Lesson 01 Demo 02

Create Maven Project and Build It in Jenkins with SCM

Objective: To demonstrate the process of creating a Maven project in Eclipse, synchronizing it with Git, and configuring Jenkins to build the project using SCM (Source Code Management)

Tools Required: Eclipse IDE, Git, and Jenkins

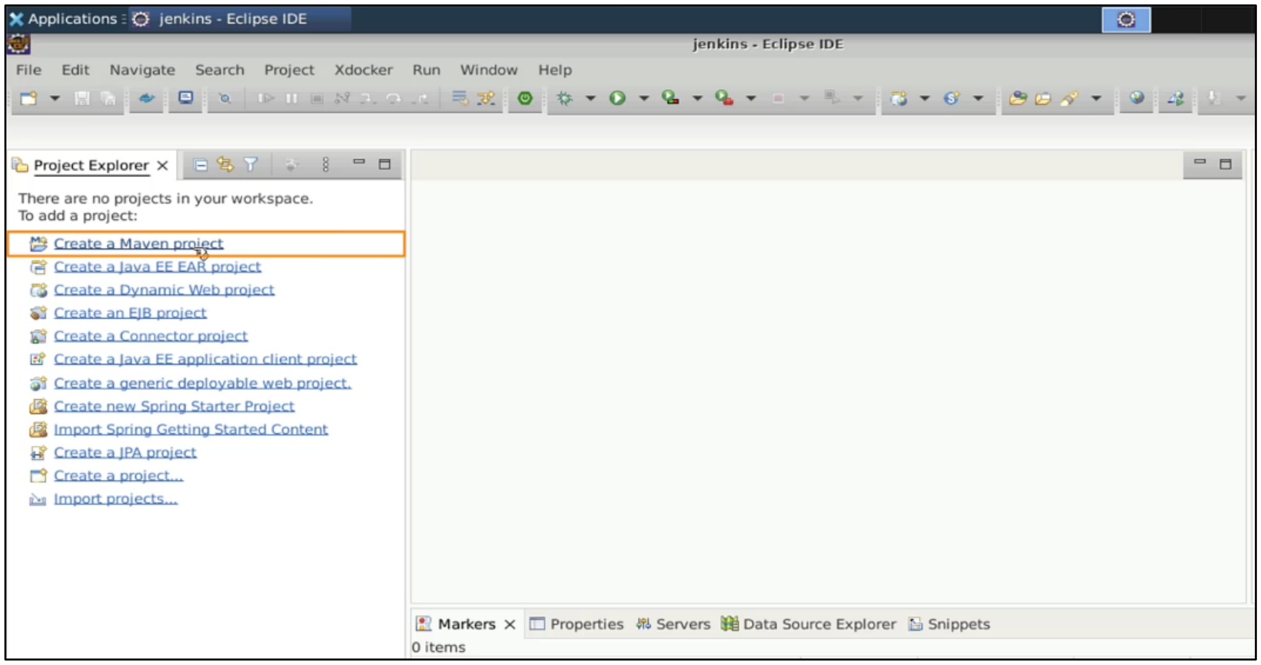
Prerequisites: None

Steps to be followed:

1. Creating a Maven project in Eclipse
2. Synchronizing the project with Git
3. Configuring Jenkins with Git

Step 1: Creating a Maven project in Eclipse

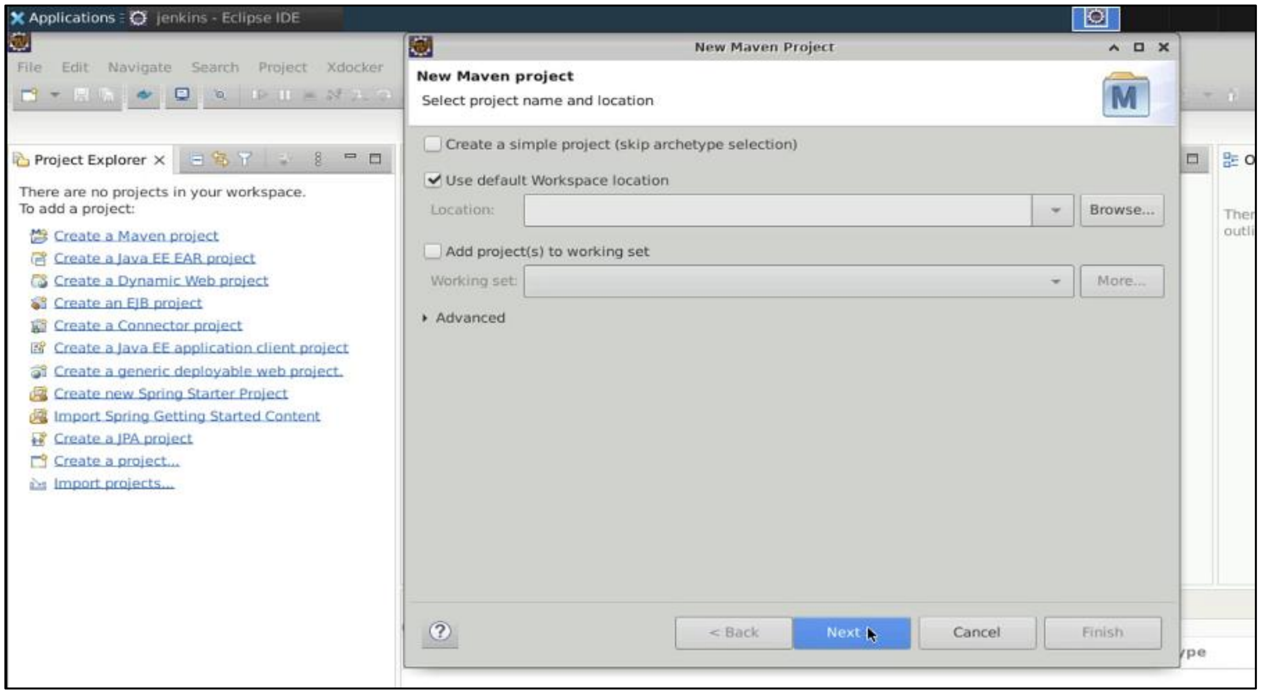
1.1 Open **Eclipse IDE** and select **Create a Maven project** option from the Project Explorer section


Applications
jenkins - Eclipse IDE
File Edit Navigate Search Project Xdocker Run Window Help
Project Explorer
Markers Properties Servers Data Source Explorer Snippets
0 items

There are no projects in your workspace.
To add a project:

- Create a Maven project
- Create a Java EE EAR project
- Create a Dynamic Web project
- Create an EJB project
- Create a Connector project
- Create a Java EE application client project
- Create a generic deployable web project.
- Create new Spring Starter Project
- Import Spring Getting Started Content
- Create a JPA project
- Create a project...
- Import projects...

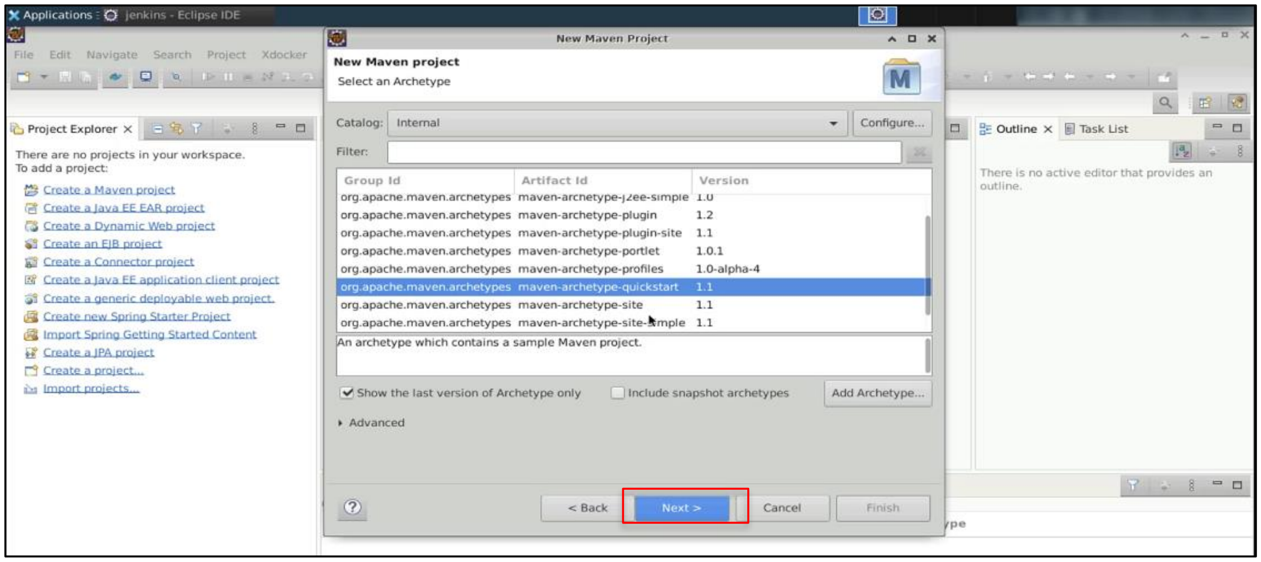
1.2 Click on **Next**



New Maven project
Select project name and location

- [ ] Create a simple project (skip archetype selection)
- [✓] Use default Workspace location
- Location:
- [ ] Add project(s) to working set
- Working set:

1.3 Select the **maven-archetype-quickstart** option from the **Select an Archetype** tab and click on the **Next** button


Applications : jenkins - Eclipse IDE
File Edit Navigate Search Project Xdocker
Project Explorer
Outline
Task List
There is no active editor that provides an outline.
Create a Maven project
Create a Java EE EAR project
Create a Dynamic Web project
Create an EJB project
Create a Connector project
Create a Java EE application client project
Create a generic deployable web project.
Create new Spring Starter Project
Import Spring Getting Started Content
Create a JPA project
Create a project...
Import projects...
Configure...
Add Archetype...
Advanced
< Back
Next >
Cancel
Finish

New Maven Project

New Maven project

Select an Archetype

There are no projects in your workspace.
To add a project:

Catalog: Internal

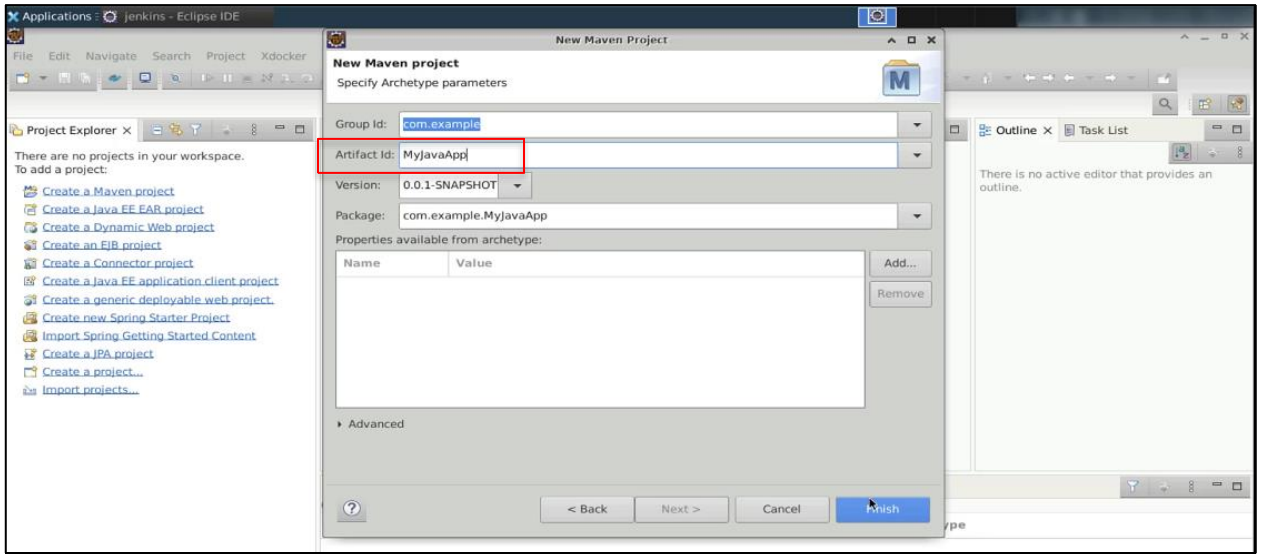
Filter:

Group Id	Artifact Id	Version
org.apache.maven.archetypes	maven-archetype-j2ee-simple	1.0
org.apache.maven.archetypes	maven-archetype-plugin	1.2
org.apache.maven.archetypes	maven-archetype-plugin-site	1.1
org.apache.maven.archetypes	maven-archetype-portlet	1.0.1
org.apache.maven.archetypes	maven-archetype-profiles	1.0-alpha-4
org.apache.maven.archetypes	maven-archetype-quickstart	1.1
org.apache.maven.archetypes	maven-archetype-site	1.1
org.apache.maven.archetypes	maven-archetype-site-simple	1.1

An archetype which contains a sample Maven project.

[✓] Show the last version of Archetype only [ ] Include snapshot archetypes

1.4 Enter **MyJavaApp** in the Artifact Id field and click on the **Finish** button


Applications : jenkins - Eclipse IDE
File Edit Navigate Search Project Xdocker
Project Explorer
Outline
Task List
There is no active editor that provides an outline.
Create a Maven project
Create a Java EE EAR project
Create a Dynamic Web project
Create an EJB project
Create a Connector project
Create a Java EE application client project
Create a generic deployable web project.
Create new Spring Starter Project
Import Spring Getting Started Content
Create a JPA project
Create a project...
Import projects...
Add...
Remove
Advanced
< Back
Next >
Cancel
Finish

New Maven Project

New Maven project

Specify Archetype parameters

There are no projects in your workspace.
To add a project:

Group Id: com.example

Artifact Id: MyJavaApp

Version: 0.0.1-SNAPSHOT

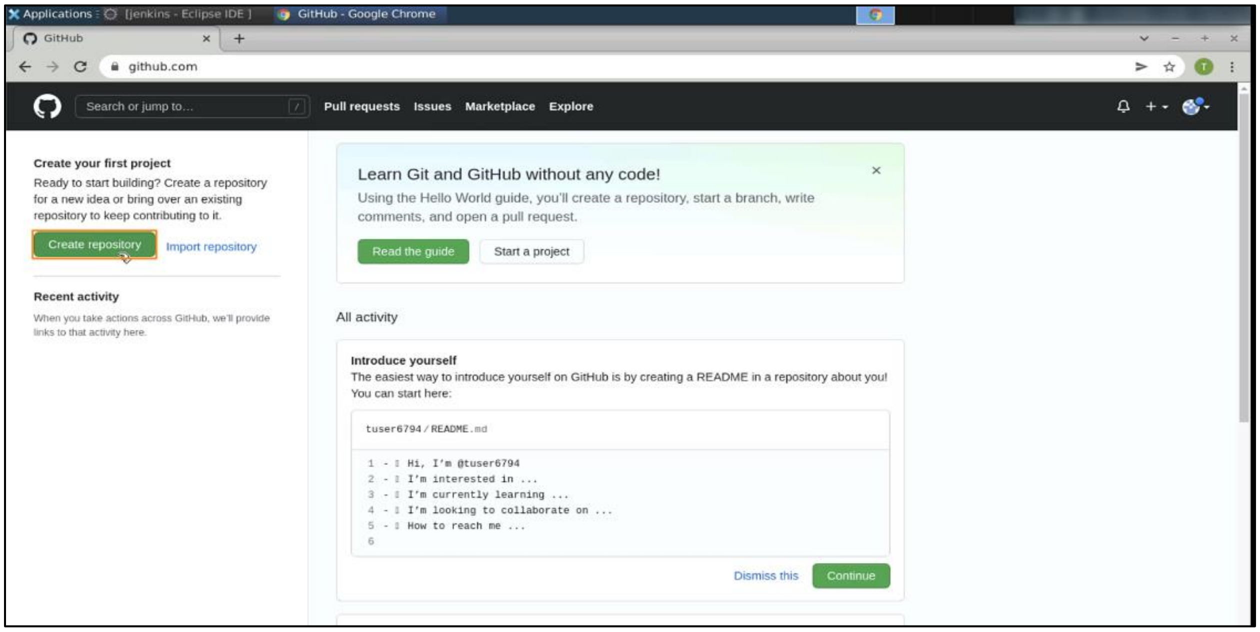
Package: com.example.MyJavaApp

Properties available from archetype:

Name	Value

Step 2: Synchronizing the project with Git

2.1 Go to **GitHub** and click on the **Create repository** button



Create your first project

Ready to start building? Create a repository for a new idea or bring over an existing repository to keep contributing to it.

Recent activity

When you take actions across GitHub, we'll provide links to that activity here.

Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

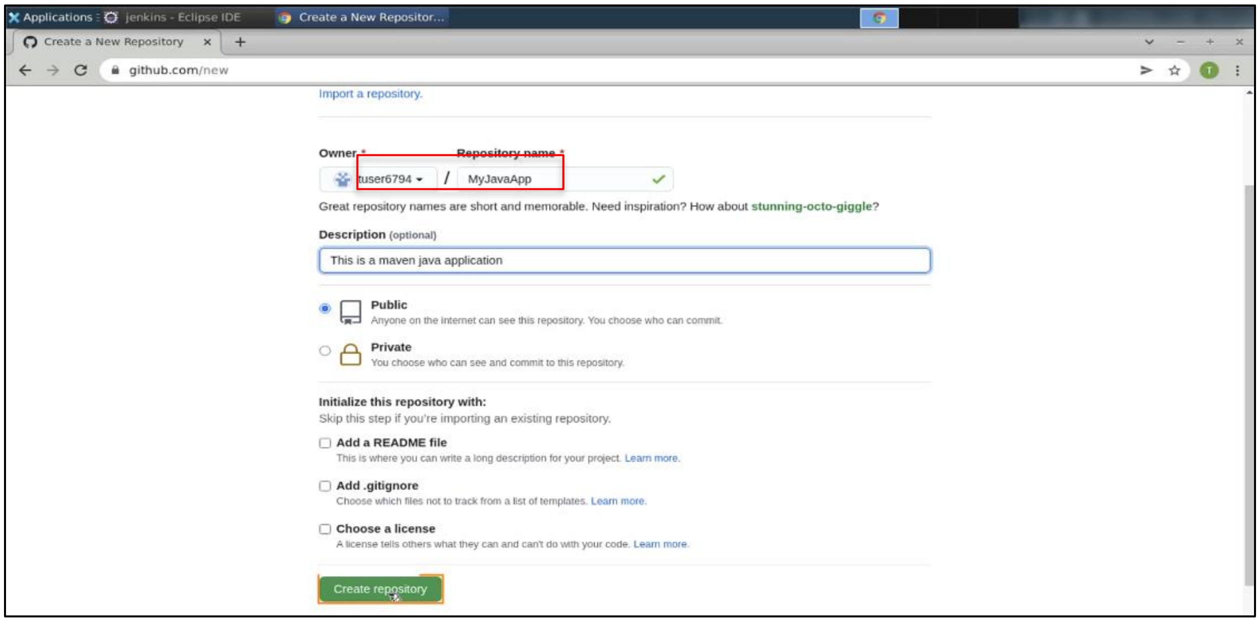
All activity

Introduce yourself

The easiest way to introduce yourself on GitHub is by creating a README in a repository about you! You can start here:

```
tuser6794 / README.md

- ▯ Hi, I'm @tuser6794
- ▯ I'm interested in ...
- ▯ I'm currently learning ...
- ▯ I'm looking to collaborate on ...
- ▯ How to reach me ...
```

2.2 Enter **MyJavaApp** in the Repository name field, add a description, and click on the **Create repository** button



Owner * tuser6794 / Repository name * MyJavaApp

Great repository names are short and memorable. Need inspiration? How about **stunning-octo-giggle**?

Description (optional)

This is a maven java application

● **Public**
Anyone on the internet can see this repository. You choose who can commit.

○ **Private**
You choose who can see and commit to this repository.

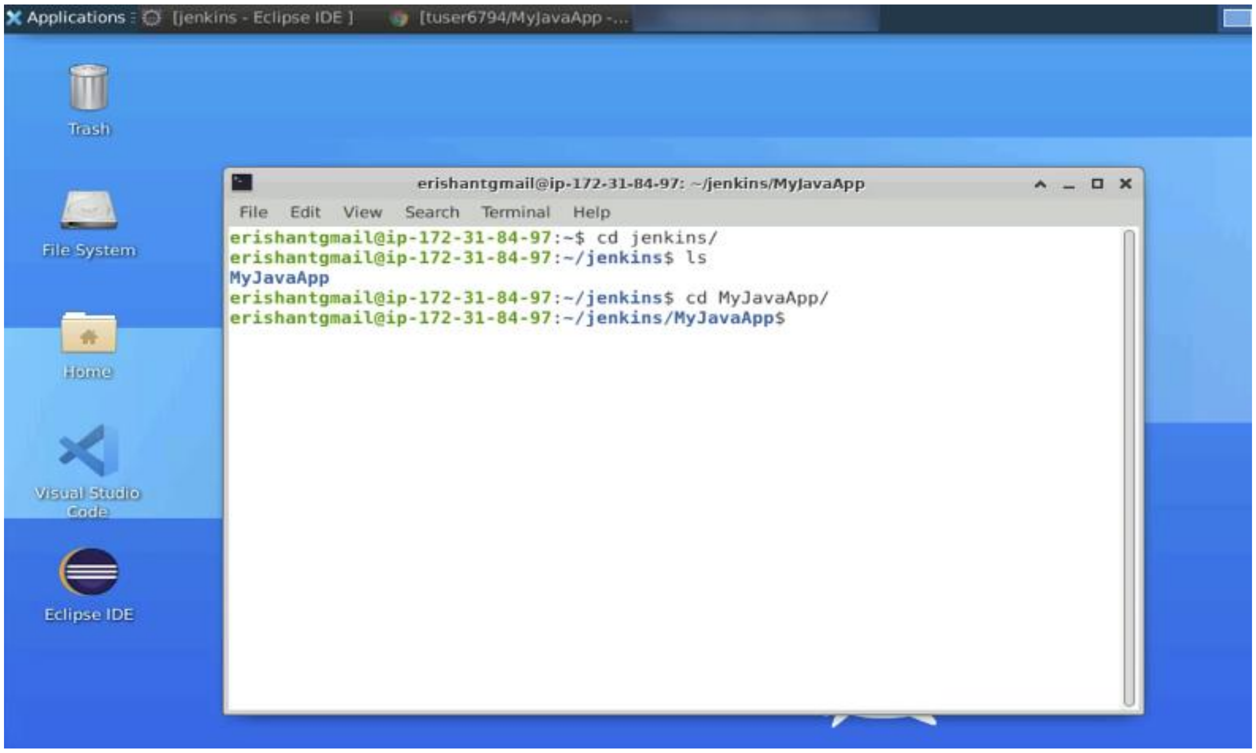
Initialize this repository with:
Skip this step if you're importing an existing repository.

☐ **Add a README file**
This is where you can write a long description for your project. Learn more.

☐ **Add .gitignore**
Choose which files not to track from a list of templates. Learn more.

☐ **Choose a license**
A license tells others what they can and can't do with your code. Learn more.

2.3 Go to the terminal and navigate to the MyJavaApp project using the **cd** command

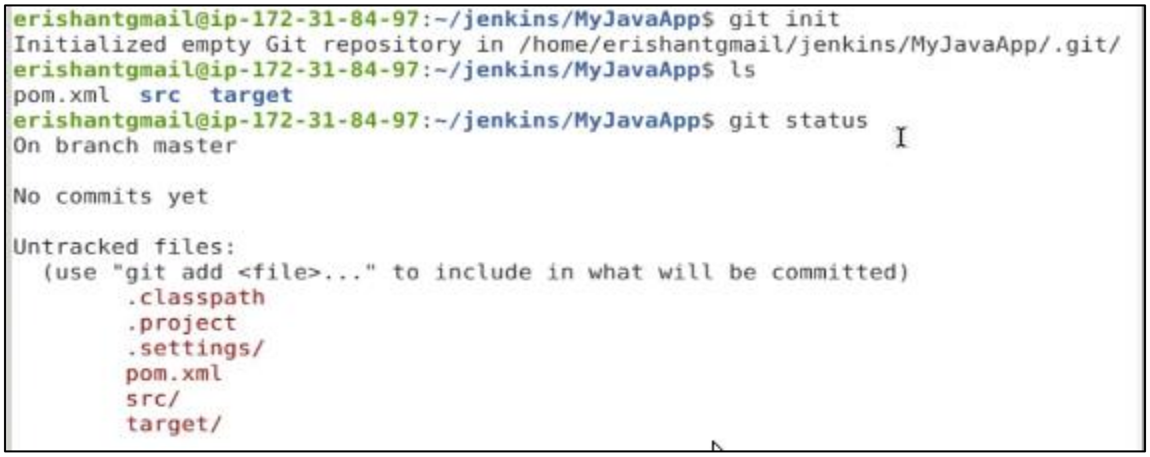


```
erishantgmail@ip-172-31-84-97:~$ cd jenkins/
erishantgmail@ip-172-31-84-97:~/jenkins$ ls
MyJavaApp
erishantgmail@ip-172-31-84-97:~/jenkins$ cd MyJavaApp/
erishantgmail@ip-172-31-84-97:~/jenkins/MyJavaApp$
```

2.4 Initialize and check the status of the Git repository using the following commands:

git init

git status

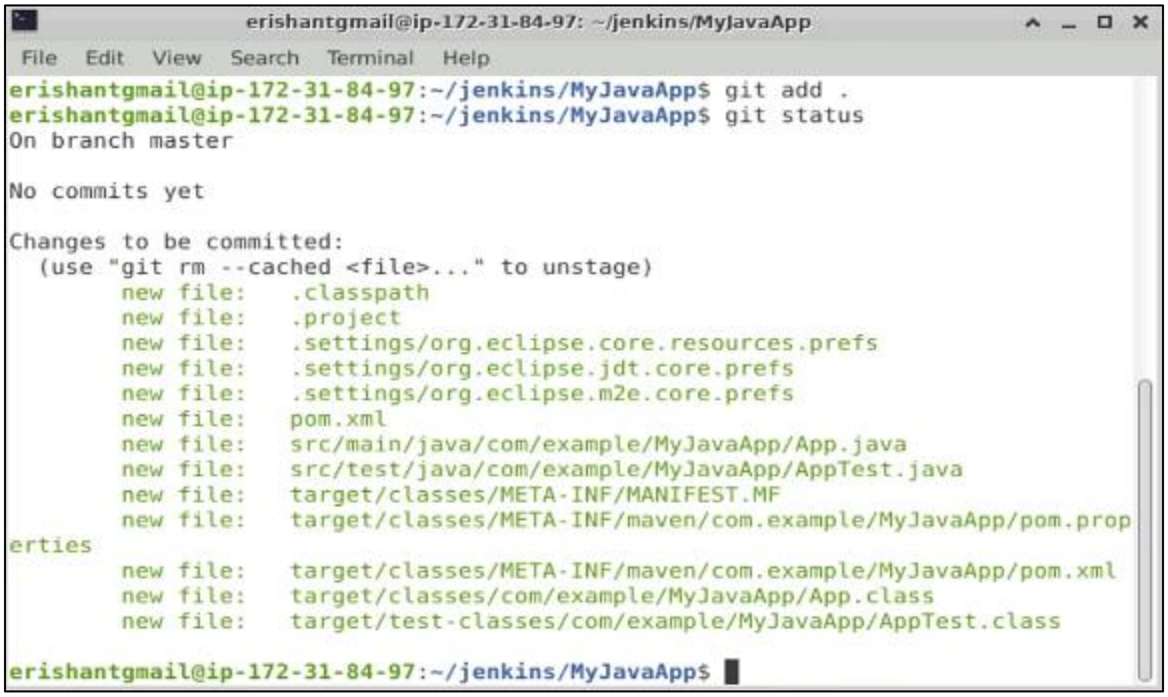


```
erishantgmail@ip-172-31-84-97:~/jenkins/MyJavaApp$ git init
Initialized empty Git repository in /home/erishantgmail/jenkins/MyJavaApp/.git/
erishantgmail@ip-172-31-84-97:~/jenkins/MyJavaApp$ ls
pom.xml  src  target
erishantgmail@ip-172-31-84-97:~/jenkins/MyJavaApp$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .classpath
        .project
        .settings/
        pom.xml
        src/
        target/
```

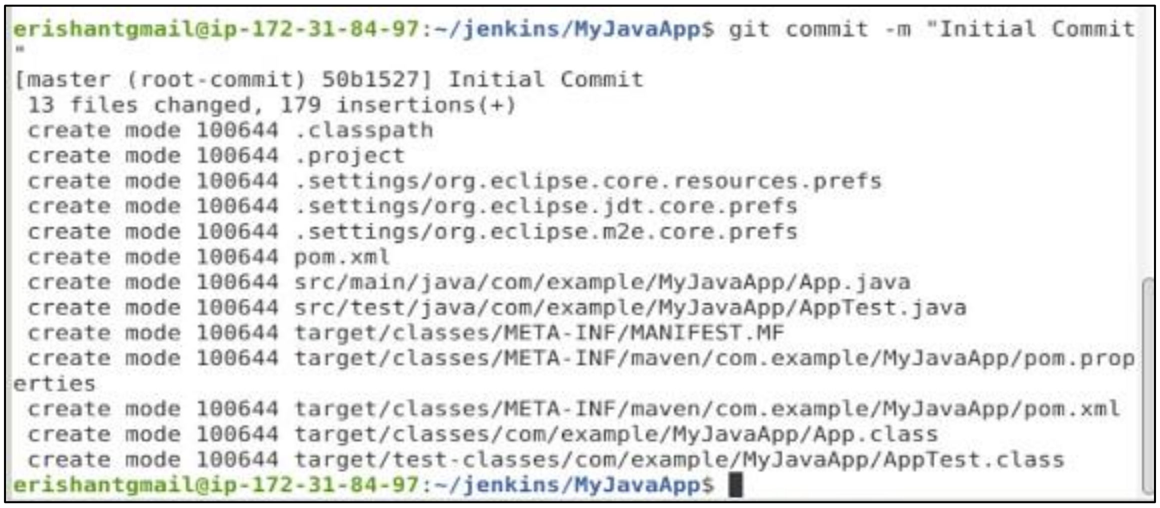
2.5 Enter the following commands to add and commit the project files:

git add .

git commit -m “Initial commit”

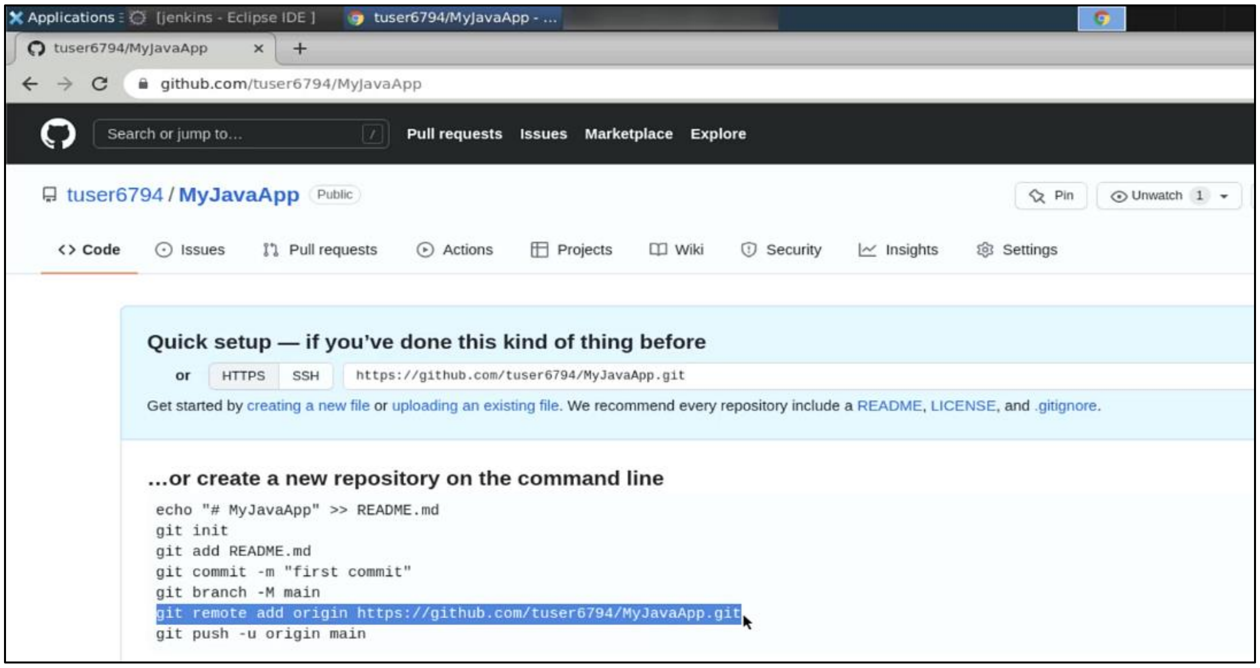

erishantgmail@ip-172-31-84-97: ~/jenkins/MyJavaApp
File Edit View Search Terminal Help

```
erishantgmail@ip-172-31-84-97:~/jenkins/MyJavaApp$ git add .
erishantgmail@ip-172-31-84-97:~/jenkins/MyJavaApp$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   .classpath
        new file:   .project
        new file:   .settings/org.eclipse.core.resources.prefs
        new file:   .settings/org.eclipse.jdt.core.prefs
        new file:   .settings/org.eclipse.m2e.core.prefs
        new file:   pom.xml
        new file:   src/main/java/com/example/MyJavaApp/App.java
        new file:   src/test/java/com/example/MyJavaApp/AppTest.java
        new file:   target/classes/META-INF/MANIFEST.MF
        new file:   target/classes/META-INF/maven/com.example/MyJavaApp/pom.prop
erties
        new file:   target/classes/META-INF/maven/com.example/MyJavaApp/pom.xml
        new file:   target/classes/com/example/MyJavaApp/App.class
        new file:   target/test-classes/com/example/MyJavaApp/AppTest.class

erishantgmail@ip-172-31-84-97:~/jenkins/MyJavaApp$
```



```
erishantgmail@ip-172-31-84-97:~/jenkins/MyJavaApp$ git commit -m "Initial Commit
"
[master (root-commit) 50b1527] Initial Commit
 13 files changed, 179 insertions(+)
 create mode 100644 .classpath
 create mode 100644 .project
 create mode 100644 .settings/org.eclipse.core.resources.prefs
 create mode 100644 .settings/org.eclipse.jdt.core.prefs
 create mode 100644 .settings/org.eclipse.m2e.core.prefs
 create mode 100644 pom.xml
 create mode 100644 src/main/java/com/example/MyJavaApp/App.java
 create mode 100644 src/test/java/com/example/MyJavaApp/AppTest.java
 create mode 100644 target/classes/META-INF/MANIFEST.MF
 create mode 100644 target/classes/META-INF/maven/com.example/MyJavaApp/pom.prop
erties
 create mode 100644 target/classes/META-INF/maven/com.example/MyJavaApp/pom.xml
 create mode 100644 target/classes/com/example/MyJavaApp/App.class
 create mode 100644 target/test-classes/com/example/MyJavaApp/AppTest.class
erishantgmail@ip-172-31-84-97:~/jenkins/MyJavaApp$
```

2.6 Copy the remote URL from **GitHub** and paste it into the terminal to sync the files to the remote repository



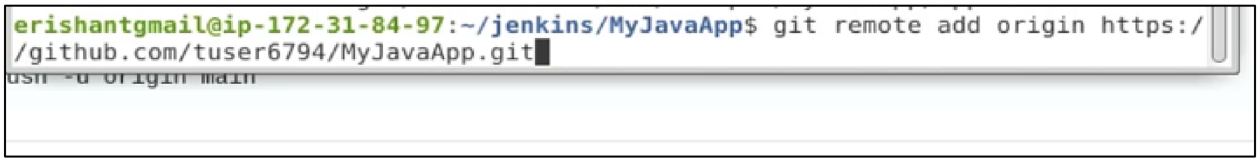
Quick setup — if you've done this kind of thing before

or https://github.com/tuser6794/MyJavaApp.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

…or create a new repository on the command line

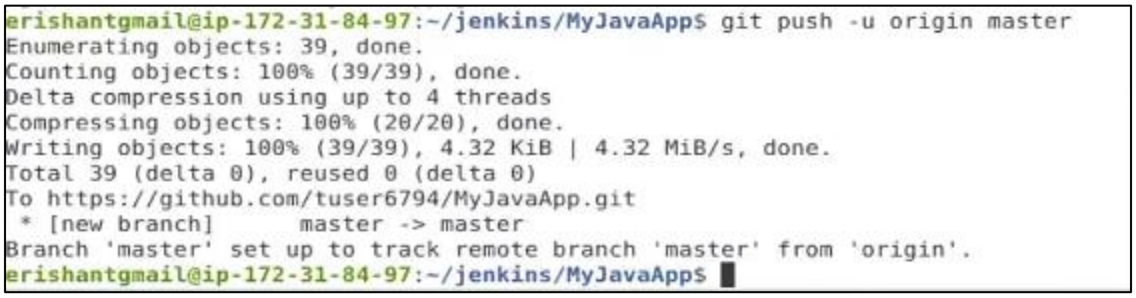
```
echo "# MyJavaApp" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/tuser6794/MyJavaApp.git
git push -u origin main
```



```
erishantgmail@ip-172-31-84-97:~/jenkins/MyJavaApp$ git remote add origin https:/
/github.com/tuser6794/MyJavaApp.git
```

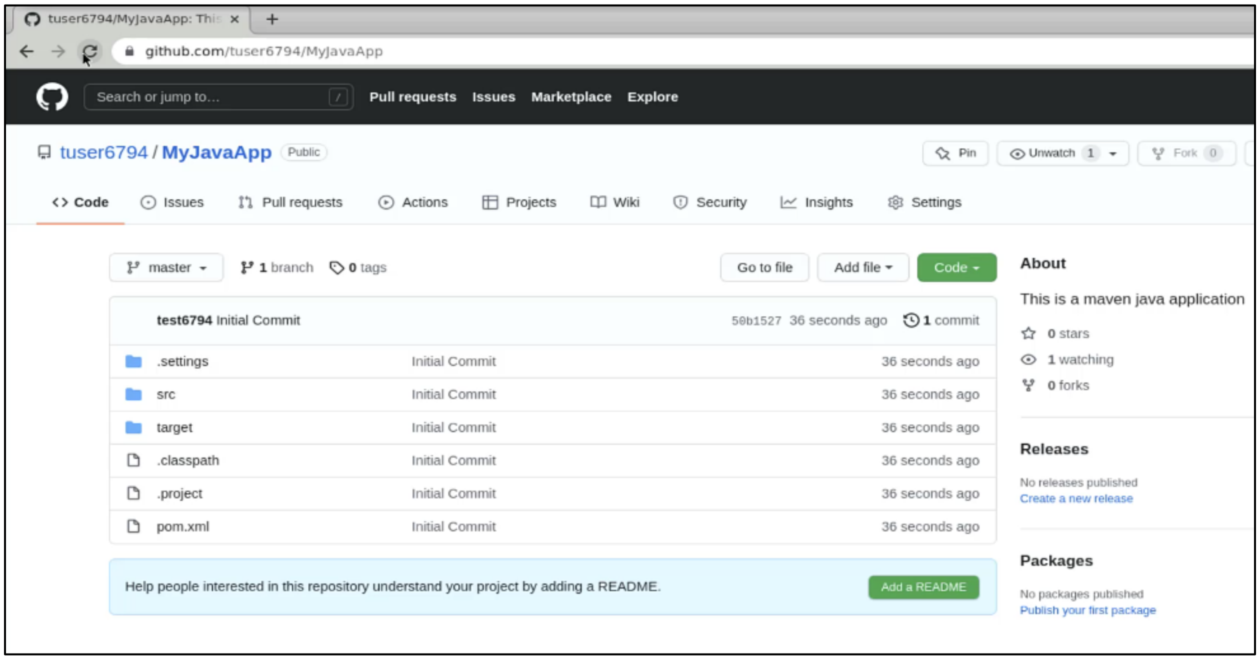
2.7 Enter the following command to complete the syncing process:

git push -u origin master



```
erishantgmail@ip-172-31-84-97:~/jenkins/MyJavaApp$ git push -u origin master
Enumerating objects: 39, done.
Counting objects: 100% (39/39), done.
Delta compression using up to 4 threads
Compressing objects: 100% (20/20), done.
Writing objects: 100% (39/39), 4.32 KiB | 4.32 MiB/s, done.
Total 39 (delta 0), reused 0 (delta 0)
To https://github.com/tuser6794/MyJavaApp.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
erishantgmail@ip-172-31-84-97:~/jenkins/MyJavaApp$
```

2.8 Refresh the GitHub page to see the uploaded project files



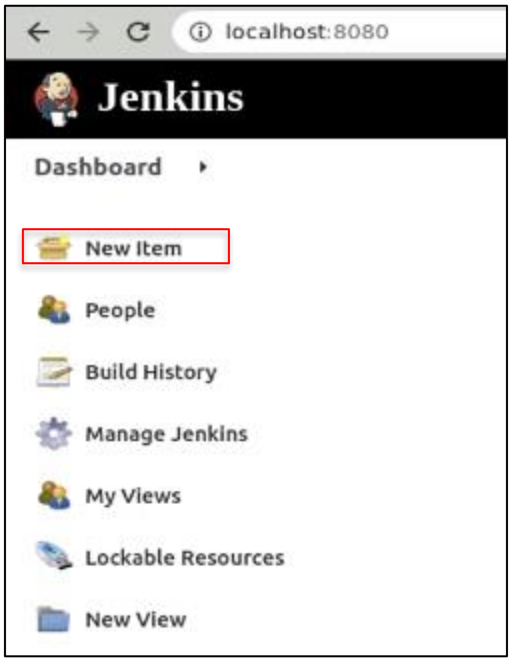
test6794 Initial Commit — 50b1527 36 seconds ago — 1 commit

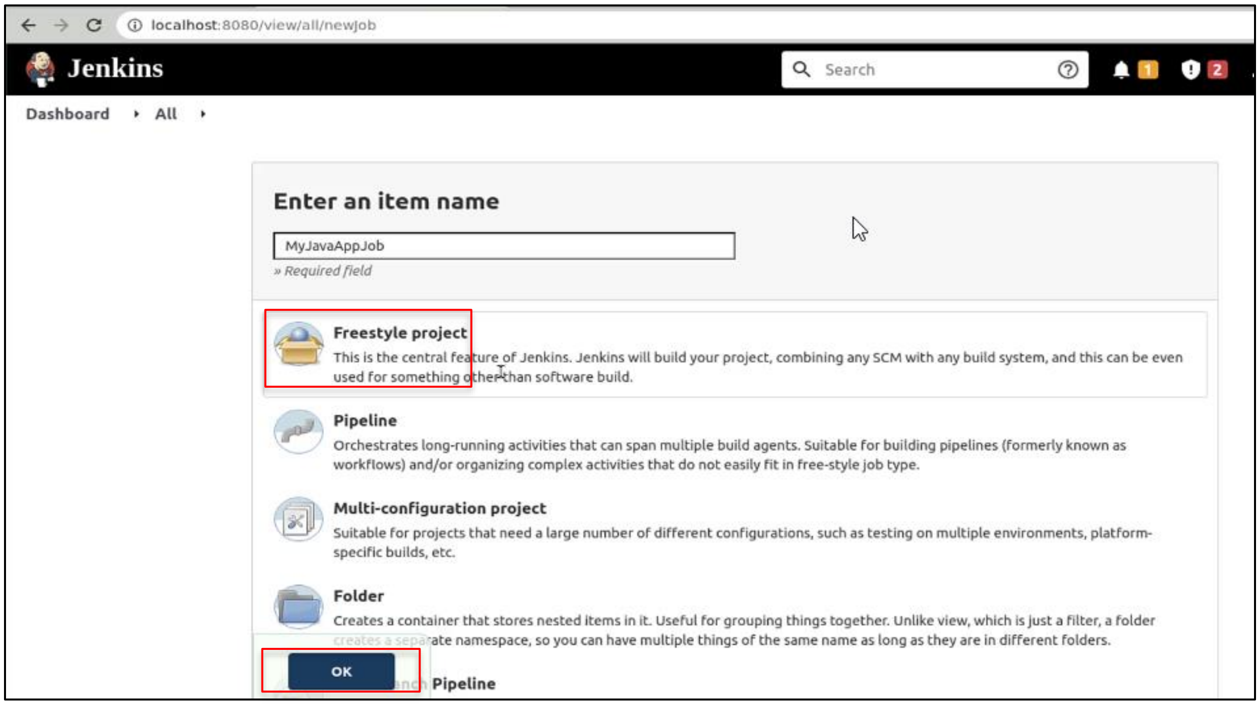
.settings	Initial Commit	36 seconds ago
src	Initial Commit	36 seconds ago
target	Initial Commit	36 seconds ago
.classpath	Initial Commit	36 seconds ago
.project	Initial Commit	36 seconds ago
pom.xml	Initial Commit	36 seconds ago

Help people interested in this repository understand your project by adding a README.

Step 3: Configuring Jenkins with Git

3.1 Go to **Jenkins**, select **New Item**, and create a **Freestyle project** on Jenkins




localhost:8080/view/all/newJob

Jenkins

Dashboard › All ›

Enter an item name

MyJavaAppJob

» 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.

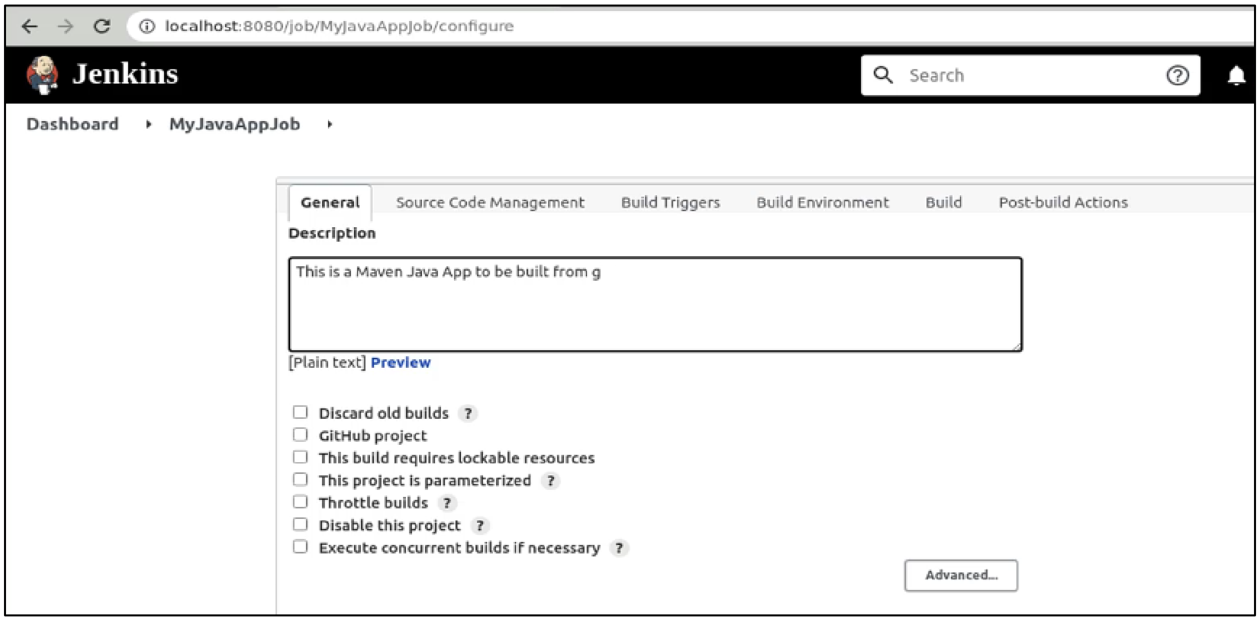
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

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.

OK

3.2 Enter a description under the **General tab**


localhost:8080/job/MyJavaAppJob/configure

Jenkins

Dashboard › MyJavaAppJob ›

General | Source Code Management | Build Triggers | Build Environment | Build | Post-build Actions

Description

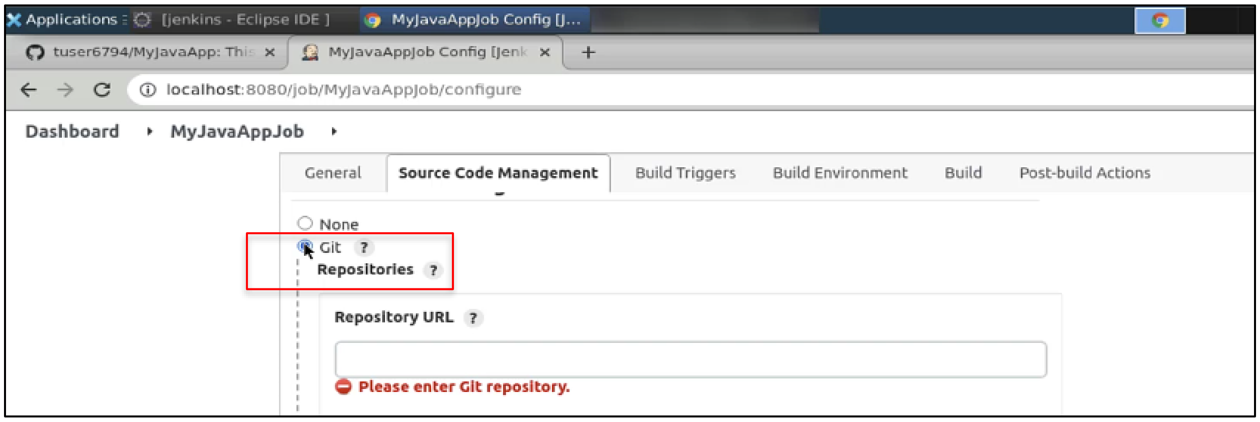
This is a Maven Java App to be built from g

[Plain text] Preview

- ☐ Discard old builds
- ☐ GitHub project
- ☐ This build requires lockable resources
- ☐ This project is parameterized
- ☐ Throttle builds
- ☐ Disable this project
- ☐ Execute concurrent builds if necessary

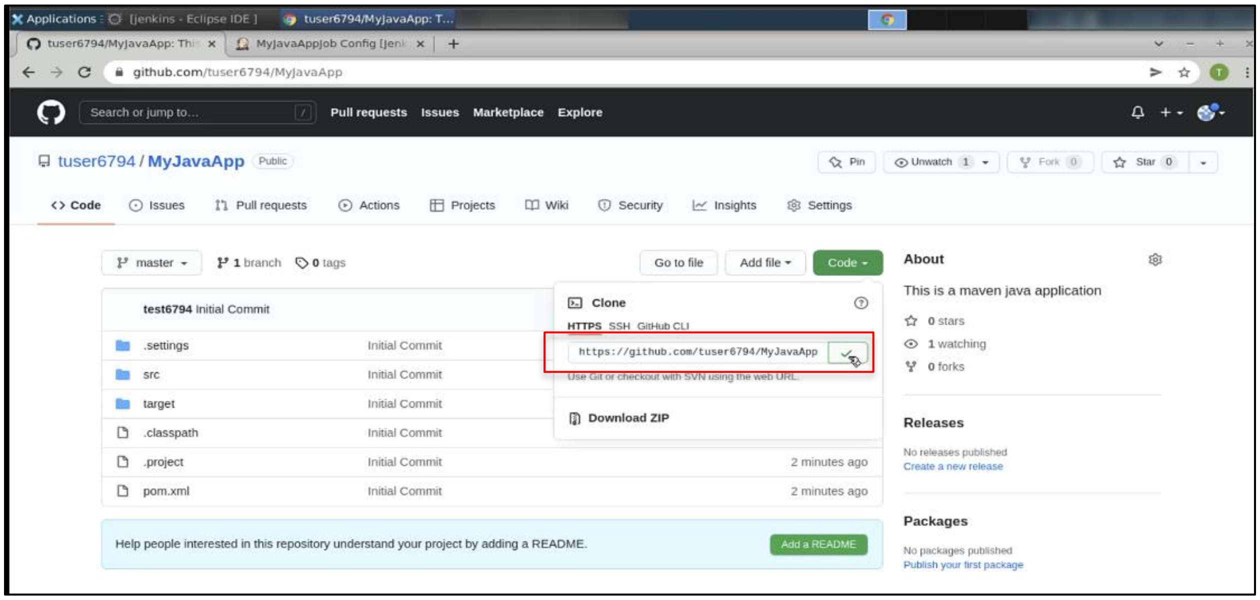
Advanced...

3.3 Go to the **Source Code Management** tab and select **Git**



Please enter Git repository.

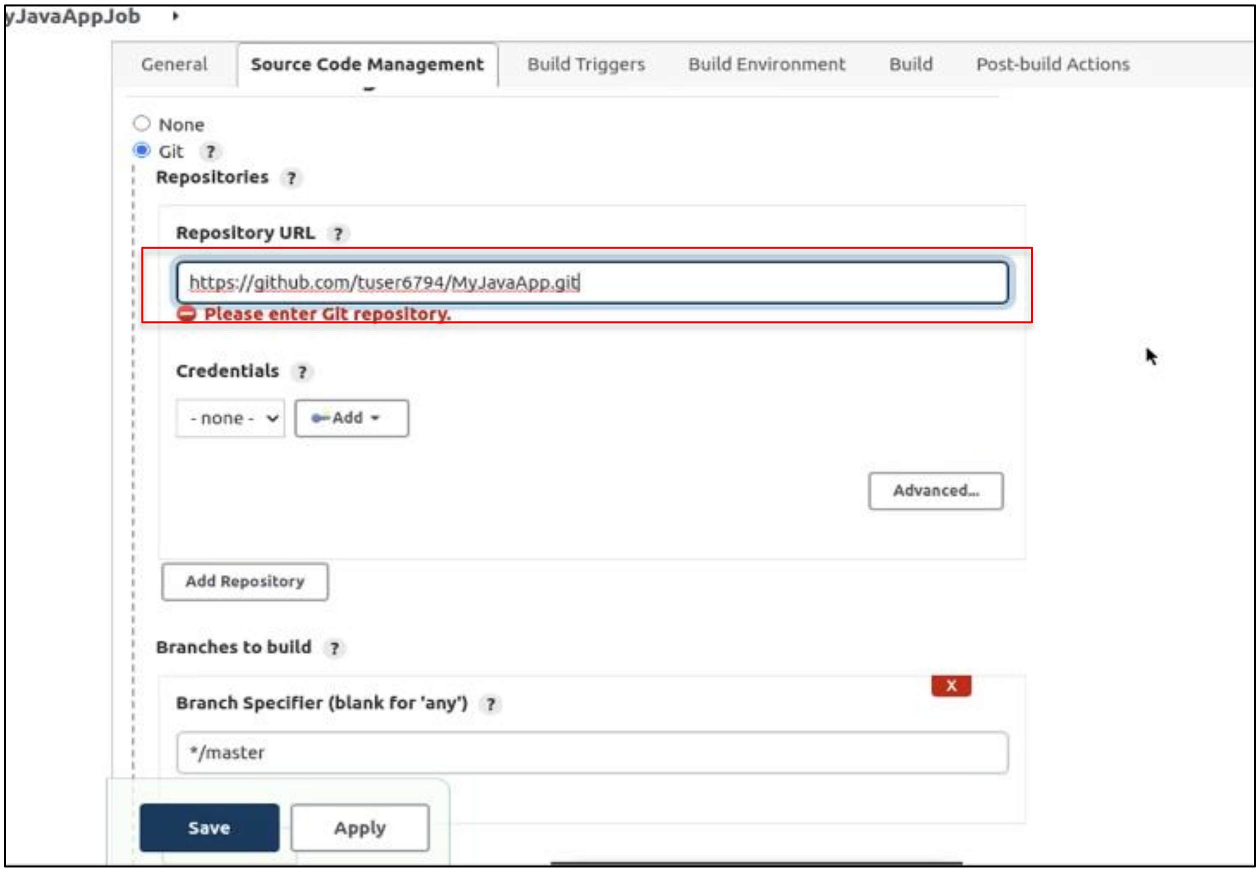
3.4 Go back to **GitHub,** click on the **Code** button, and copy the URL



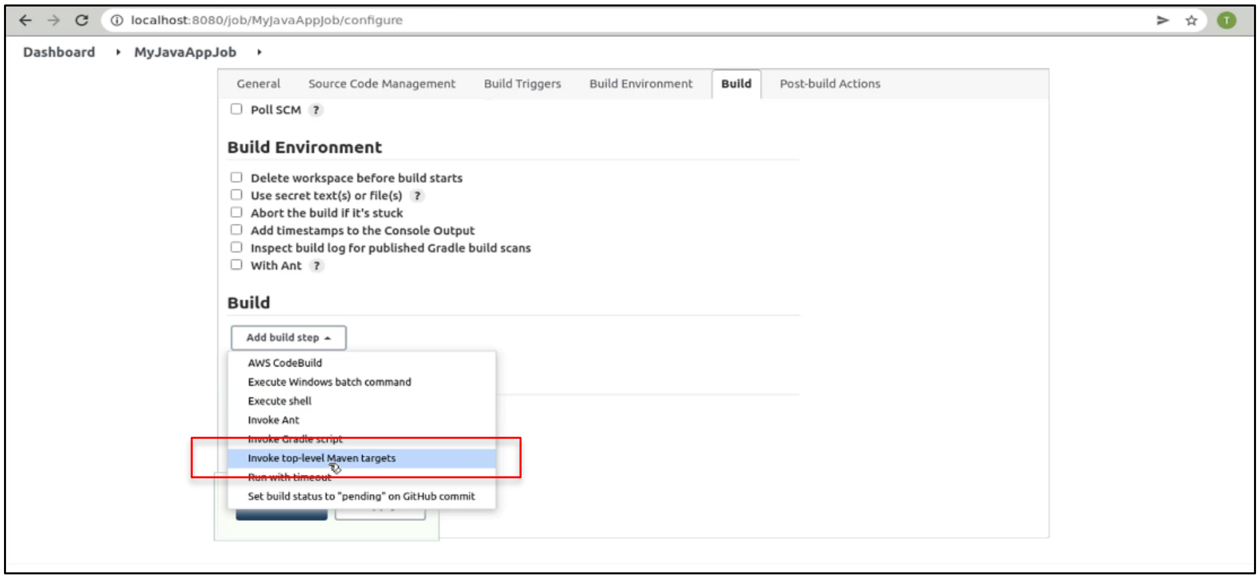
test6794 Initial Commit		
.settings	Initial Commit	
src	Initial Commit	
target	Initial Commit	
.classpath	Initial Commit	
.project	Initial Commit	2 minutes ago
pom.xml	Initial Commit	2 minutes ago

Help people interested in this repository understand your project by adding a README.

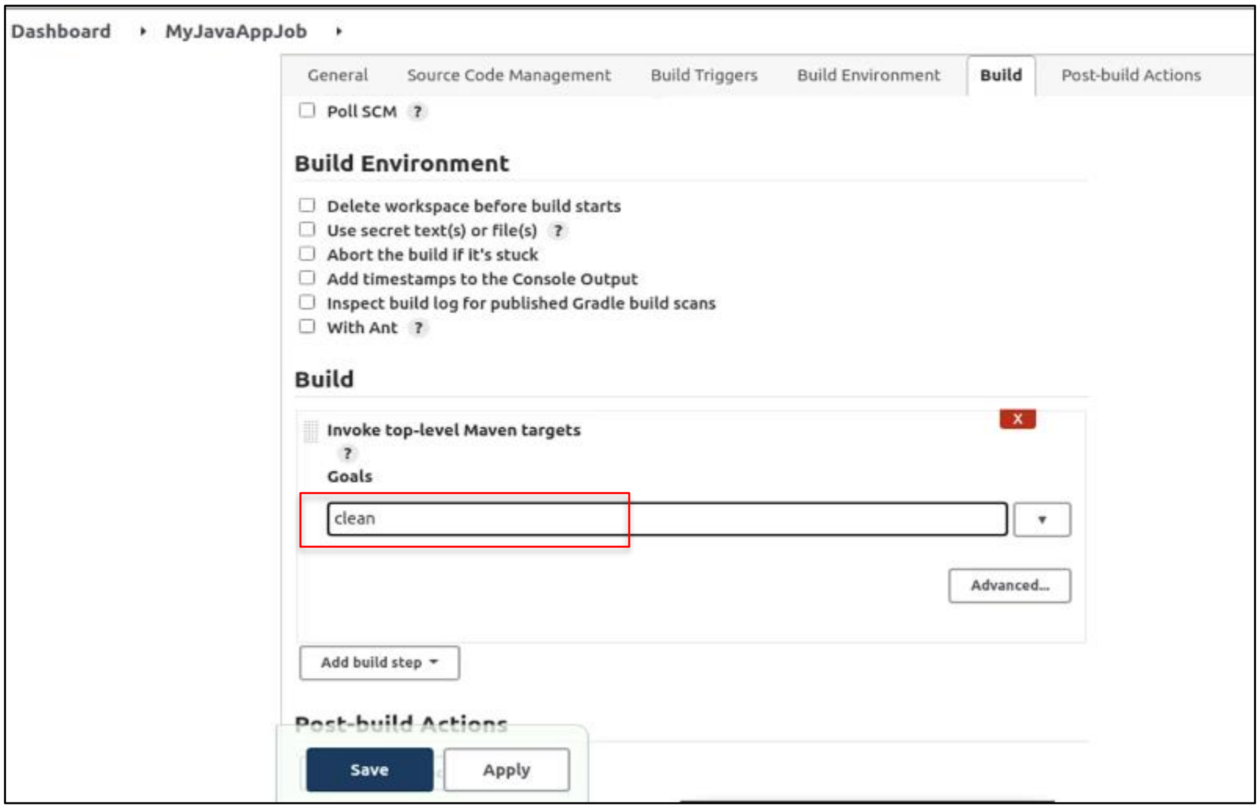
3.5 Paste the URL in the **Repository URL** field



3.6 Go to the **Build** tab and select **Invoke top-level Maven targets** from the dropdown menu


localhost:8080/job/MyJavaAppJob/configure
Dashboard
MyJavaAppJob
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Poll SCM
Build Environment
Delete workspace before build starts
Use secret text(s) or file(s)
Abort the build if it's stuck
Add timestamps to the Console Output
Inspect build log for published Gradle build scans
With Ant
Build
Add build step
AWS CodeBuild
Execute Windows batch command
Execute shell
Invoke Ant
Invoke Gradle script
Invoke top-level Maven targets
Run with timeout
Set build status to "pending" on GitHub commit

3.7 Enter **clean** in the **Goals** field


Dashboard › MyJavaAppJob ›

Poll SCM

Build Environment

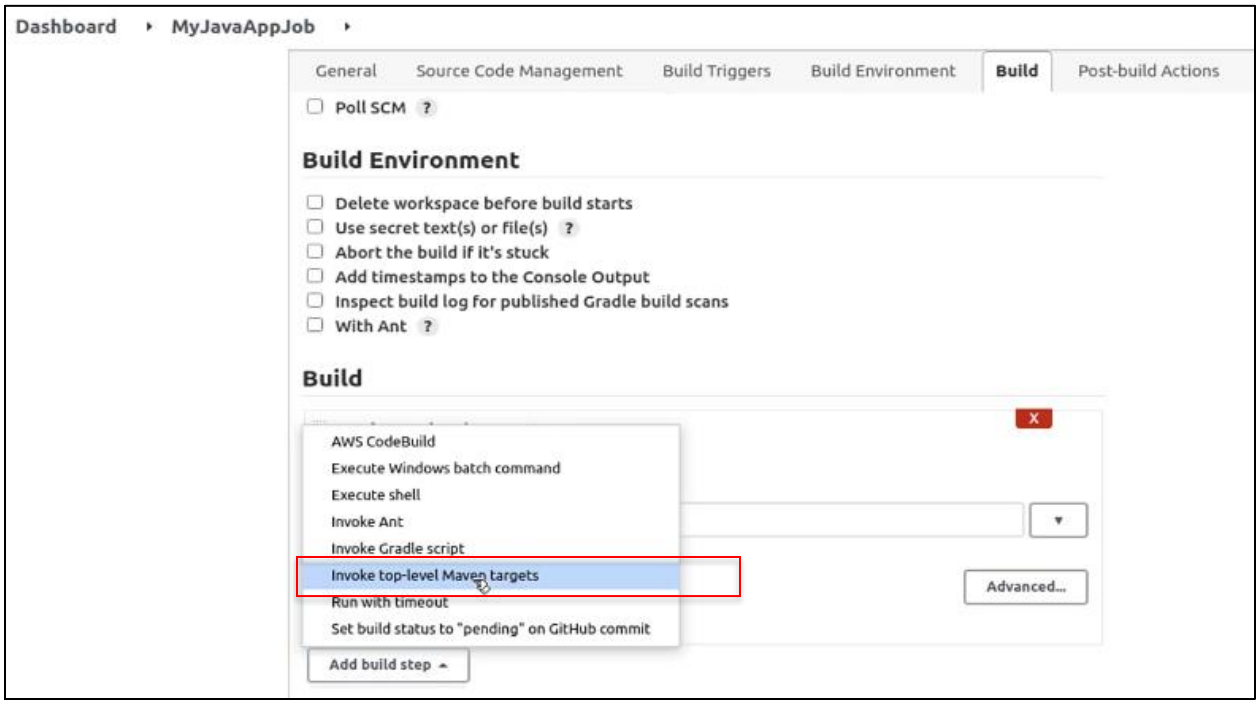
- Delete workspace before build starts
- Use secret text(s) or file(s)
- Abort the build if it's stuck
- Add timestamps to the Console Output
- Inspect build log for published Gradle build scans
- With Ant

Build

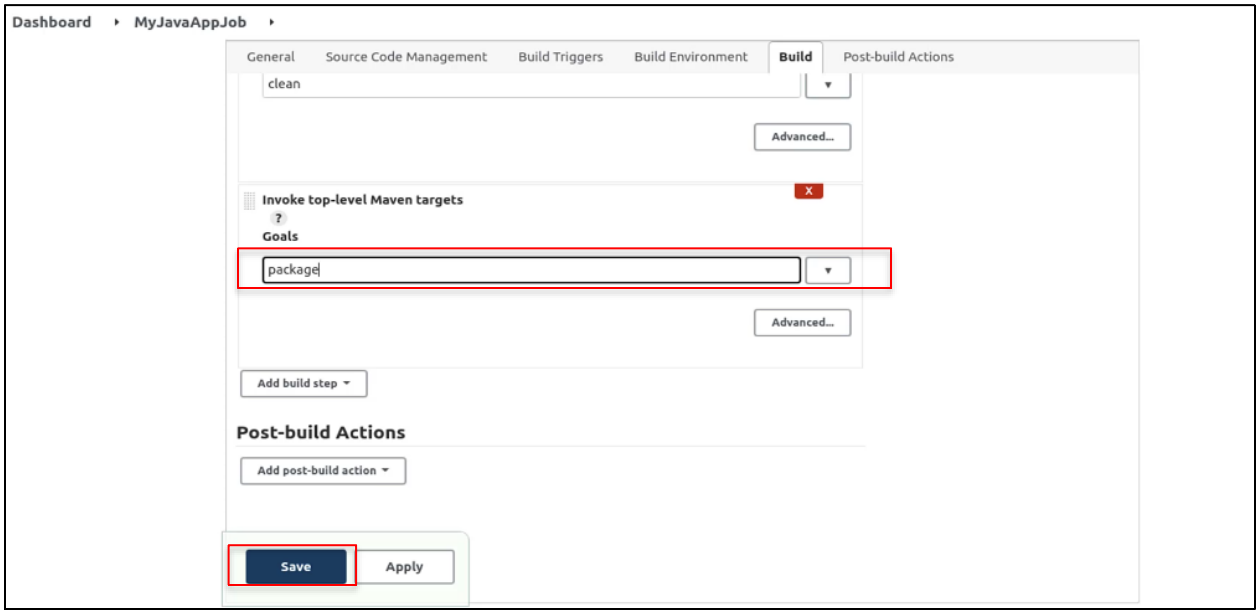
Invoke top-level Maven targets

Goals: clean

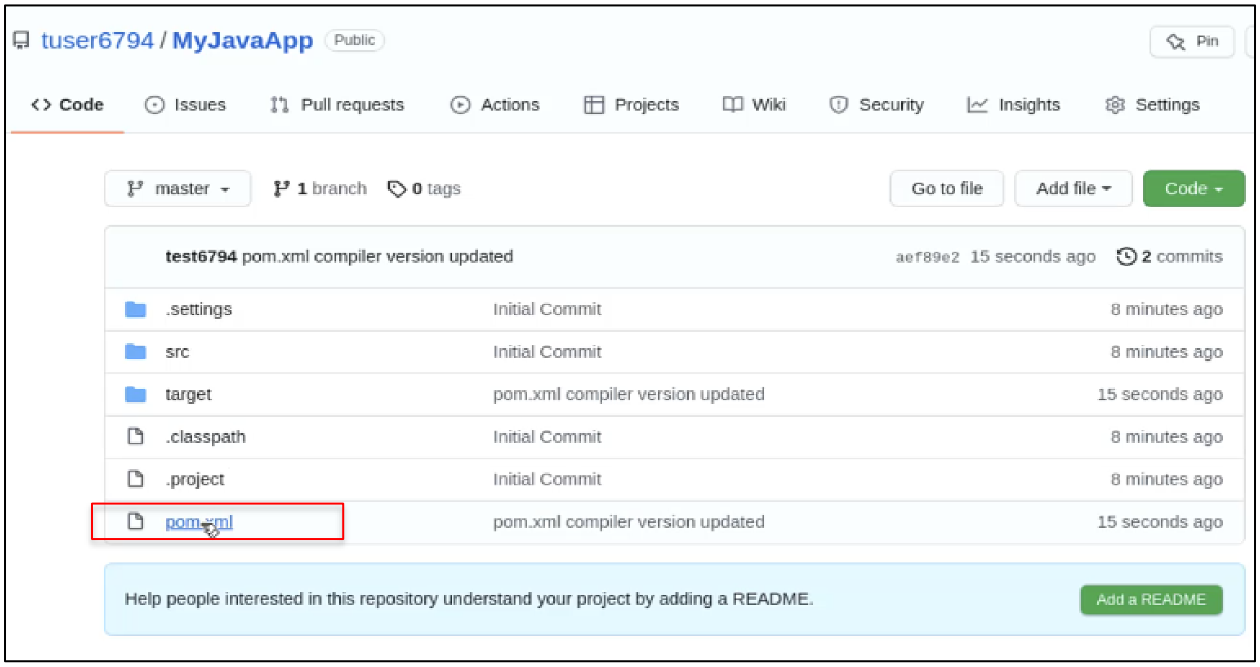
3.8 Click on the **Build** tab again and select **Invoke top-level Maven targets** from the dropdown menu


Dashboard
MyJavaAppJob
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Poll SCM
Build Environment
Delete workspace before build starts
Use secret text(s) or file(s)
Abort the build if it's stuck
Add timestamps to the Console Output
Inspect build log for published Gradle build scans
With Ant
Build
AWS CodeBuild
Execute Windows batch command
Execute shell
Invoke Ant
Invoke Gradle script
Invoke top-level Maven targets
Run with timeout
Set build status to "pending" on GitHub commit
Advanced...
Add build step

3.9 Enter **package** in the **Goals** field and click on the **Save** button



3.10 In the **GitHub** repository, open the **pom.xml** file in the root directory



tuser6794 / MyJavaApp Public

<> Code · Issues · Pull requests · Actions · Projects · Wiki · Security · Insights · Settings

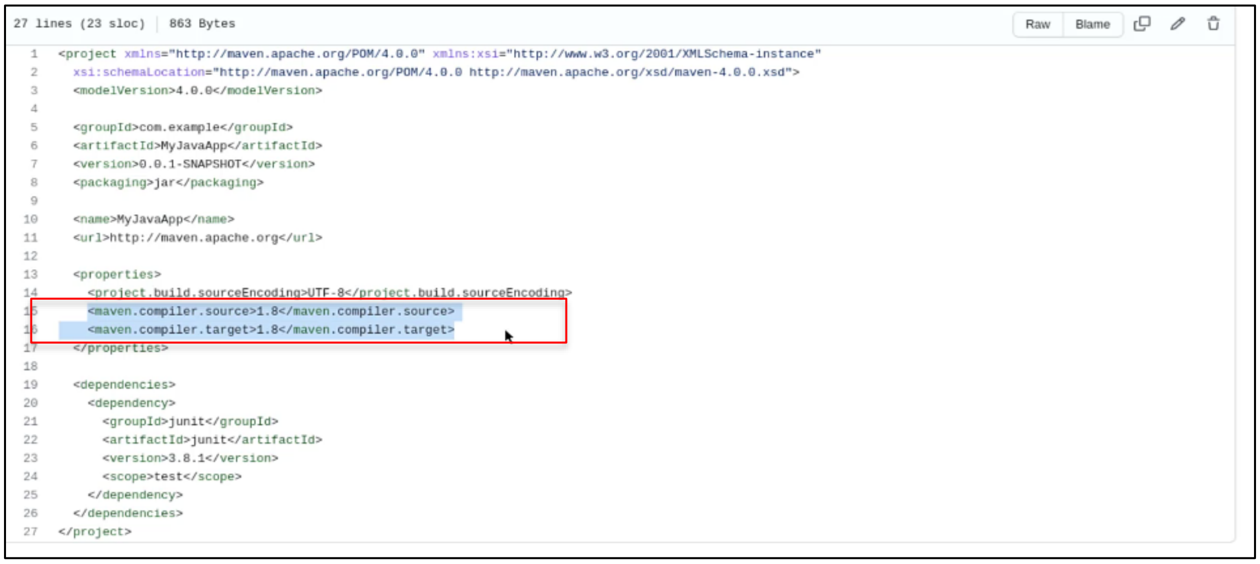
master · 1 branch · 0 tags · Go to file · Add file · Code

test6794 pom.xml compiler version updated · aef89e2 15 seconds ago · 2 commits

.settings	Initial Commit	8 minutes ago
src	Initial Commit	8 minutes ago
target	pom.xml compiler version updated	15 seconds ago
.classpath	Initial Commit	8 minutes ago
.project	Initial Commit	8 minutes ago
pom.xml	pom.xml compiler version updated	15 seconds ago

Help people interested in this repository understand your project by adding a README. Add a README

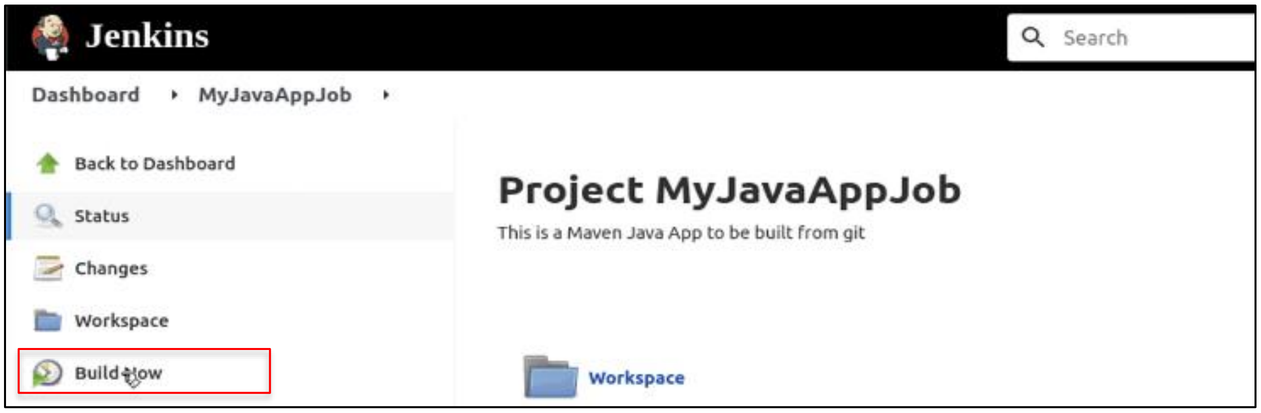
3.11 Update the following highlighted code and save the file:



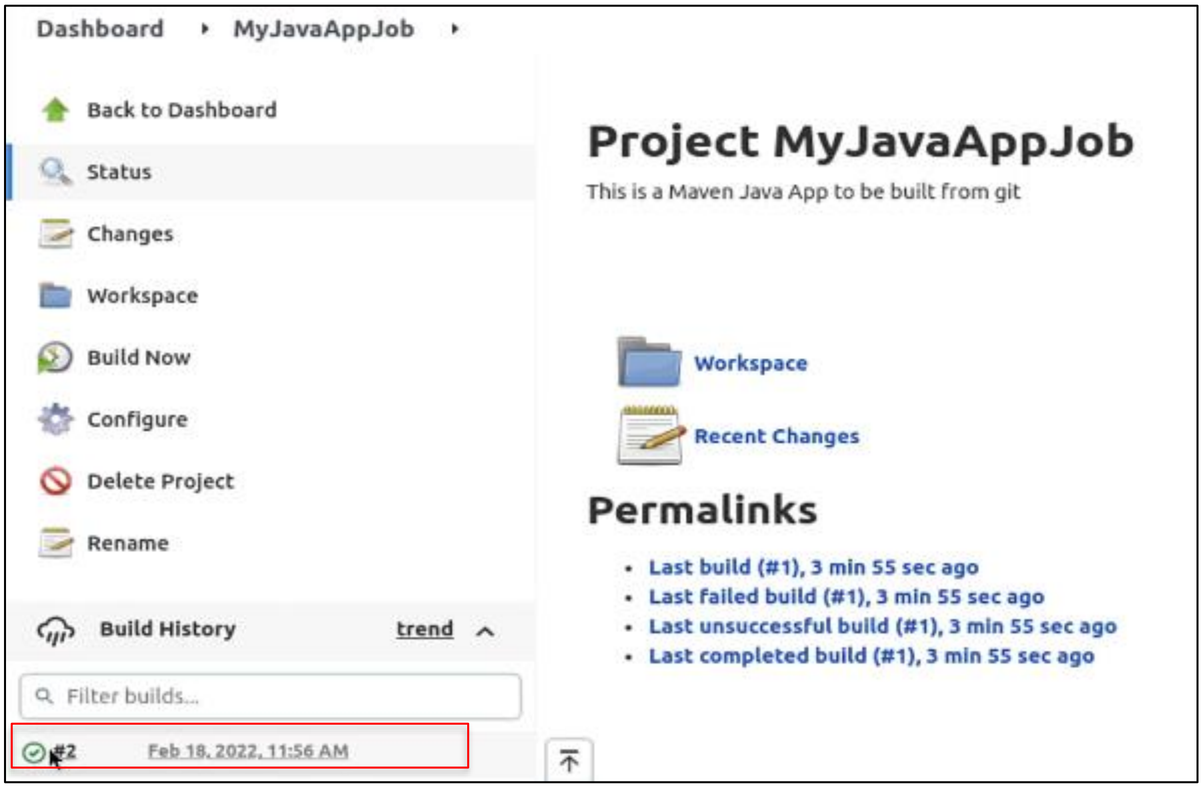
27 lines (23 sloc) | 863 Bytes · Raw · Blame

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>
  <artifactId>MyJavaApp</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>MyJavaApp</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

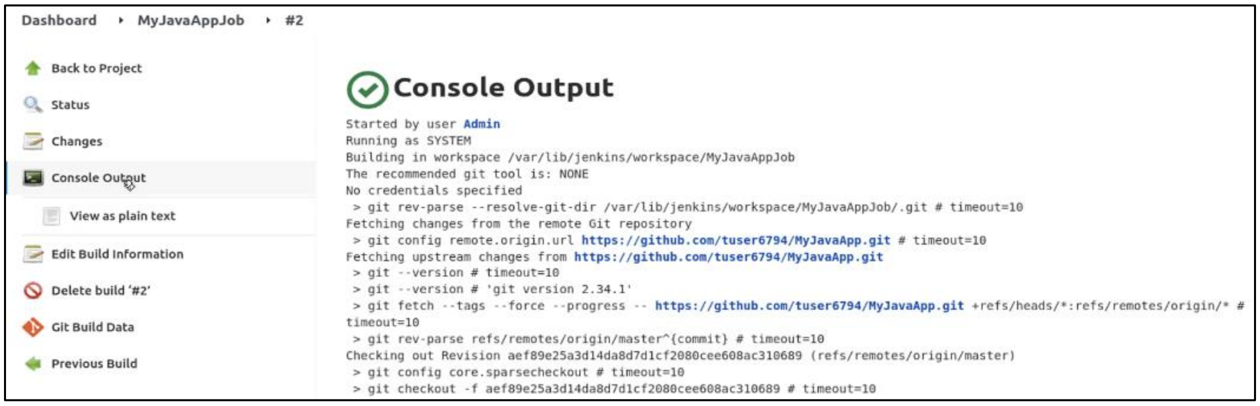
3.12 Go to Jenkins and select the **Build Now** option from the dashboard



3.13 Once the build process is complete, select the build from the **Build History** section



3.14 Click on the **console output** option to view the output of the build in detail



```
Started by user Admin
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/MyJavaAppJob
The recommended git tool is: NONE
No credentials specified
 > git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/MyJavaAppJob/.git # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/tuser6794/MyJavaApp.git # timeout=10
Fetching upstream changes from https://github.com/tuser6794/MyJavaApp.git
 > git --version # timeout=10
 > git --version # 'git version 2.34.1'
 > git fetch --tags --force --progress -- https://github.com/tuser6794/MyJavaApp.git +refs/heads/*:refs/remotes/origin/* #
timeout=10
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision aef89e25a3d14da8d7d1cf2080cee608ac310689 (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f aef89e25a3d14da8d7d1cf2080cee608ac310689 # timeout=10
```

You have successfully synchronized the Maven project with Git and configured Jenkins to build the project using SCM. This enables you to automate the build process and ensure a streamlined development workflow.