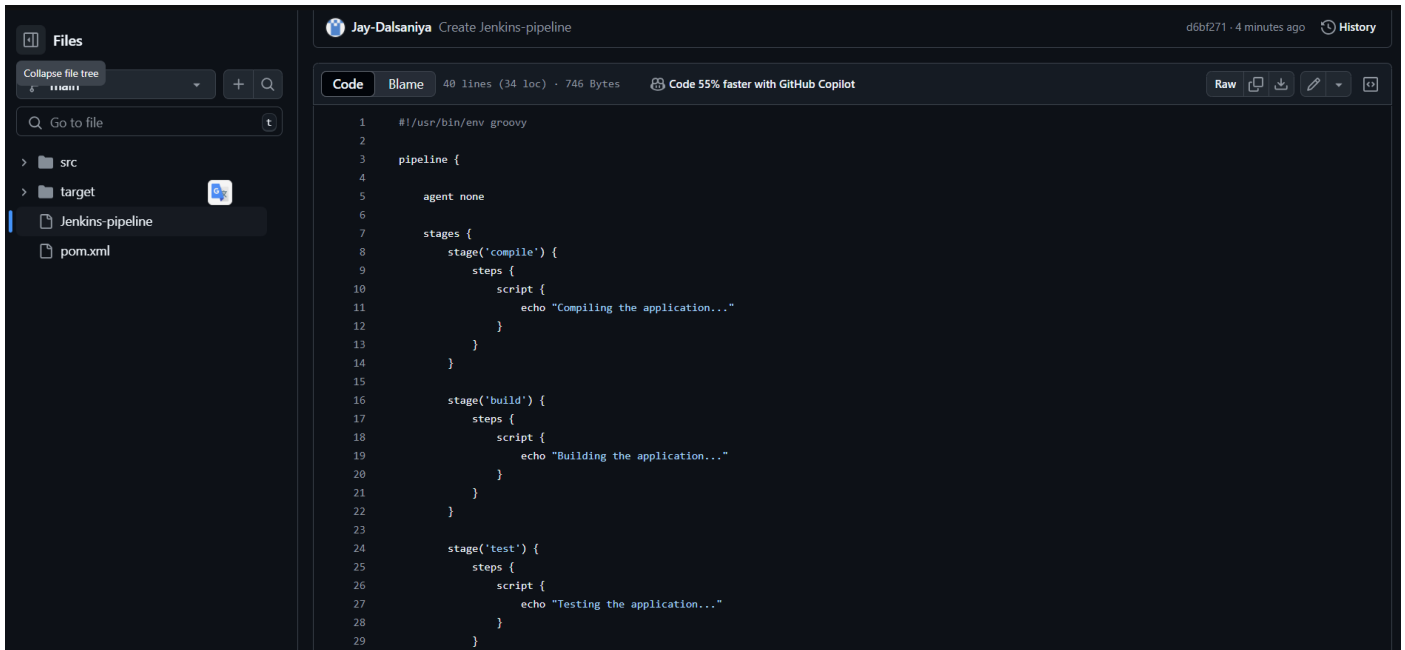


➤ Pipeline Script

- Create a new-file in git repo
- Give filename "Jenkins-pipeline"



- Write following script in that file



```
1  #!/usr/bin/env groovy
2
3  pipeline {
4
5      agent none
6
7      stages {
8          stage('compile') {
9              steps {
10                 script {
11                     echo "Compiling the application..."
12                 }
13             }
14         }
15
16         stage('build') {
17             steps {
18                 script {
19                     echo "Building the application..."
20                 }
21             }
22         }
23
24         stage('test') {
25             steps {
26                 script {
27                     echo "Testing the application..."
28                 }
29             }
30         }
31     }
32 }
```

- Click on the Jenkins Dashboard
 - Click on “new item”
 - Give the project Name

New Item

Enter an item name

Select an item type



Freestyle project

Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.



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.



Multibranch Pipeline

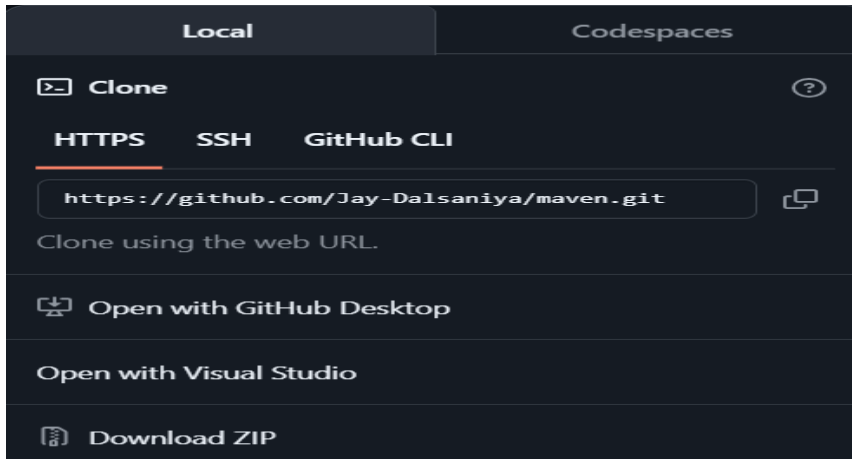
Creates a set of Pipeline projects according to detected branches in one SCM repository.



Organization Folder

Creates a set of multibranch project subfolders by scanning for repositories.

OK



Dashboard > practical3 > Configuration

Configure

- General
- Advanced Project Options
- Pipeline**

Pipeline

Definition

Pipeline script from SCM

SCM

Git

Repositories

Repository URL

https://github.com/Jay-Dalsaniya/maven.git

Credentials

- none -

+ Add

Advanced

Add Repository

Branches to build

Branch Specifier (blank for 'any')

*/main

Save

Apply

- Click on “Apply” and “save”

➤ Go to the Tools

● Add Maven

Dashboard > Manage Jenkins > Tools

Ant installations

Add Ant

Maven installations

Maven installations ^ Edited

Add Maven

Maven

Name

Maven

MAVEN_HOME

C:\Program Files\maven\apache-maven-3.9.8

☐ Install automatically ?

Add Maven

Save Apply

➤ Pipeline Script

● Write following script in “Jenkins-pipeline”

Jay-Dalsaniya / maven

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Files

main

Go to file

src

target

Jenkins-pipeline

pom.xml

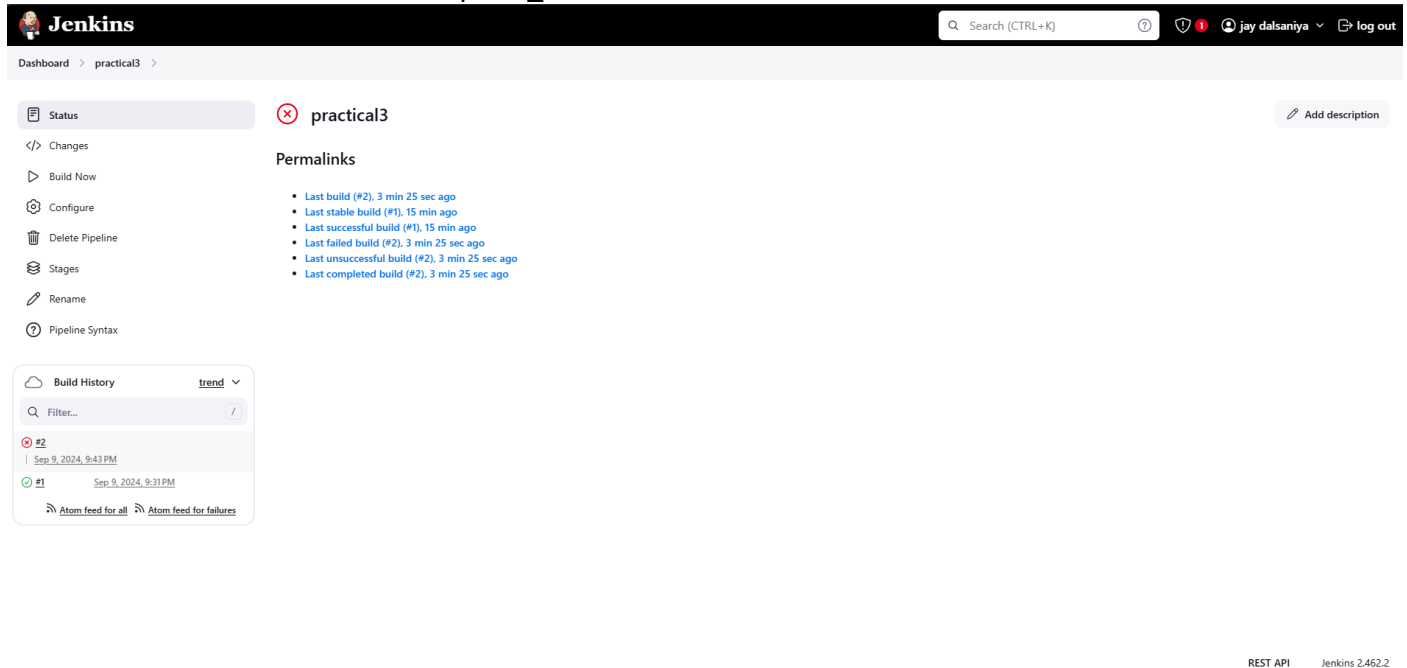
maven / Jenkins-pipeline

Jay-Dalsaniya Update Jenkins-pipeline cb94c74 · 4 minutes ago History

Code Blame 52 lines (51 loc) · 767 Bytes Code 55% faster with GitHub Copilot Raw

```
1 #!/usr/bin/env groovy
2
3 pipeline{
4   agent any
5   tools{
6     maven 'Maven'
7   }
8   stages{
9     stage('Validate Maven Project'){
10      steps{
11        script{
12          bat 'mvn validate'
13        }
14      }
15    }
16    stage('Compile Maven Project'){
17      steps{
18        script{
19          bat 'mvn compile'
20        }
21      }
22    }
23  }
24 }
```

➤ Go to Jenkins and Build now in pract_3



The screenshot shows the Jenkins web interface. At the top, there's a navigation bar with the Jenkins logo, a search bar, and user information (jay dalsaniya). Below the navigation bar, the breadcrumb trail shows 'Dashboard > practical3'. The main content area is divided into two columns. The left column contains a sidebar with various actions: Status (selected), Changes, Build Now, Configure, Delete Pipeline, Stages, Rename, and Pipeline Syntax. The right column displays the 'practical3' pipeline status, which is currently in a failed state (indicated by a red 'X' icon). Below the status, there's a 'Permalinks' section listing various build links. At the bottom left, a 'Build History' panel shows a list of builds, with the most recent build (#2) being failed and the previous build (#1) being successful. The bottom right corner of the interface shows 'REST API' and 'Jenkins 2.462.2'.

Dashboard > practical3

practical3 Add description

Permalinks

- Last build (#2), 3 min 25 sec ago
- Last stable build (#1), 15 min ago
- Last successful build (#1), 15 min ago
- Last failed build (#2), 3 min 25 sec ago
- Last unsuccessful build (#2), 3 min 25 sec ago
- Last completed build (#2), 3 min 25 sec ago

Build History trend

Filter...

#2 Sep 9, 2024, 9:43 PM

#1 Sep 9, 2024, 9:31 PM

Atom feed for all Atom feed for failures

REST API Jenkins 2.462.2