The project can be found at: <https://github.com/ciucacosmin109/SQMA_Ciuca_Marius_Cosmin.git>

# Jenkins setup

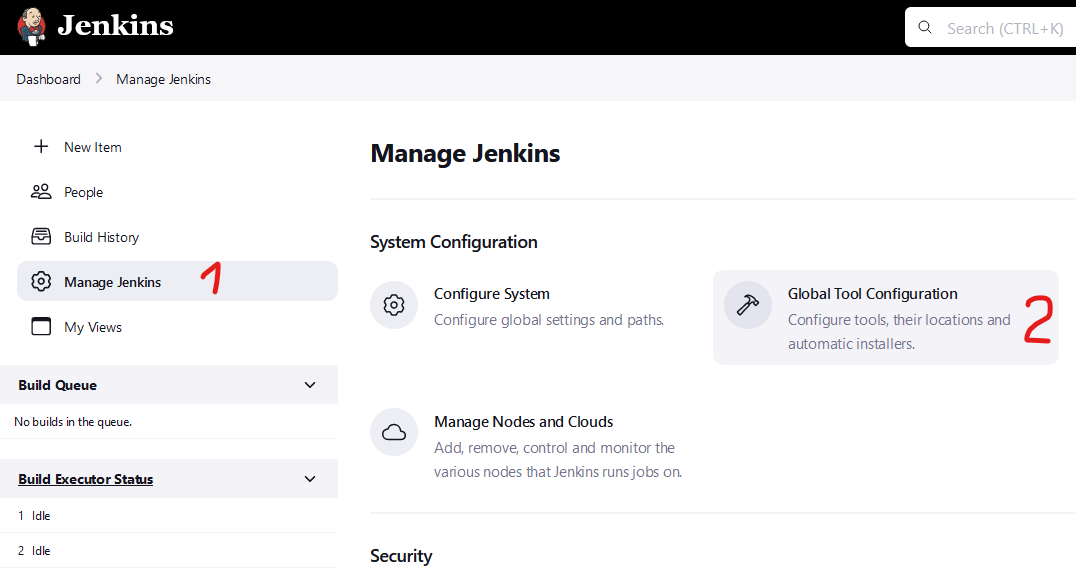
## 1. Run Jenkins

1. Install Docker
2. Run ‘docker run -p 8080:8080 -p 50000:50000 -v jenkins\_home:/var/jenkins\_home jenkins/jenkins:lts-jdk11’

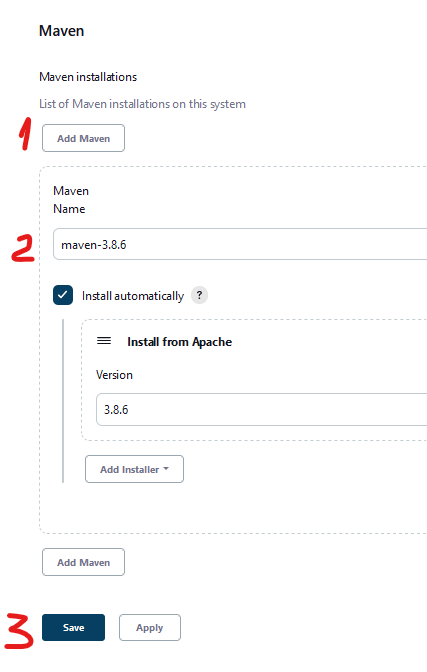
## 2. Add a Maven installation to Jenkins

To run the Junit unit tests, the pipeline will use Maven. For Jenkins to be able to run the ‘mvn’ command, we must add a Maven installation and specify Maven as a tool required for the pipeline (in the ‘Jenkinsfile’ file).

From the dashboard, click on ‘Manage Jenkins’ and then click on ‘Global Tool Configuration’



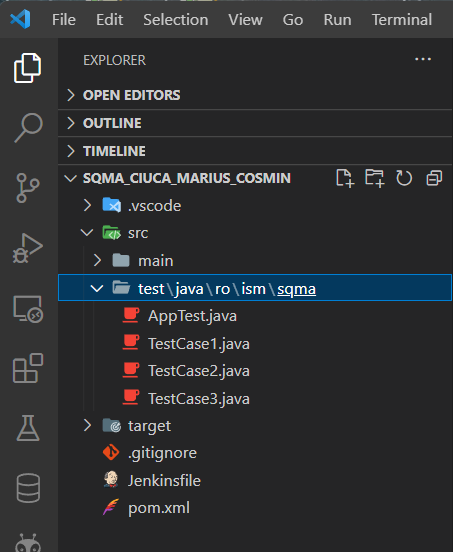
On the ‘Maven’ category, add a new maven installation:



IMPORTANT: Make sure you have the same name as the one defined in ‘Jenkinsfile’

# Java project

For this homework I have created a java project using Maven and created 3 test classes (TestCase1 and TestCase2 should pass and ThestCase3 should fail)



The project can be found at: <https://github.com/ciucacosmin109/SQMA_Ciuca_Marius_Cosmin.git>

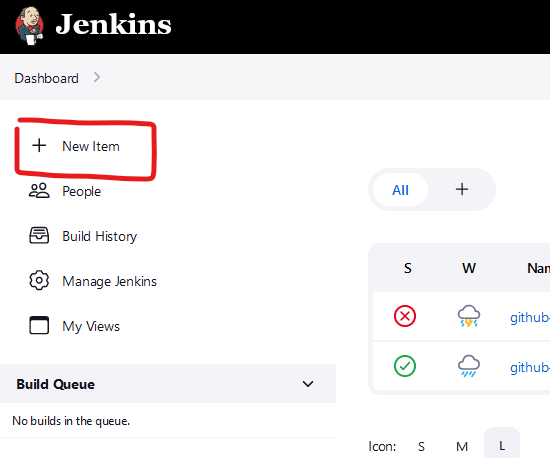
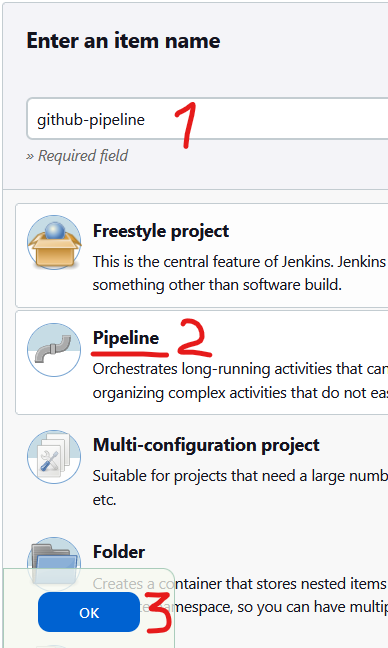
# Homework 1

Create a Jenkins job that connect to a GitHub repository where you have minimum 2 tests.

The Jenkins user should decide by a parameter which one of these tests to run. (you can have the tests in different TestCases).

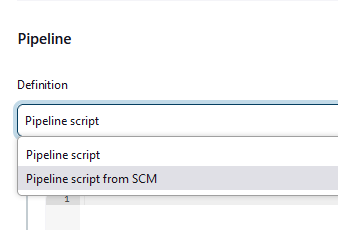
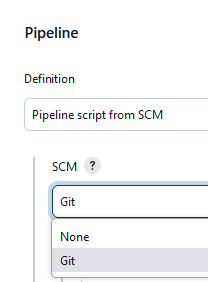
## 1. Create the job

From the dashboard click on ‘New Item’, choose a name, and then choose the ‘Pipeline’ option.

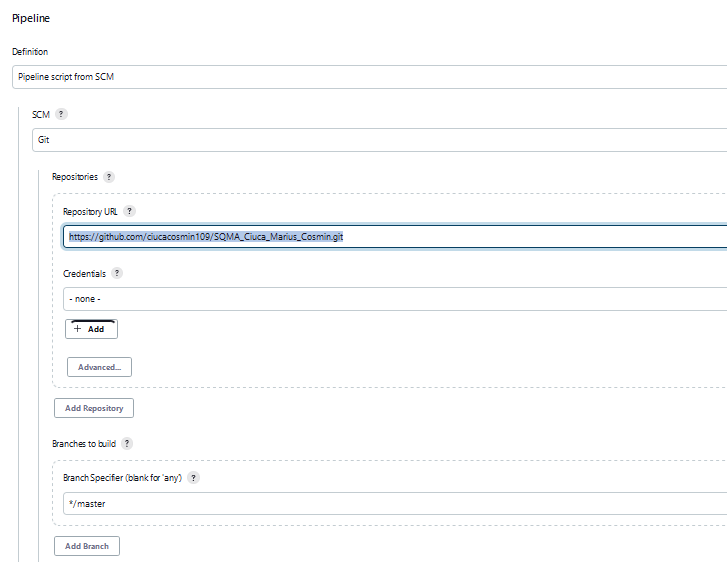
 

## 2. Configure the git repository to pull from

From the ‘Pipeline’ category, choose ‘Pipeline script from SCM’ and the select ‘Git’ as the SCM

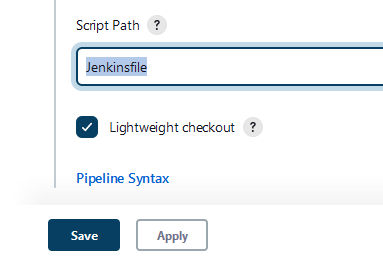
 

Paste your git remote link and choose the master branch

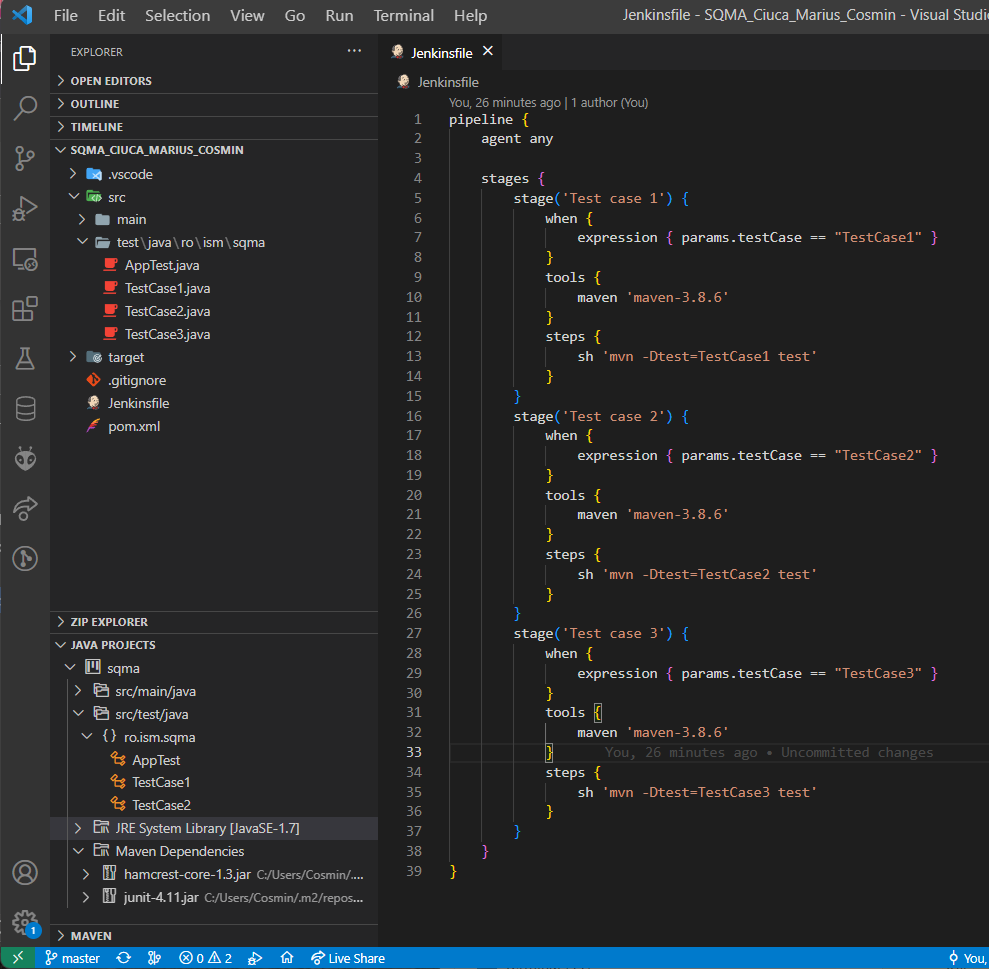


## 3. Set what the job should do

From the pipeline category, choose what script file from the repository should be executed



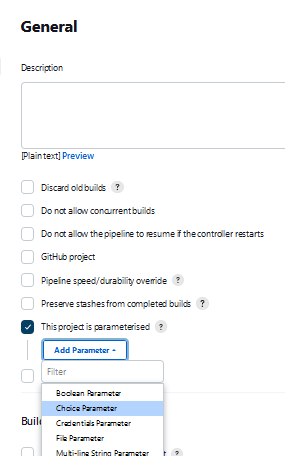
Create the ‘Jenkinsfile’ file



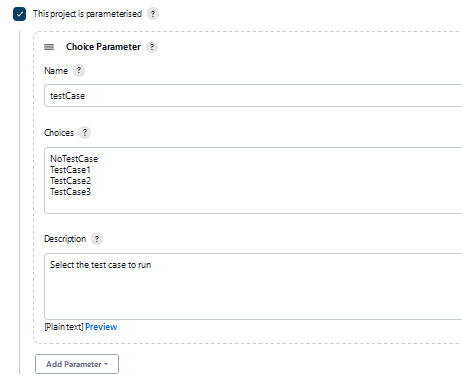
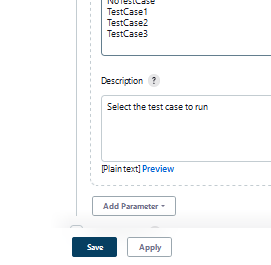
The file has 3 stages, one for each test case, that will run based on a condition set on the parameter testcase, defined at the step 4.

## 4. Creating a parameter for the job

From the ‘General’ category, enable the parameters for the project and create a new choice parameter



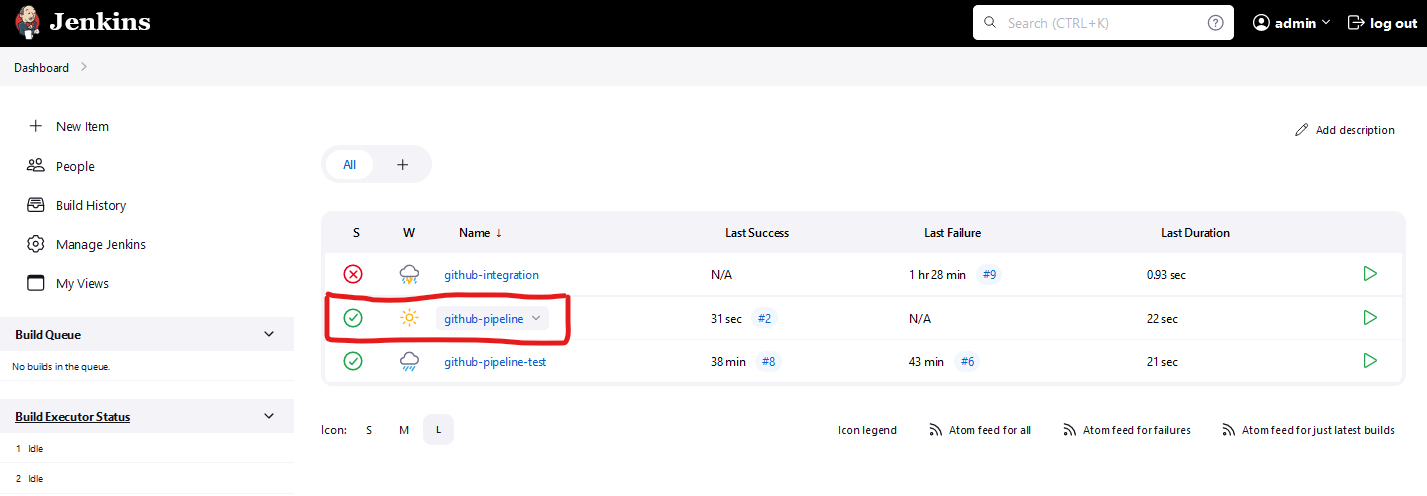
Enter a name that will be used to identify this parameter and a list of choices

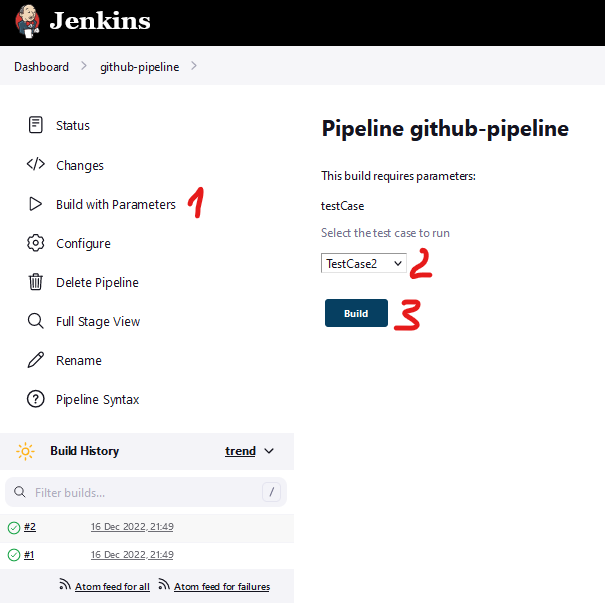
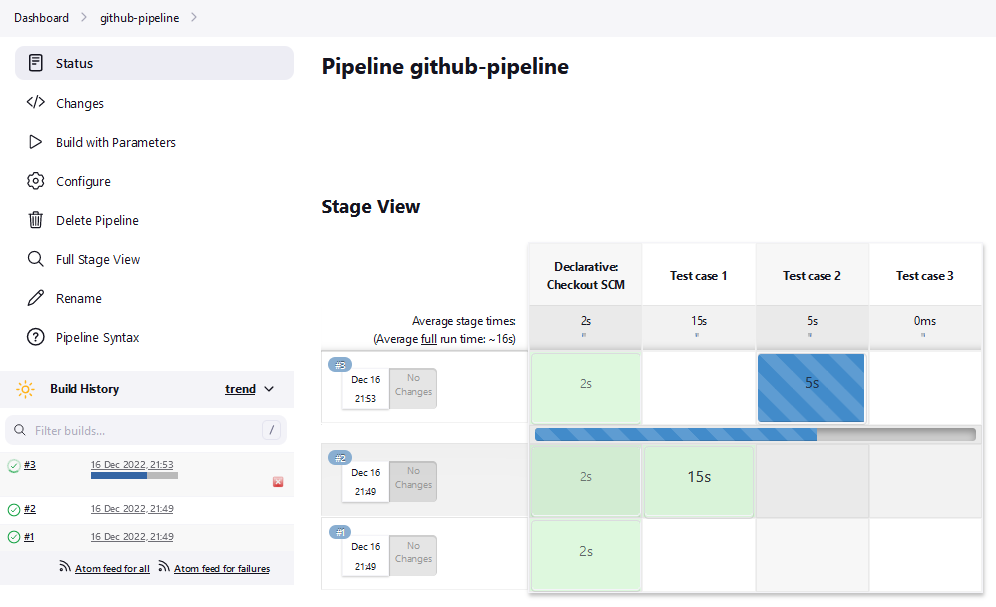
Save the job

## 6. Test the job

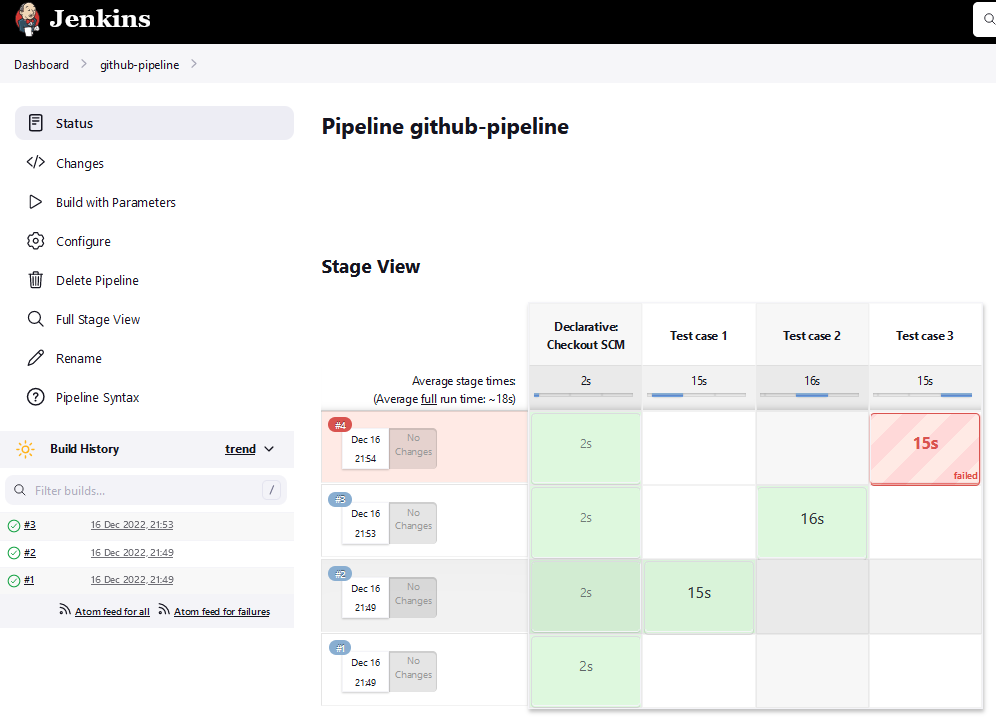
From the main page, select the newly created job:



Choose the value for the parameter and run the pipeline:

TestCase3 should fail:

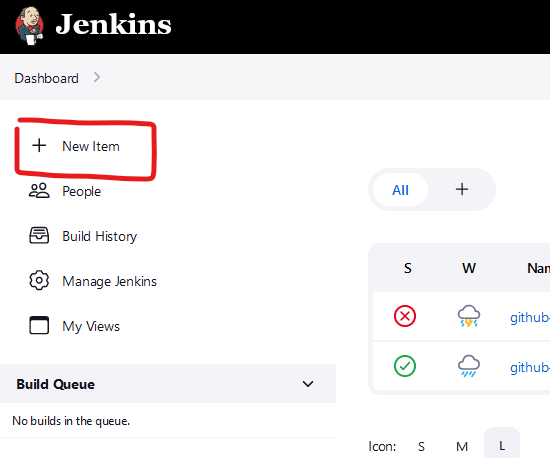
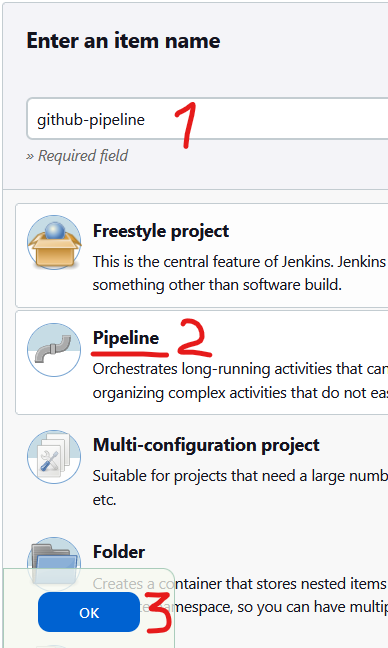


# Homework 2

You have to create more jobs in a pipeline that will run all tests from your repository.

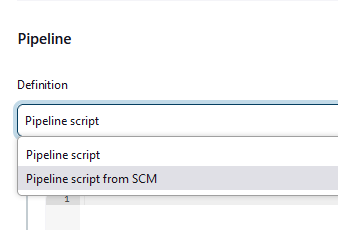
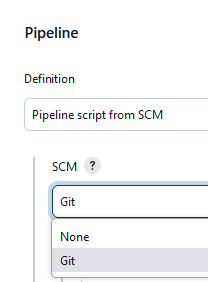
## 1. Create the pipeline job

From the dashboard click on ‘New Item’, choose a name, and then choose the ‘Pipeline’ option.

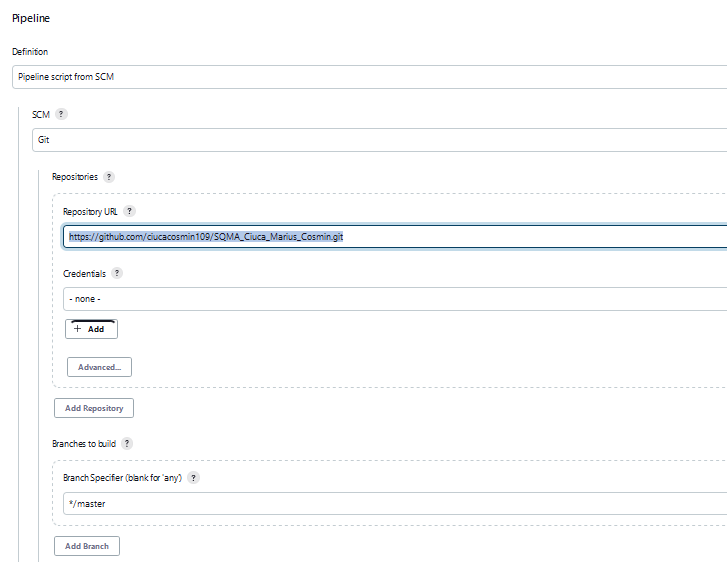
 

## 2. Configure the git repository to pull from

From the ‘Pipeline’ category, choose ‘Pipeline script from SCM’ and the select ‘Git’ as the SCM

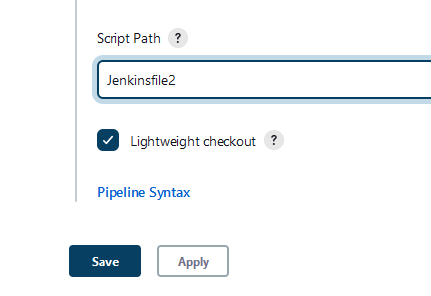
 

Paste your git remote link and choose the master branch

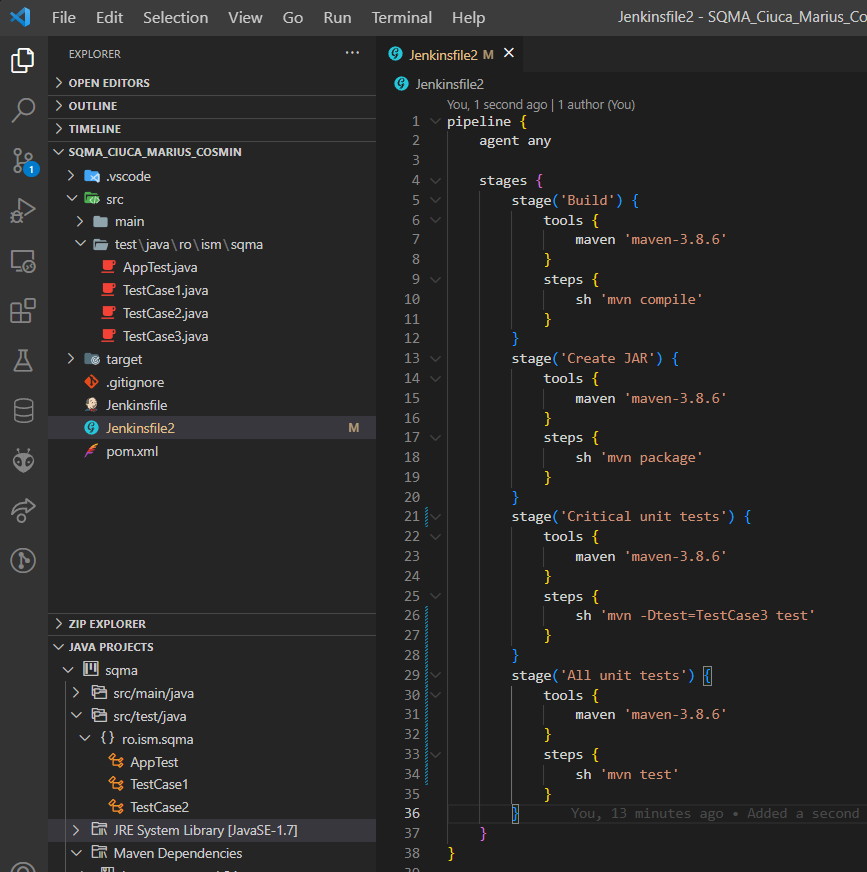


## 3. Set what the job should do

From the pipeline category, choose what script file from the repository should be executed



Create the ‘Jenkinsfile2’ file



The file has 4 stages:

* The build stage
* A stage that will create a .jar file from the solution
* A stage to run the TestCase3
* The Unit tests stage that will run all the tests

For this pipeline, I will change the TestCase3 to pass.

## 4. Testing the pipeline

