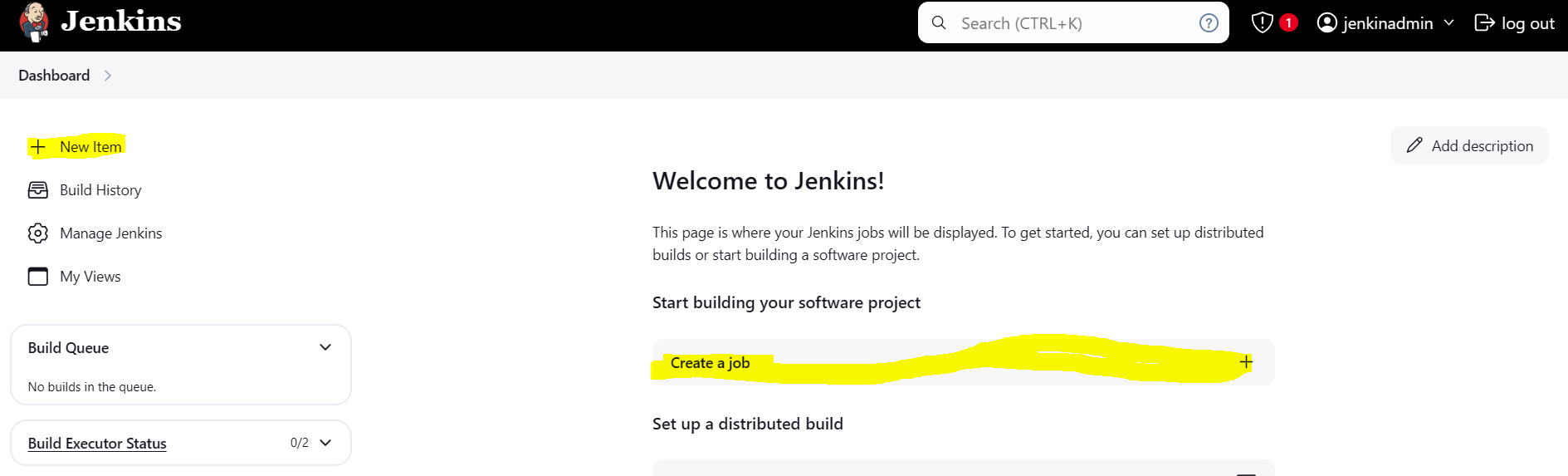
CREATE JENKINS JOB

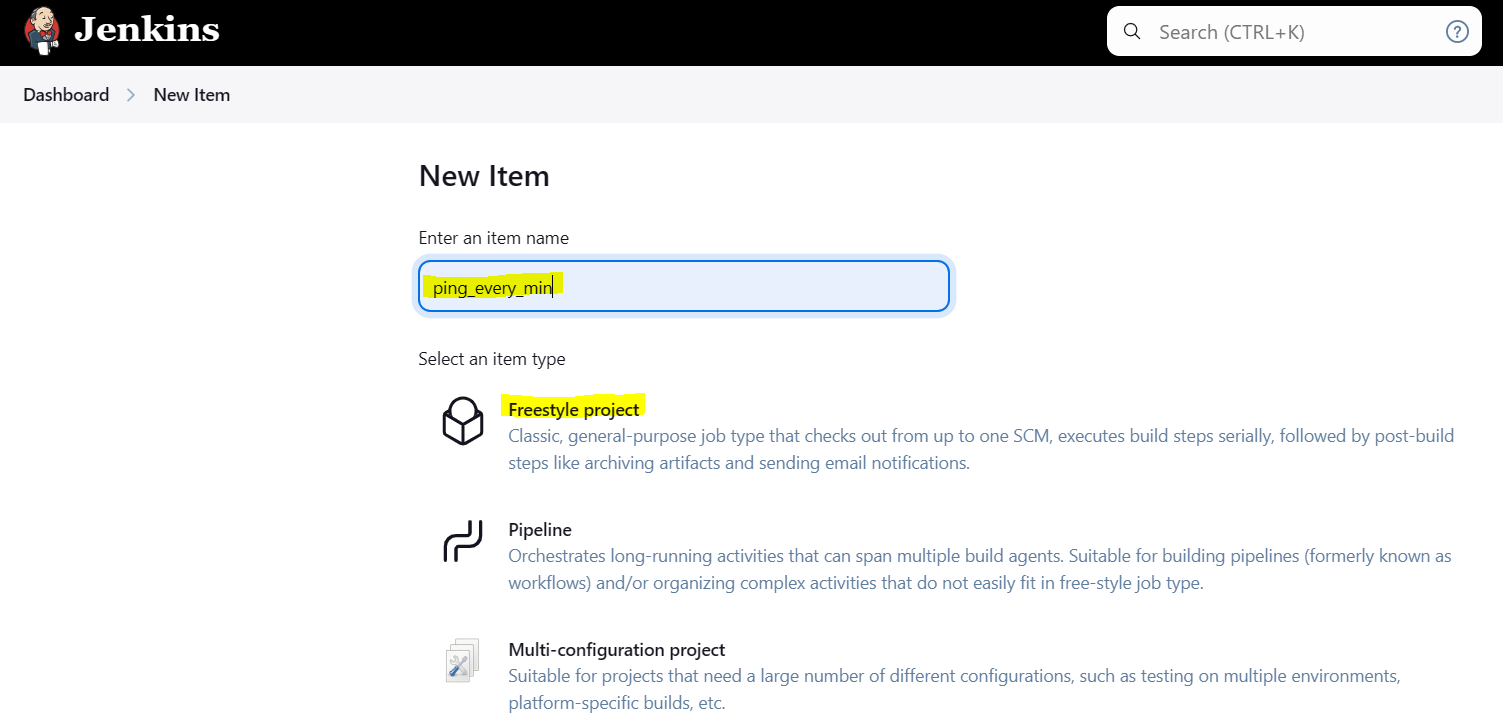
Jenkins is an open-source automation server and Continues Integration Server.

Step Build periodically job:

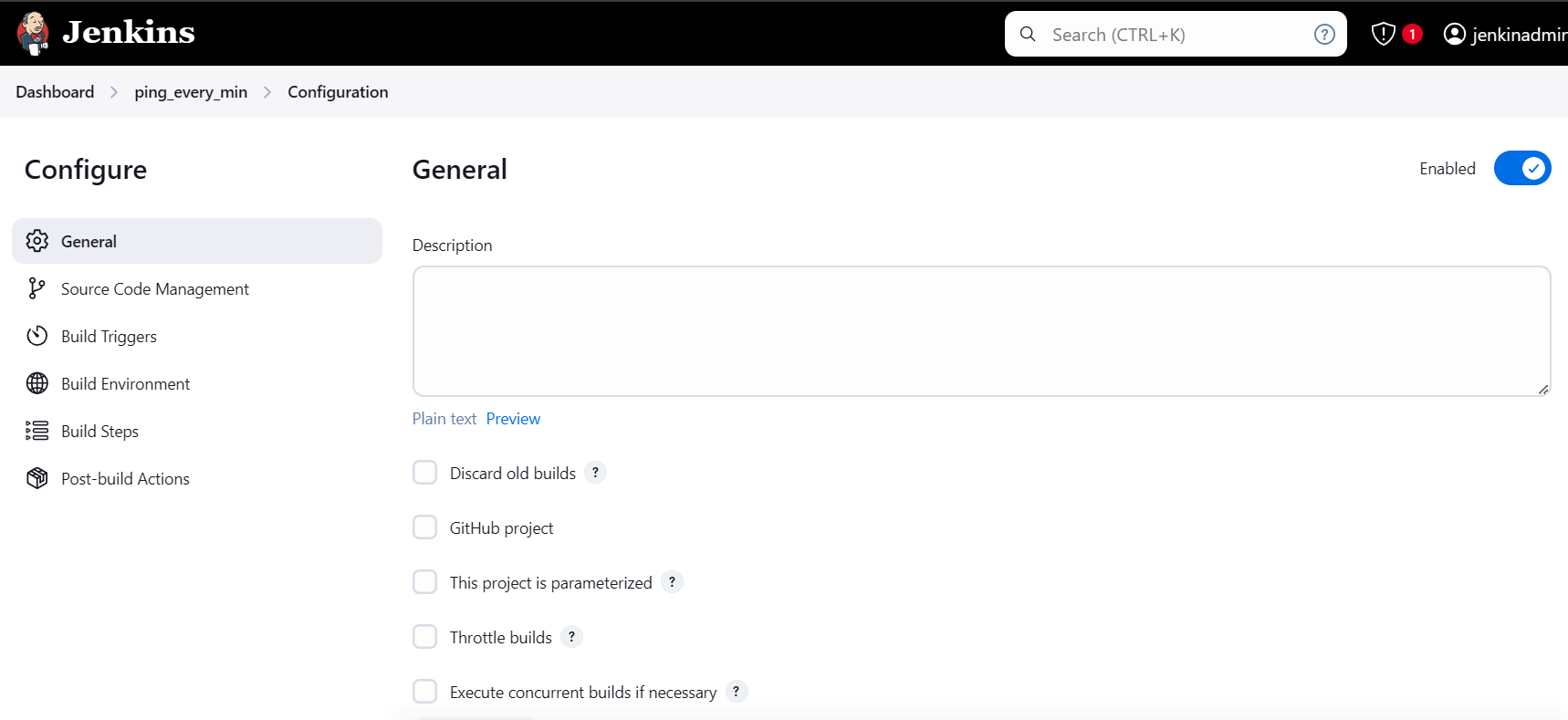
Click on +New Item or Click on Create job as per below screen shot.

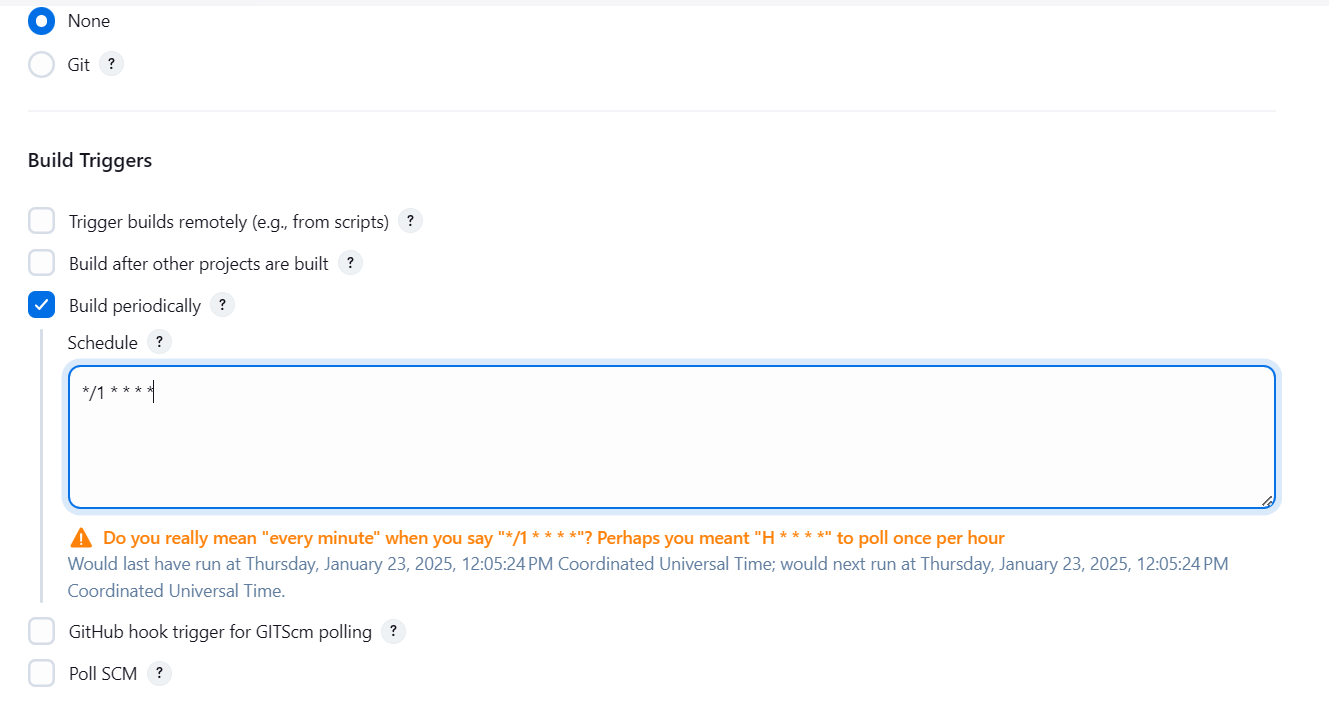


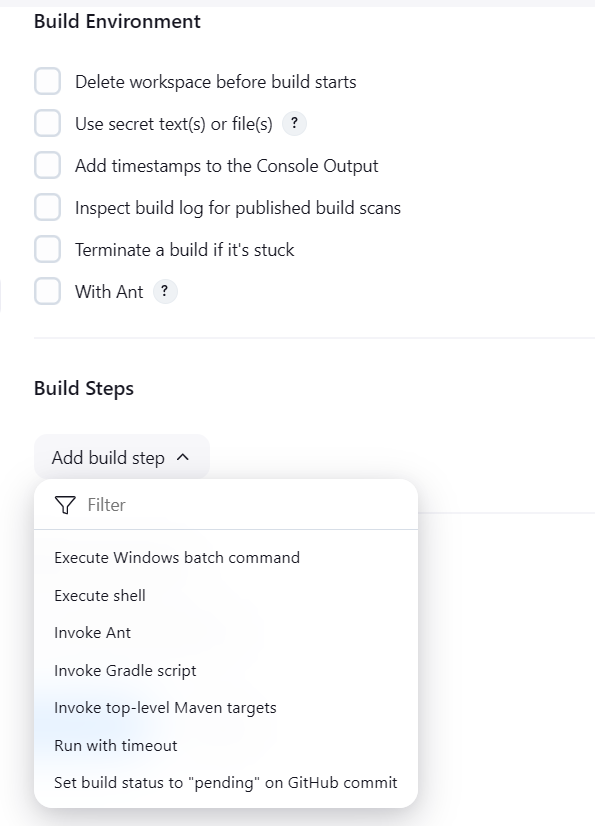
2. provide project name and click on “FreeStyle Project” and click on OK.



3. Fill below details

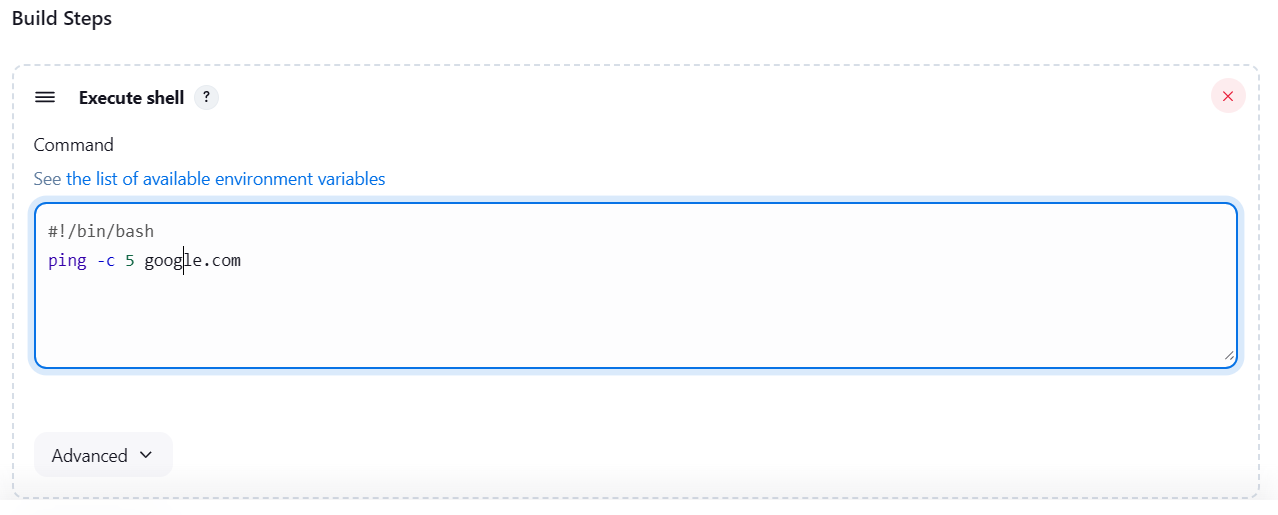






Click on Build Steps and select Execute Shell.

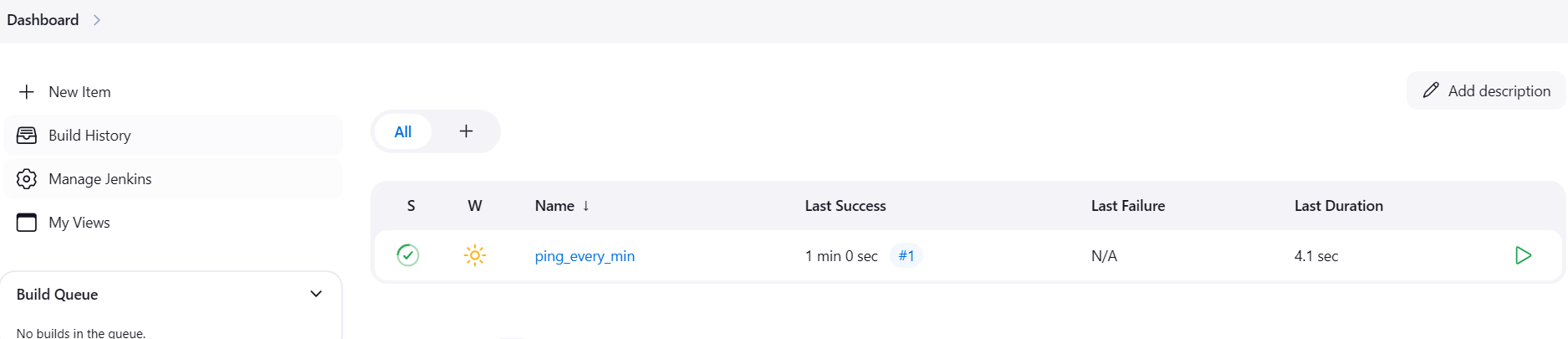
Add details about the script which you want to execute



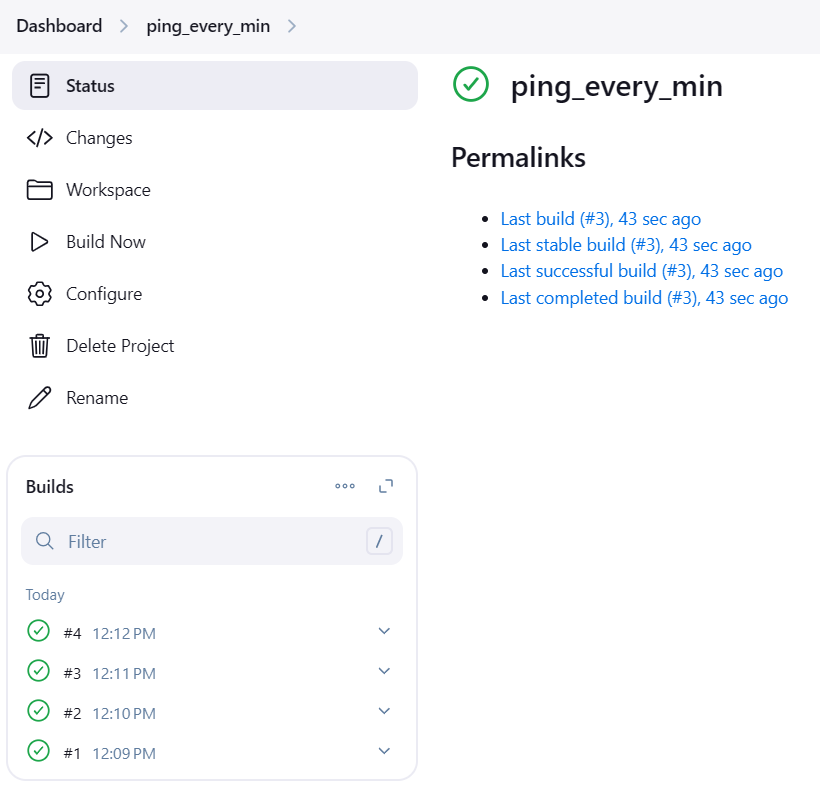
Click on Save button so script will execute every minute



Click on Dashboard upper left corner to check job is running or not



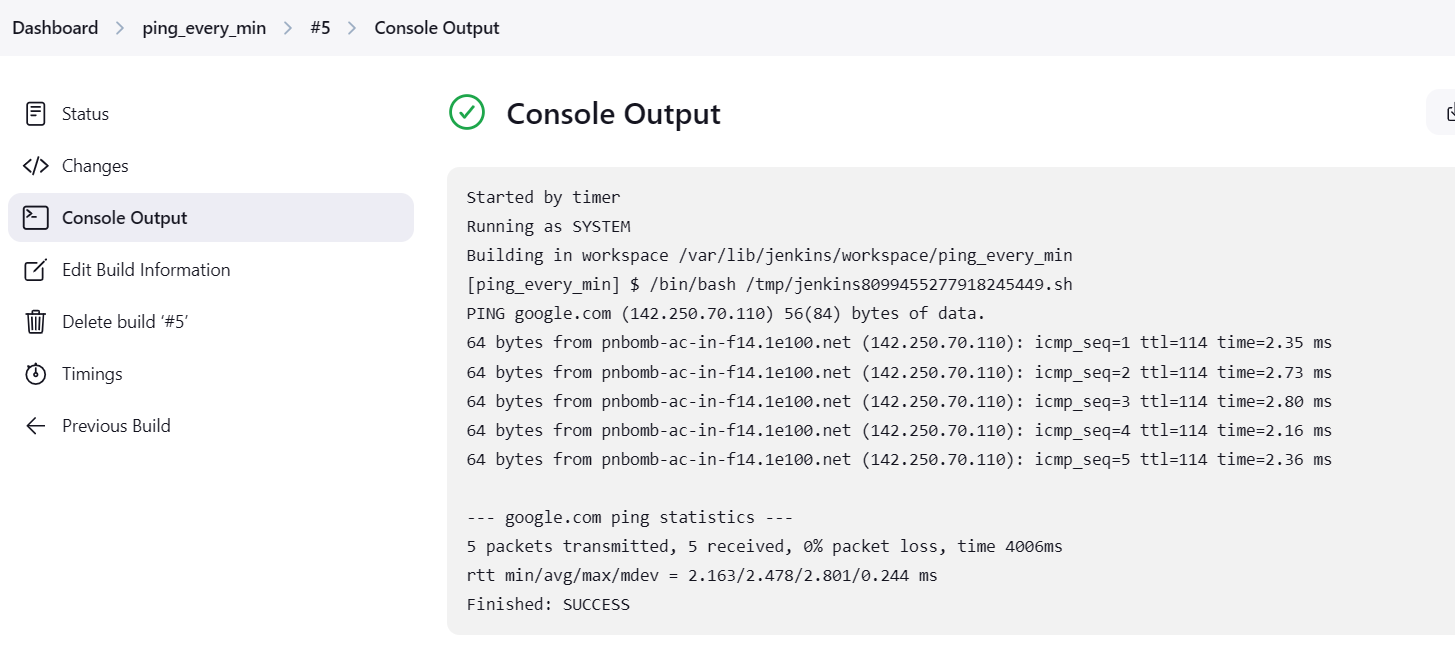
Click on job i.e. Ping\_every\_min to check the log and execution of job.



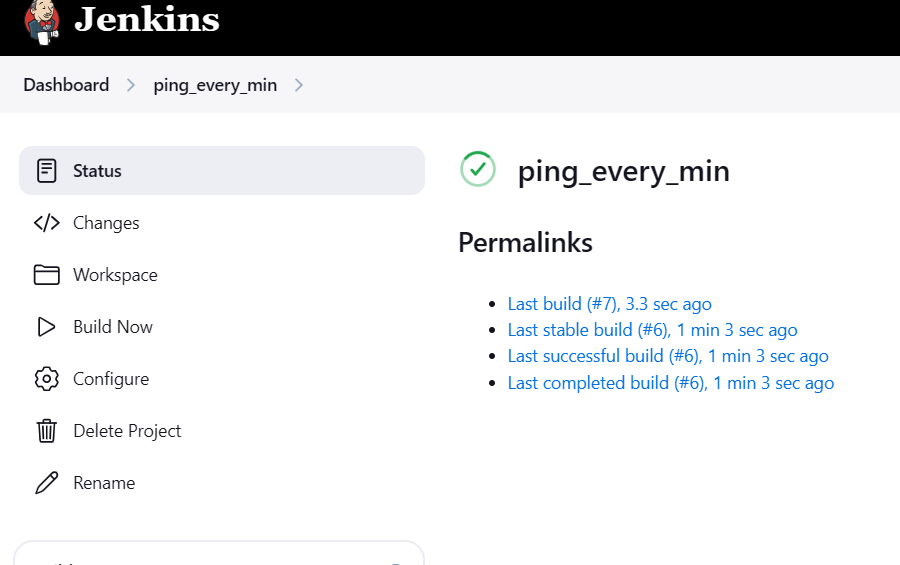
In builds, we can see job cycle execution, click on any number to check the log.

Green means job completed successfully and Red means job failed with some error.

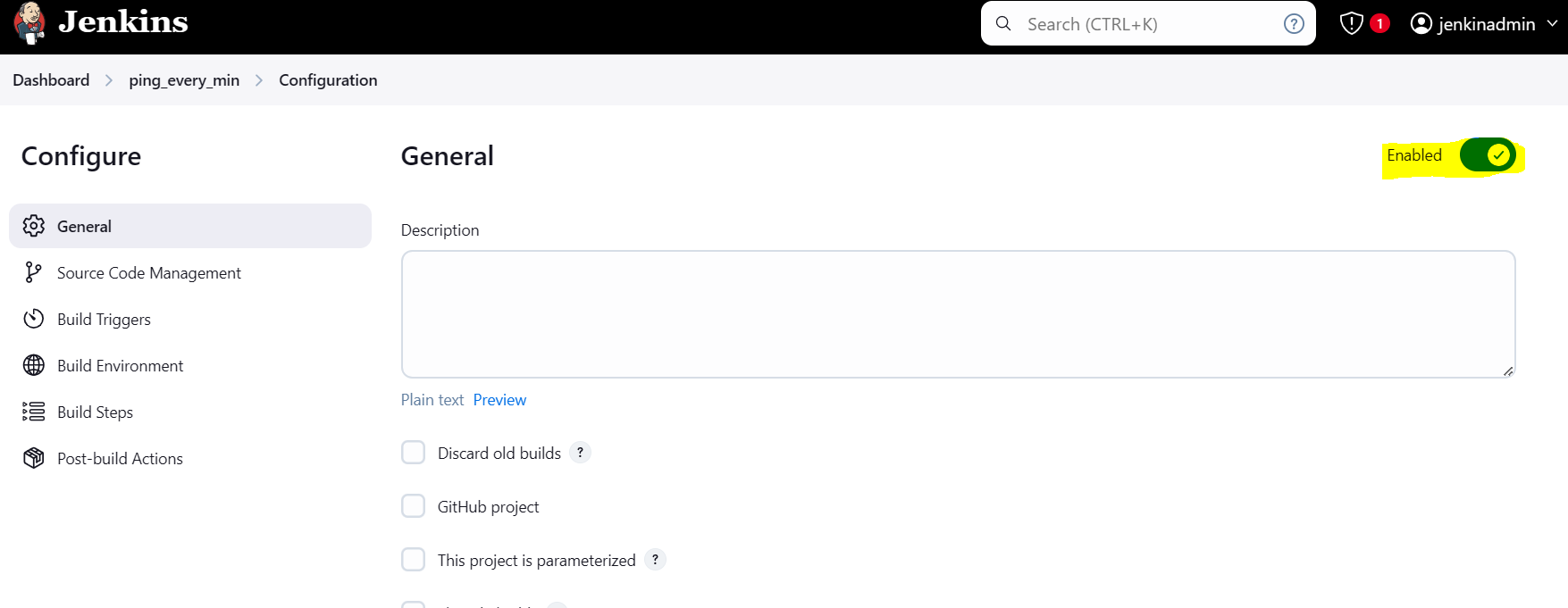
Once you click on any cycle. It open new window and click on console output to see the logs.



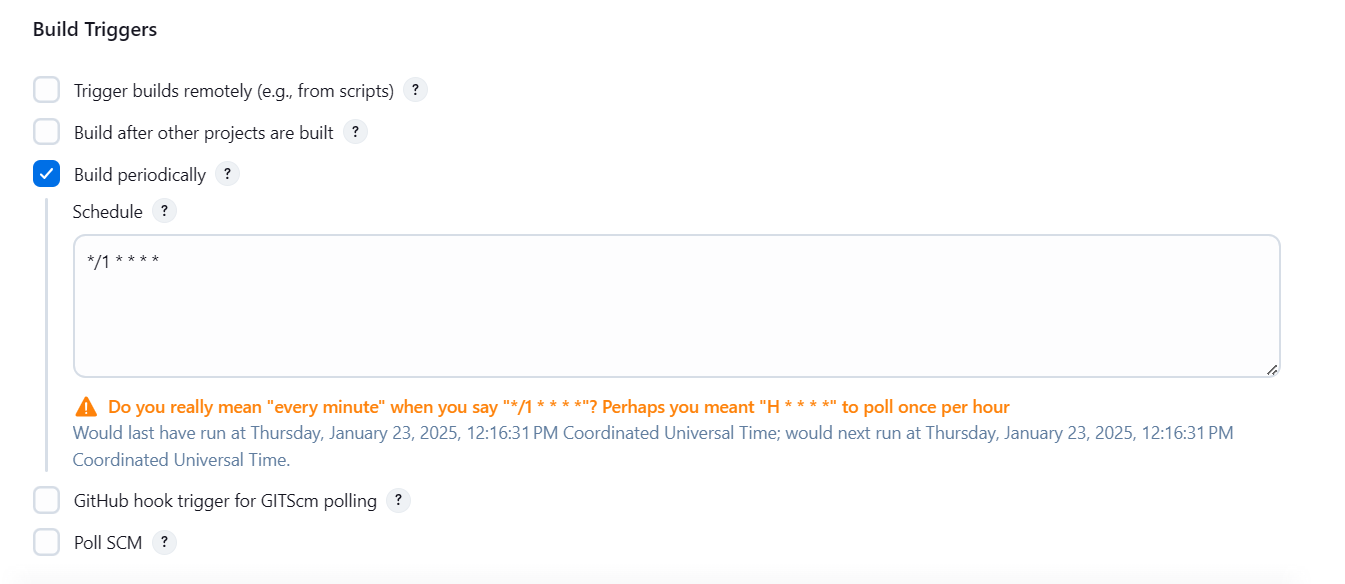
To stop the job or disable the job, Click on Dashboard -> click on job name 🡪 click on configure

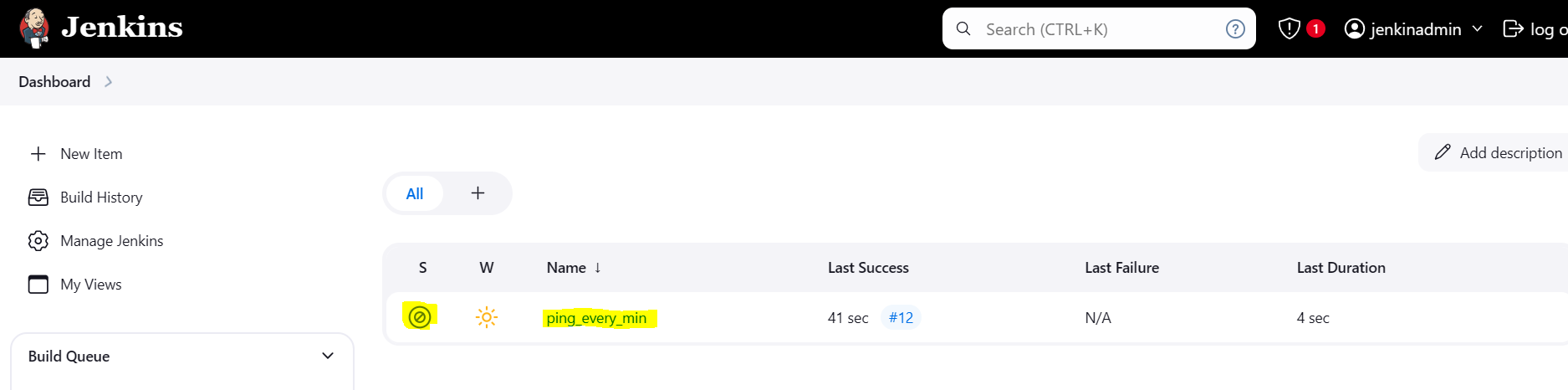


Click on Enable button as per screen shot..



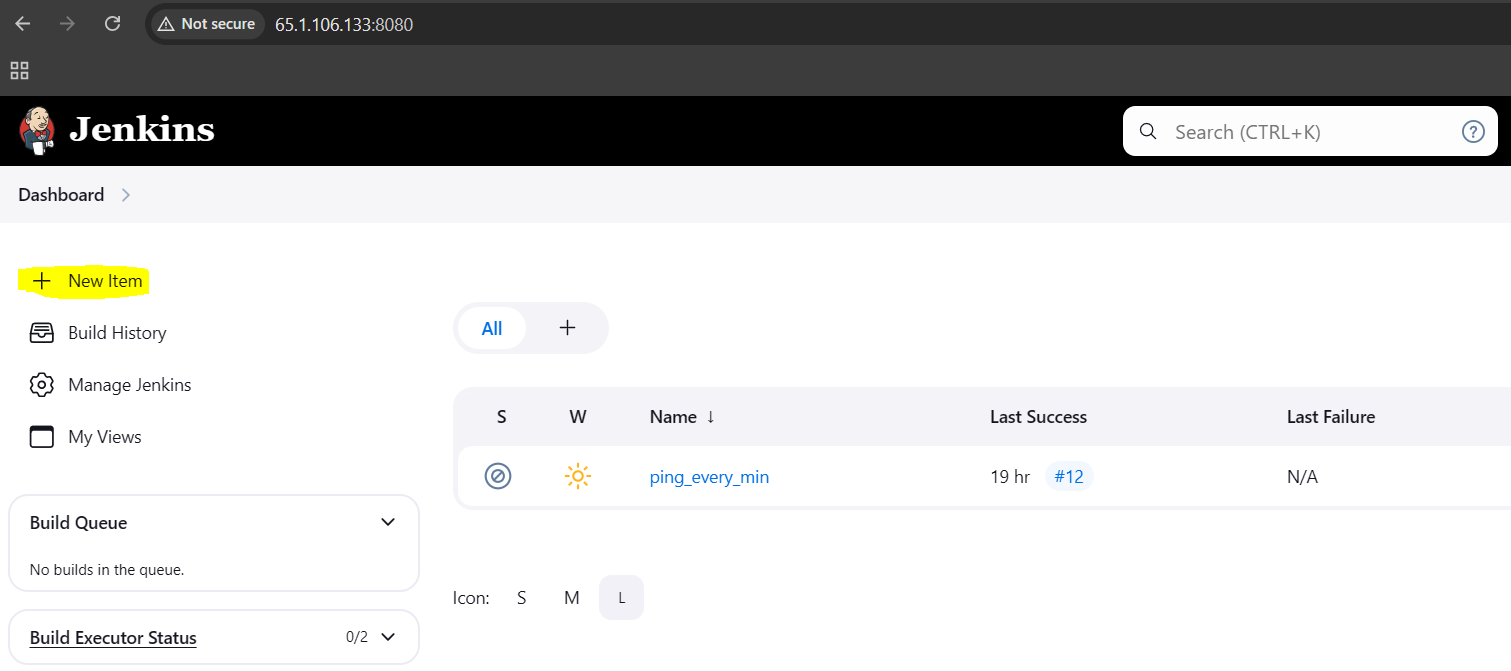
So project will be disable other wise we need to un tick build Periodically



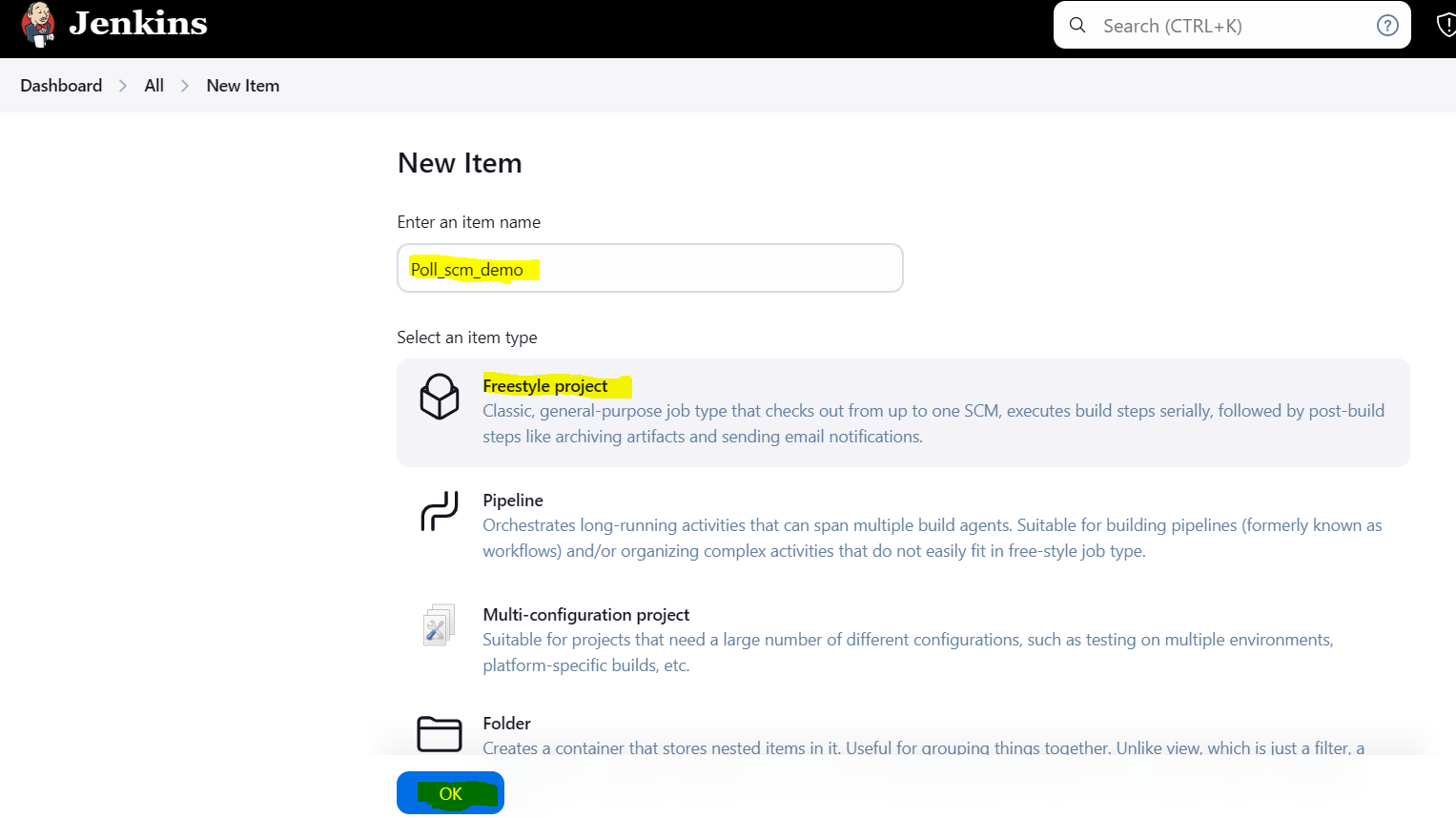


2. PULL SCM

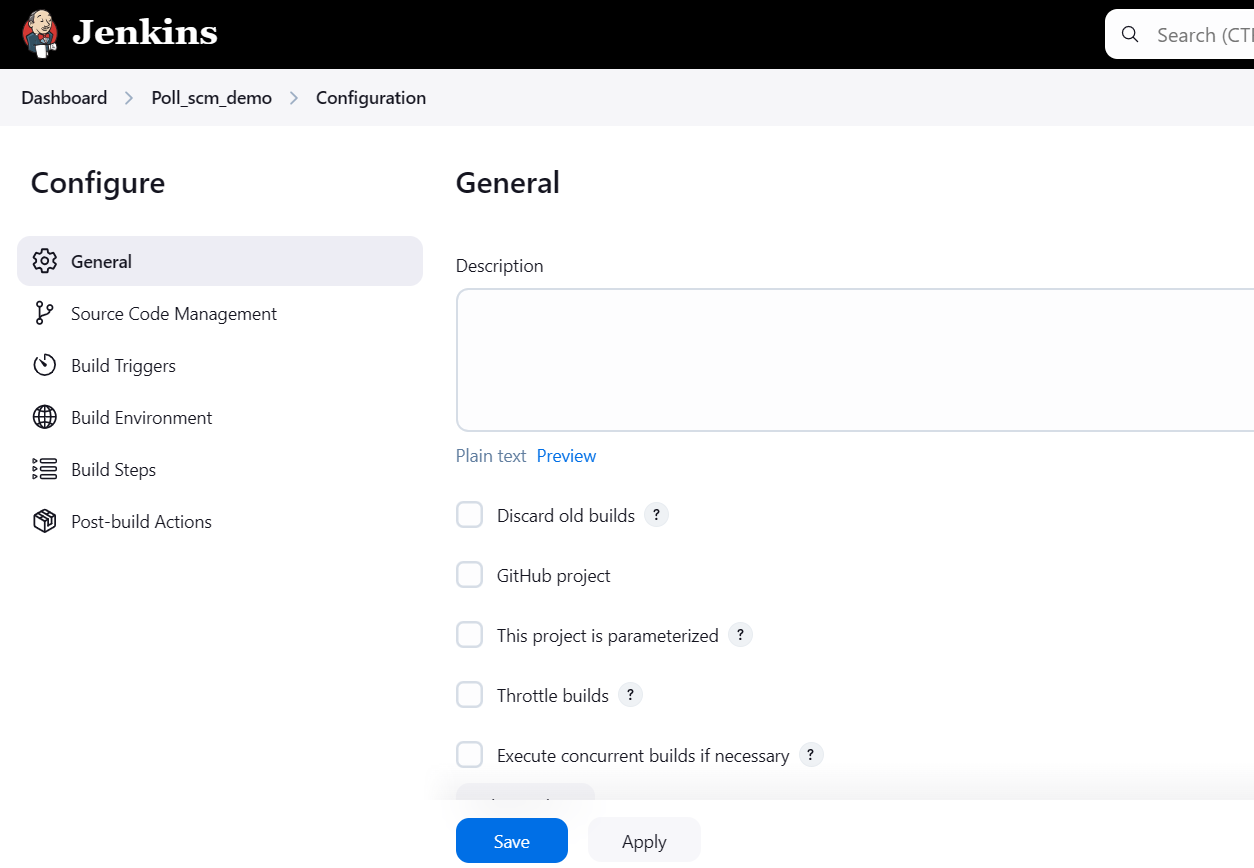
a. Click on + New Item



b. Give project Name 🡪 Select Free Style Project 🡪 Click on OK

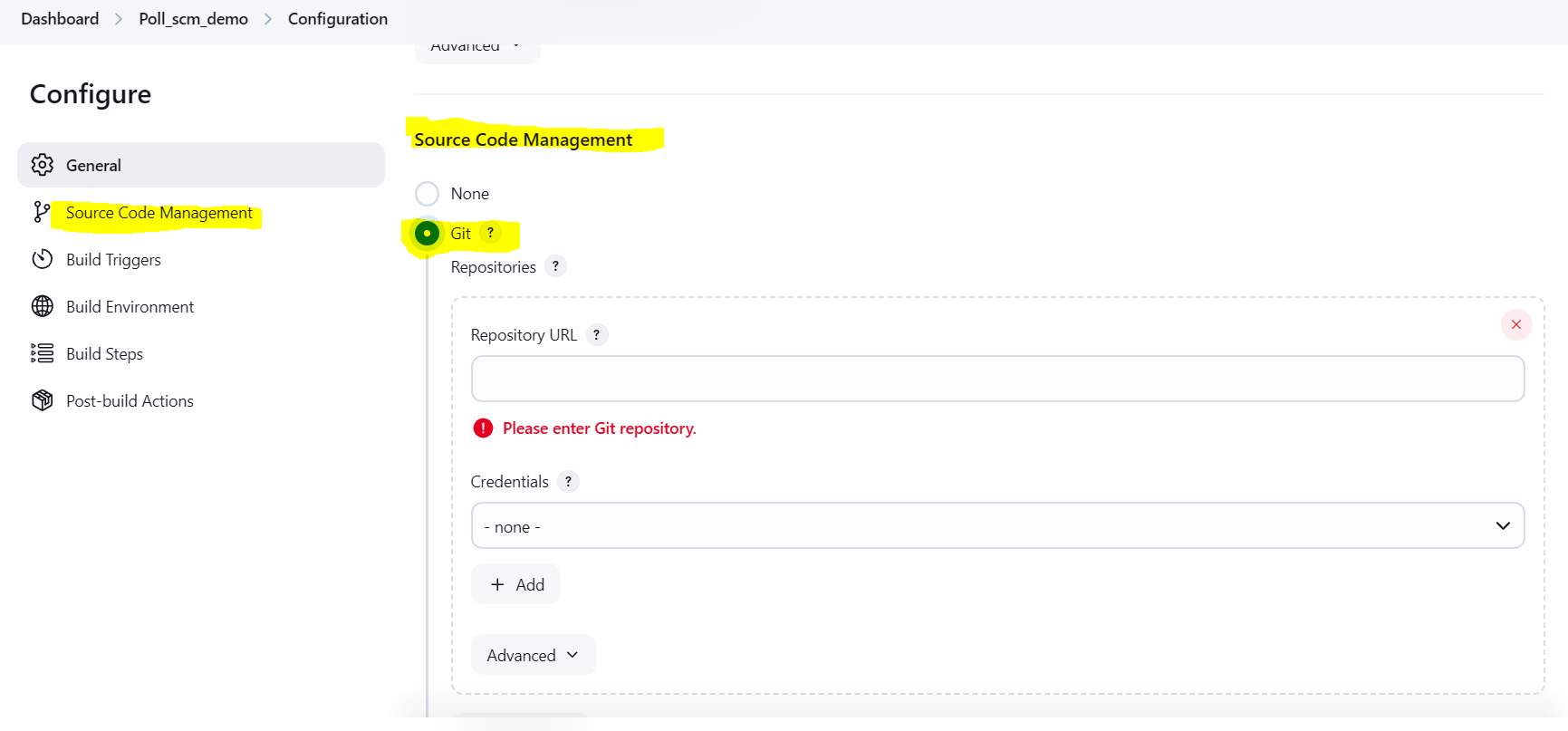


c. It will open new Window for Configuration of project.

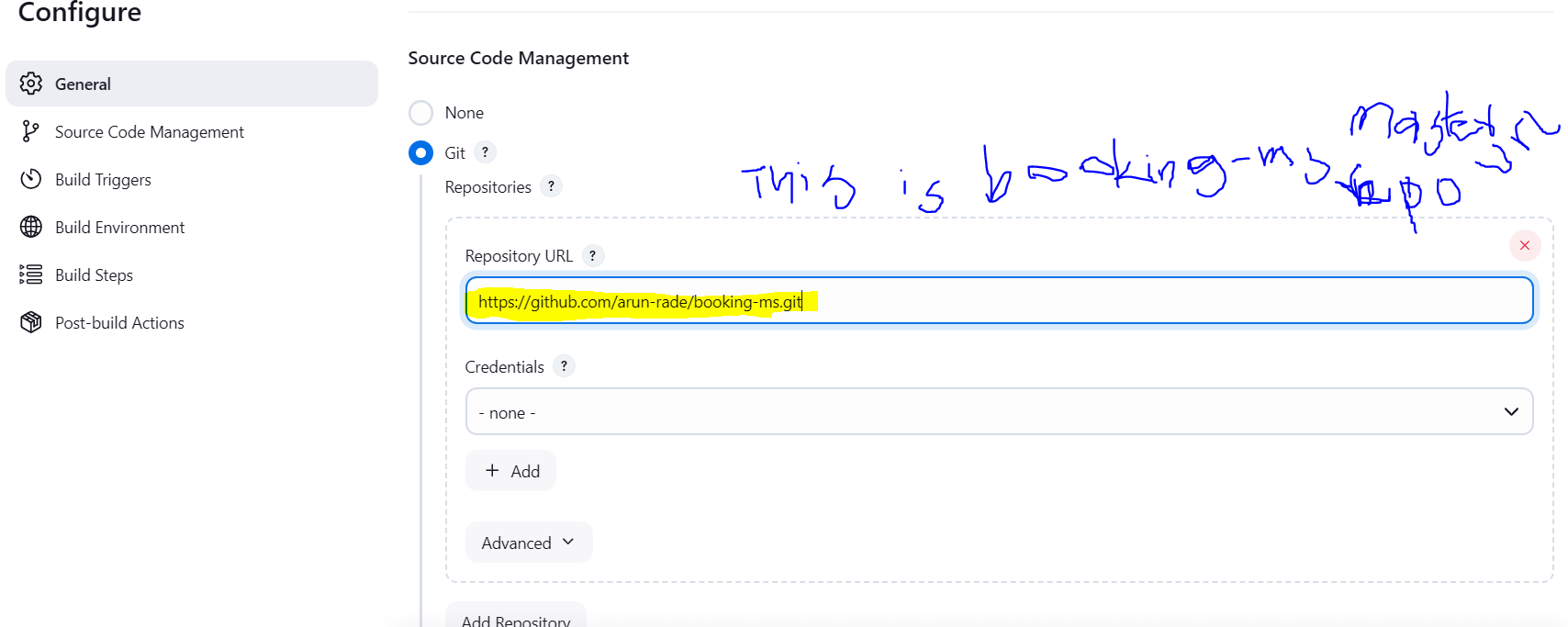


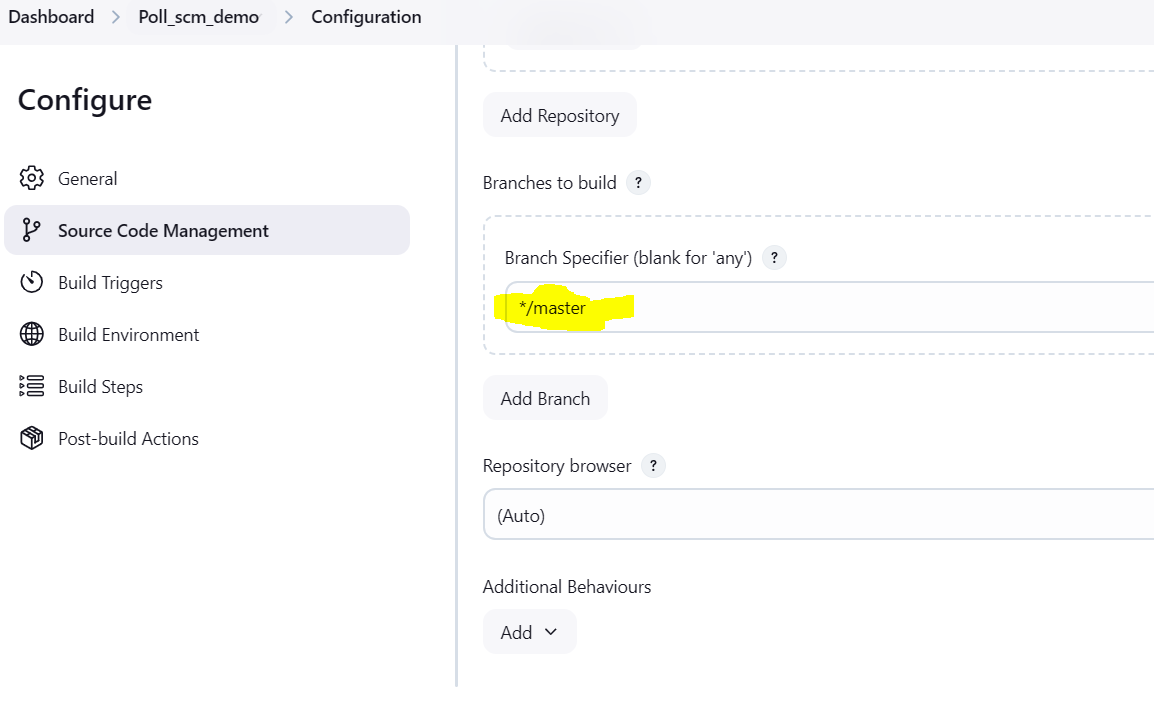
d. Fill below details as per screen shots

Goto Source Code management 🡪 Select Git

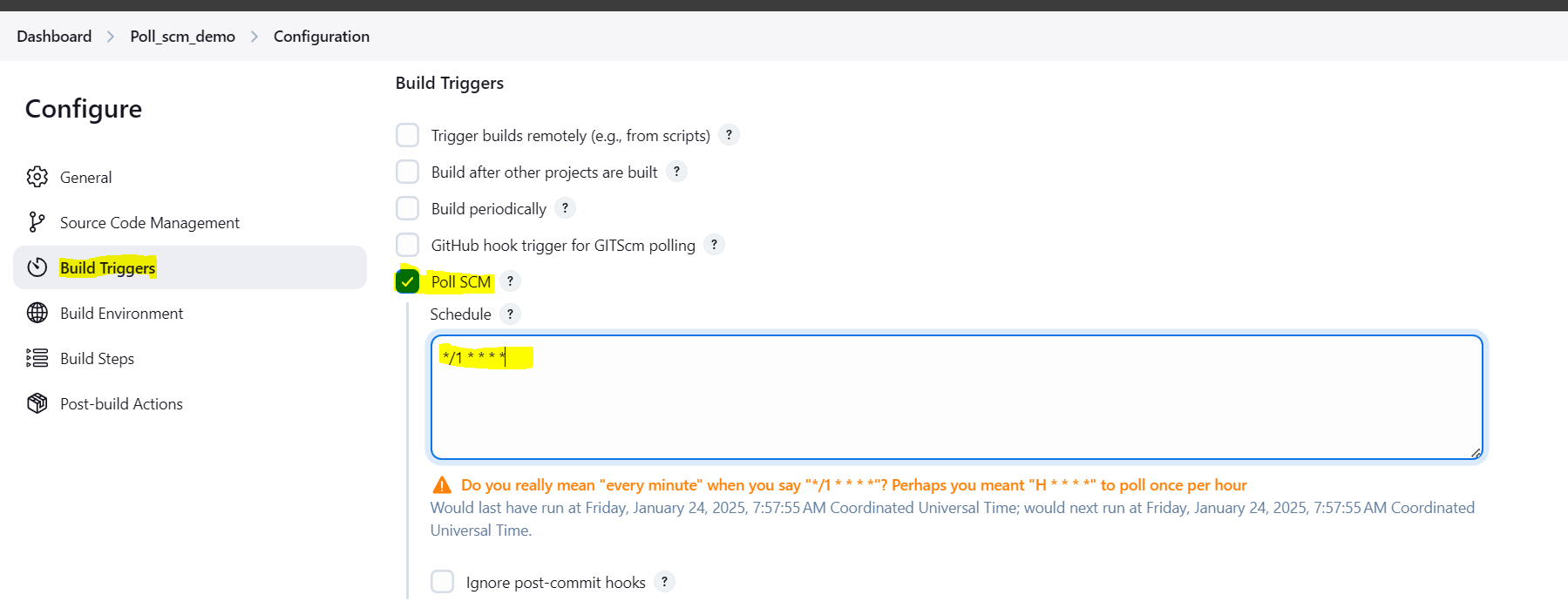


e. To add the Repository URL, login on GitHub select specific Repository and get repository URL and paste in Jenkins Source Code management under Git

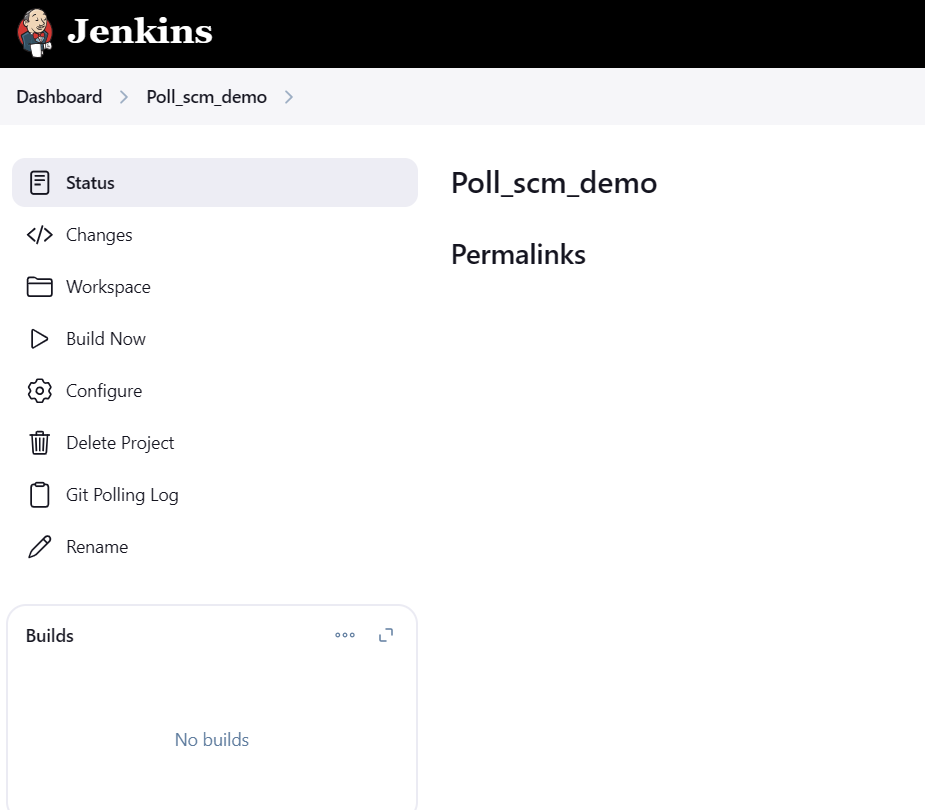


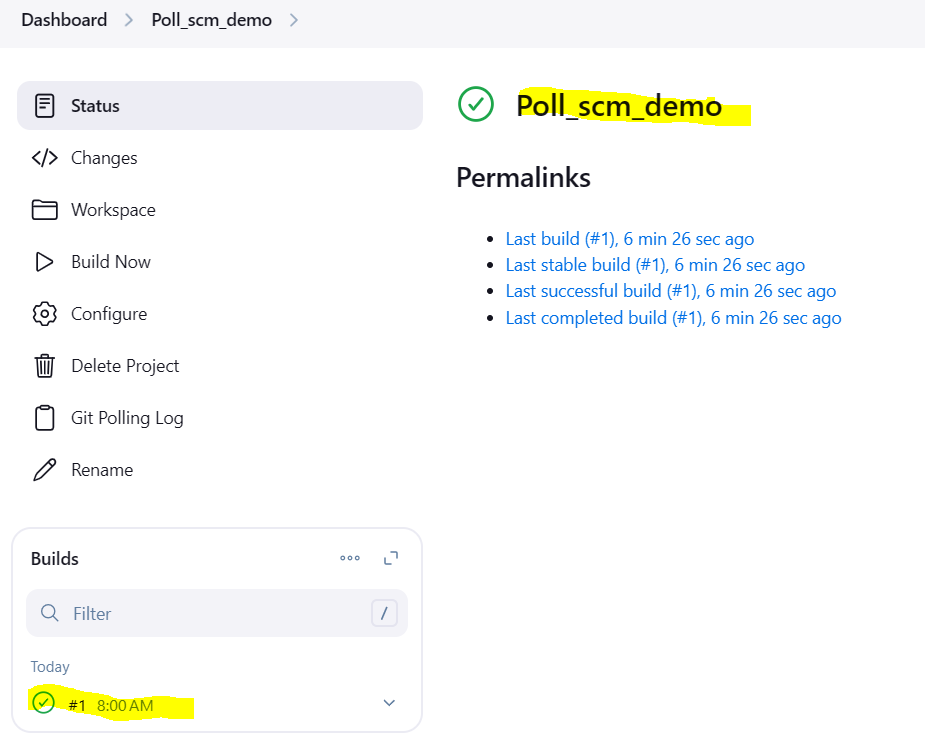


f. Goto Build Trigger 🡪 Select Poll SCM 🡪 provide timings at what time you want run the script.

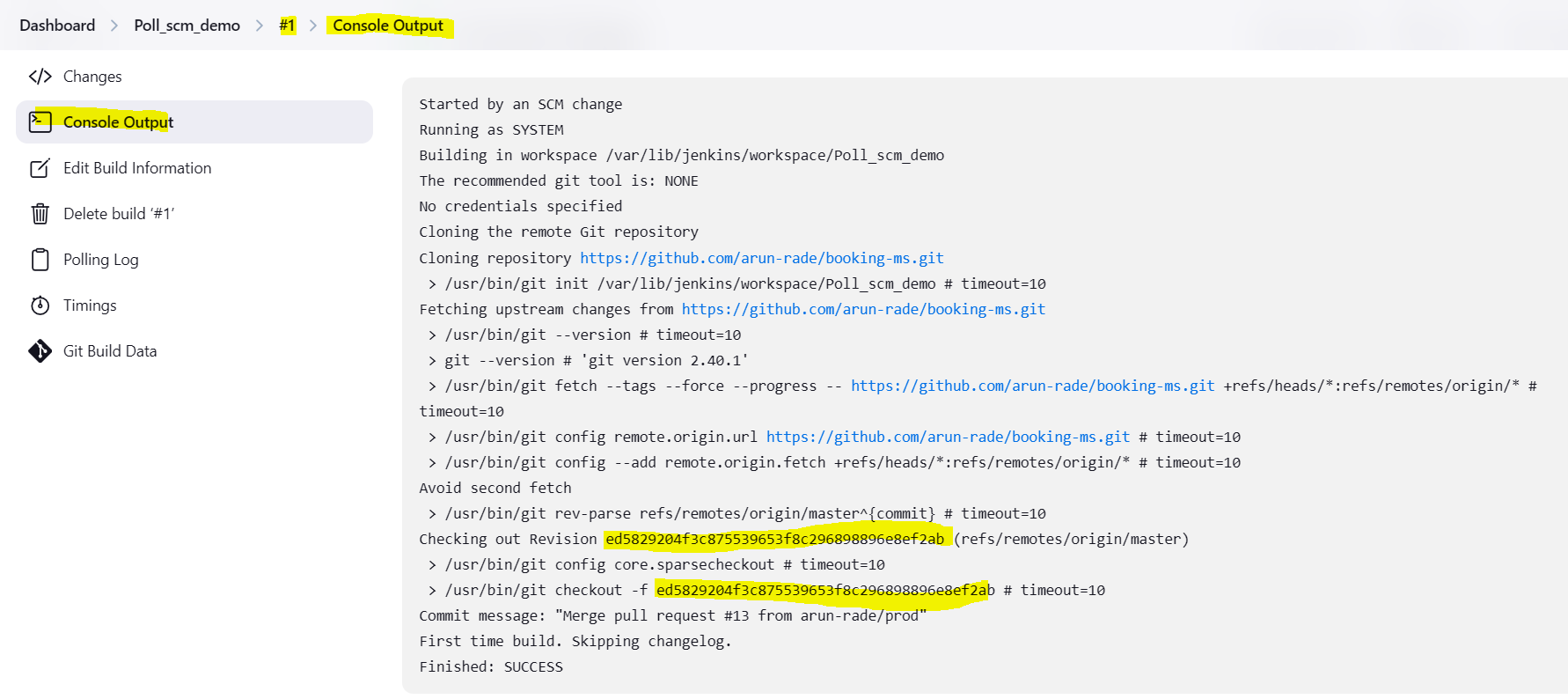


And Save the changes





First Jenkins Pull SCM Demo job run and fetch last commit from github repo.

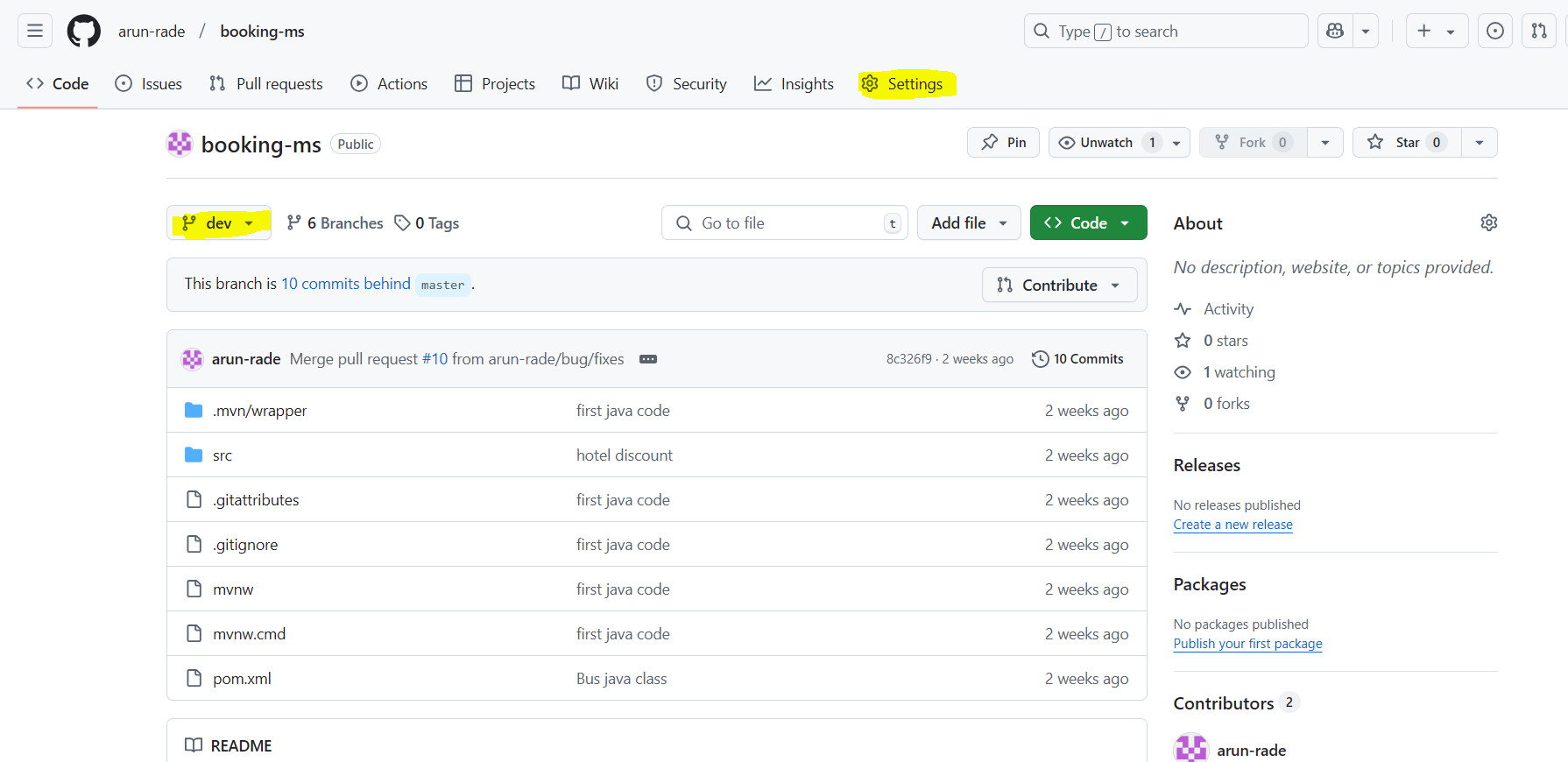


And waiting for next commit on configured repo. This job will poll/trigger and check any commit happened or not. If commit not happened job will not trigger.

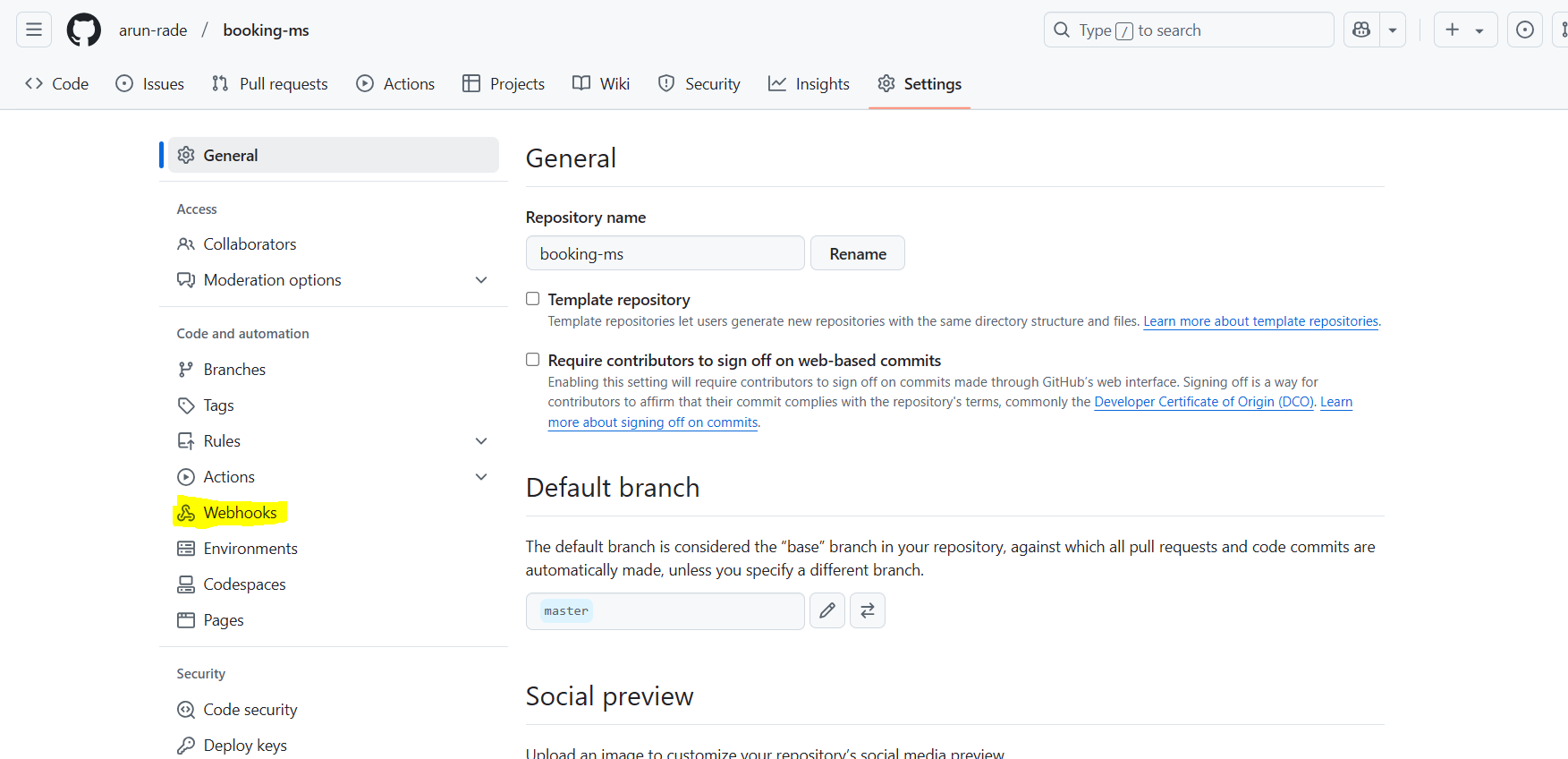
Go to the Git Hub and make some changes in respective repo and save changes/commit the changes. The Jenkins job will poll and fetch the last commit.

3. **Github hook trigger for GITscm Polling**

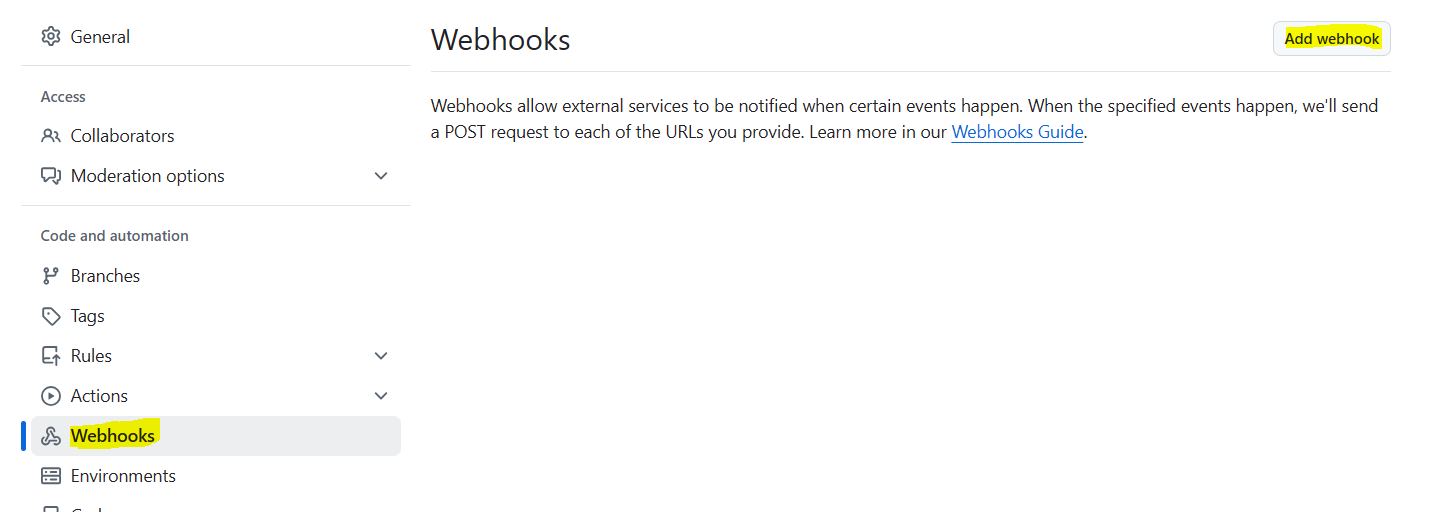
a. Goto Git Hub repository 🡪 select Dev branch (Which most updated branch) -- > click on Setting



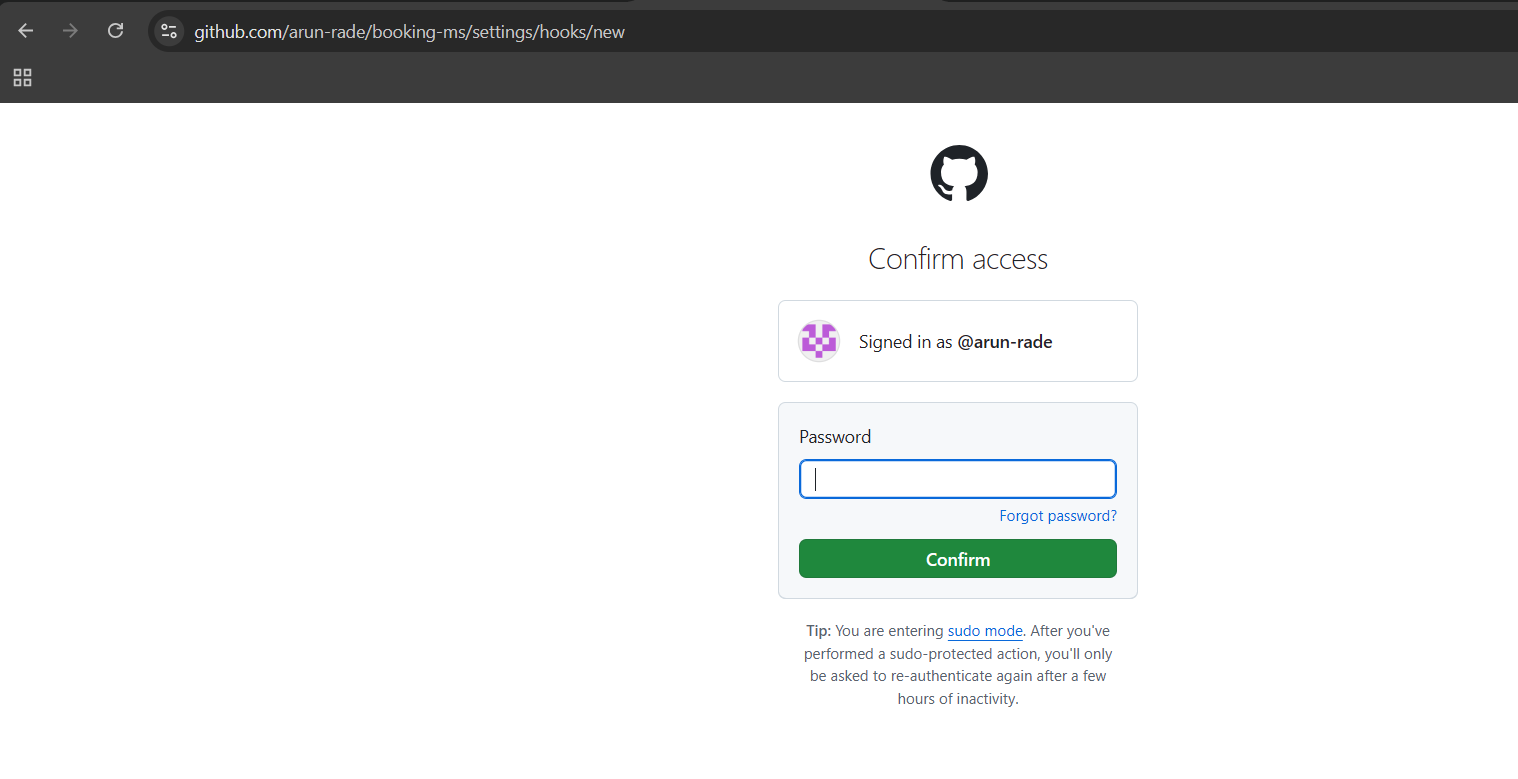
b. Click on Webhooks



c. Click on “Add Webhook” button as per screen shot.

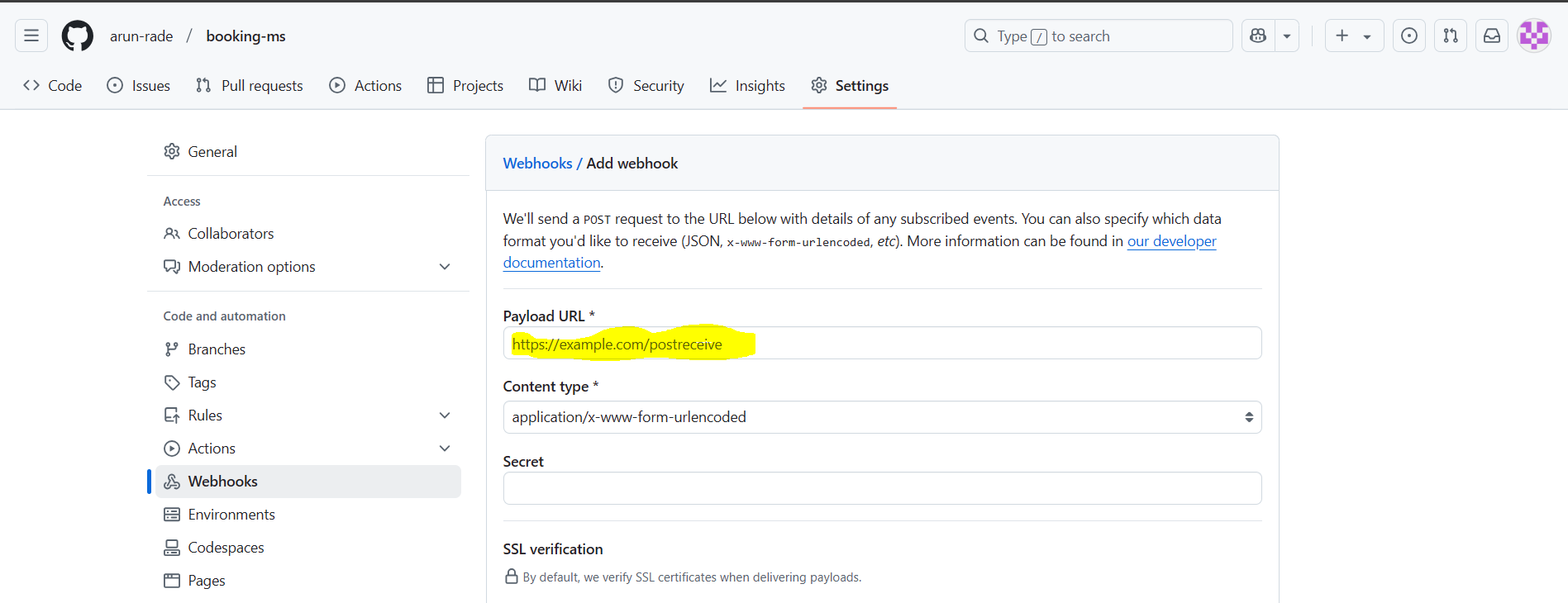


d. Provide Git Hub password : It will ask for password

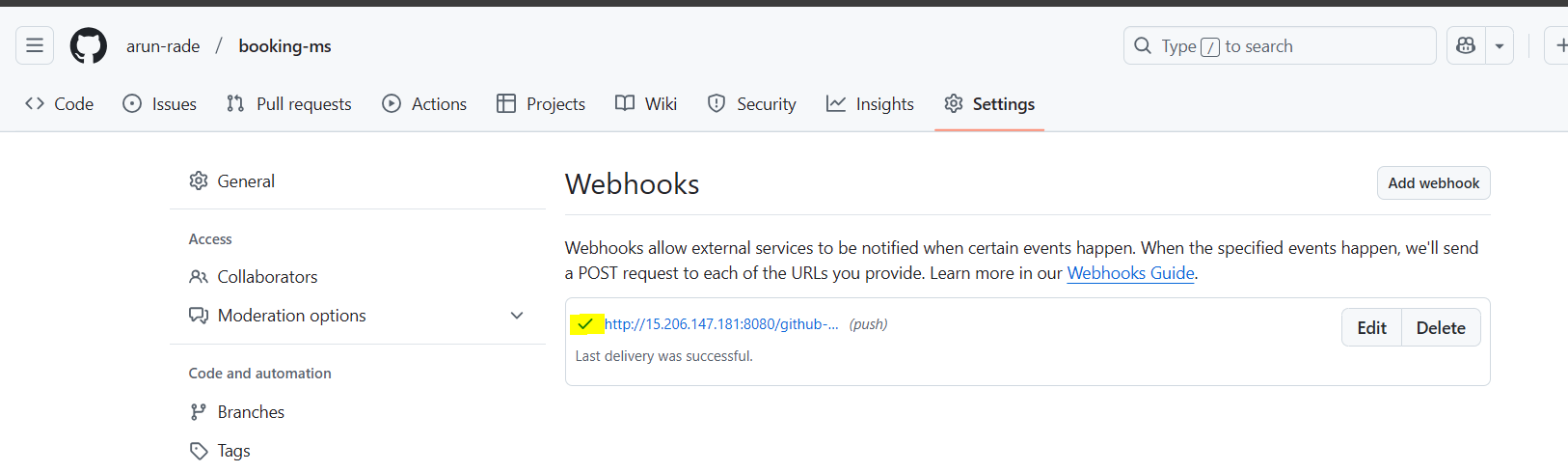


e. adds payload URL: as Jenkins URL + github-webhook

e.g. <http://15.206.147.181:8080/github-webhook/>

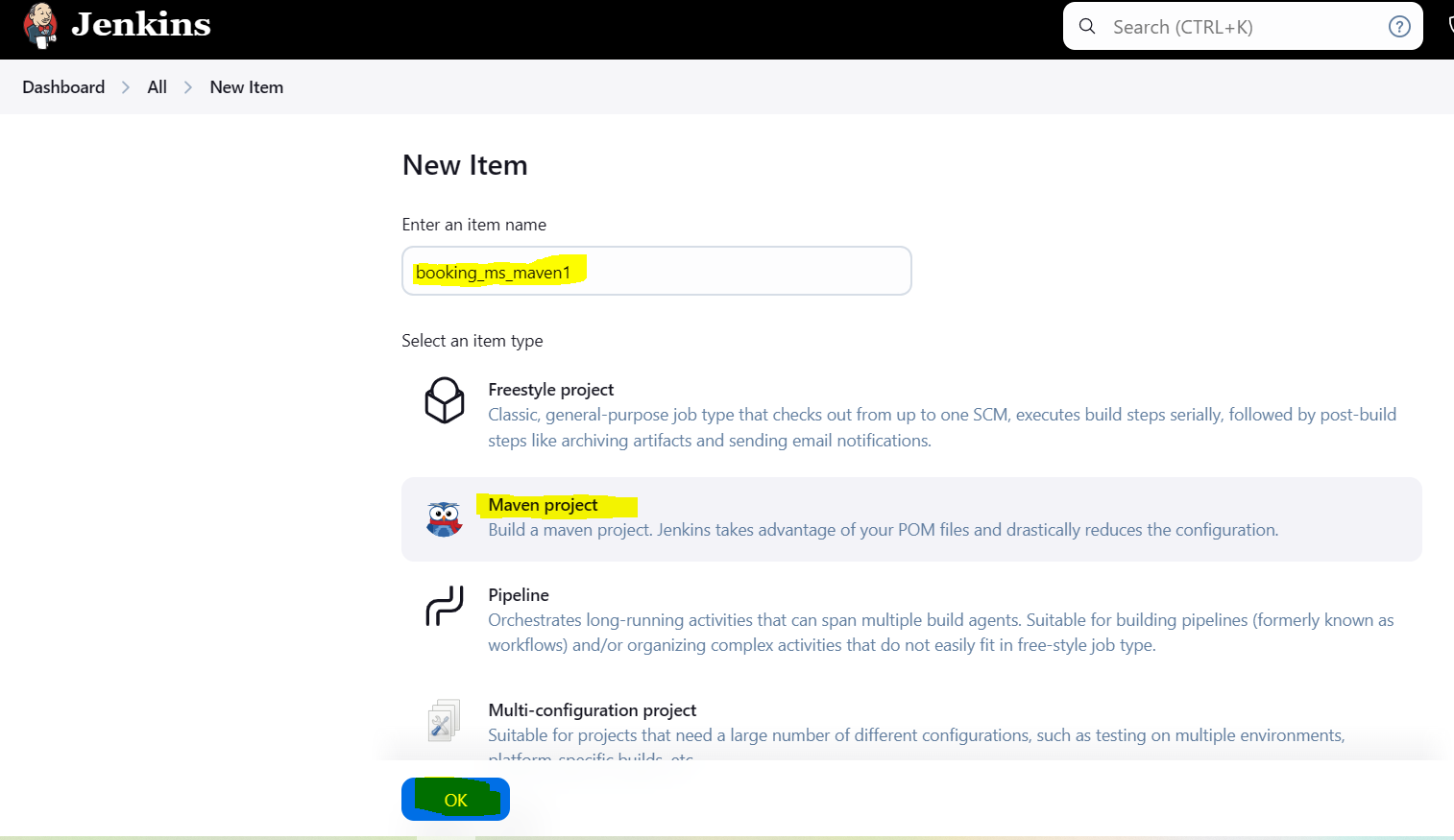


f. Click on “Add Webhook” and wait till we can get correct symbol like highlighted below screenshot

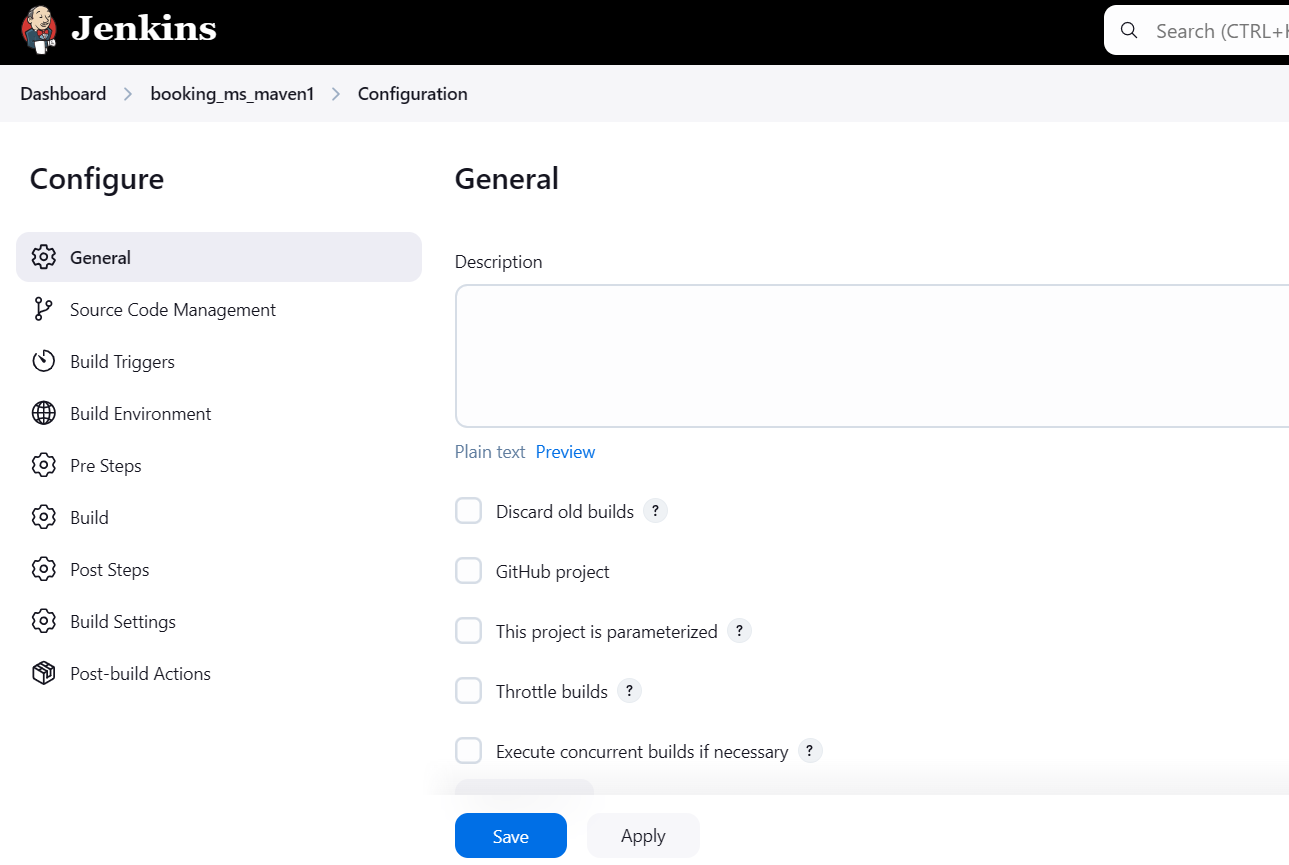


g. login on Jenkins

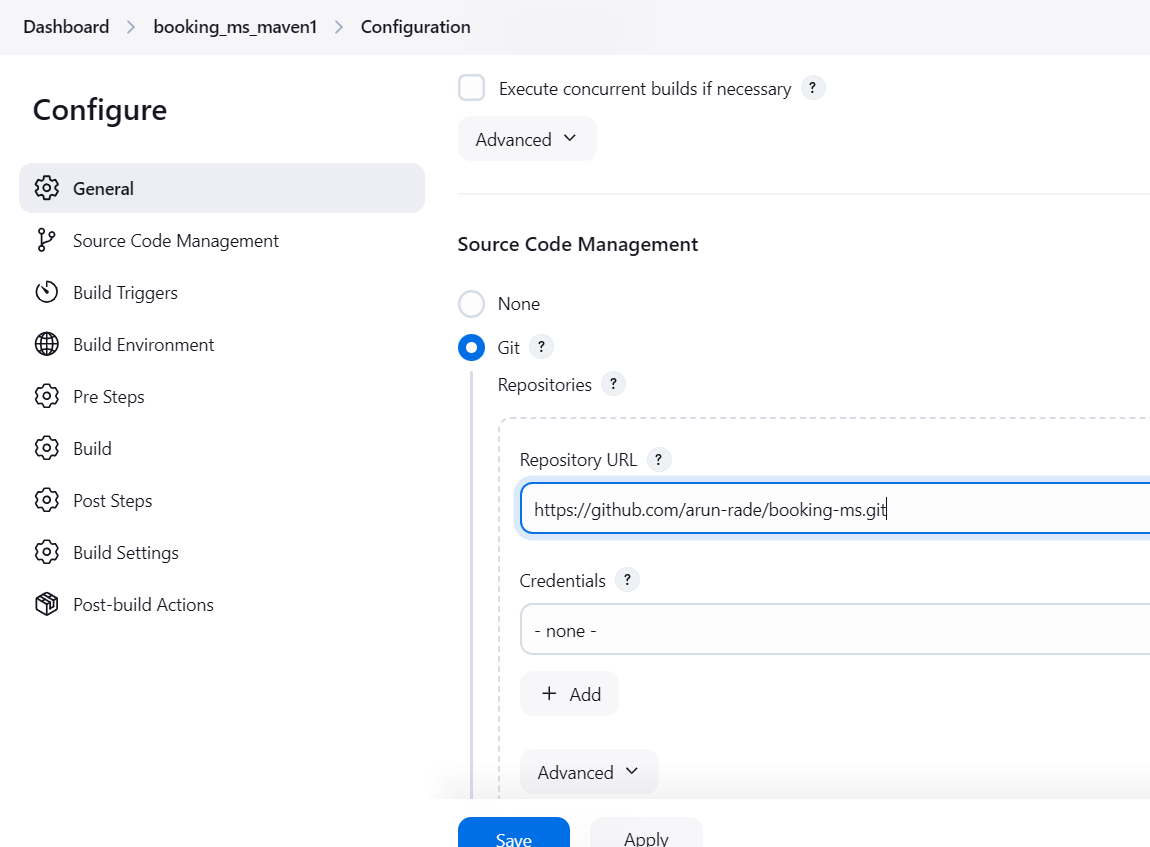
h. Click on new item 🡪 provide project name 🡪 click on maven project 🡪 click on ok

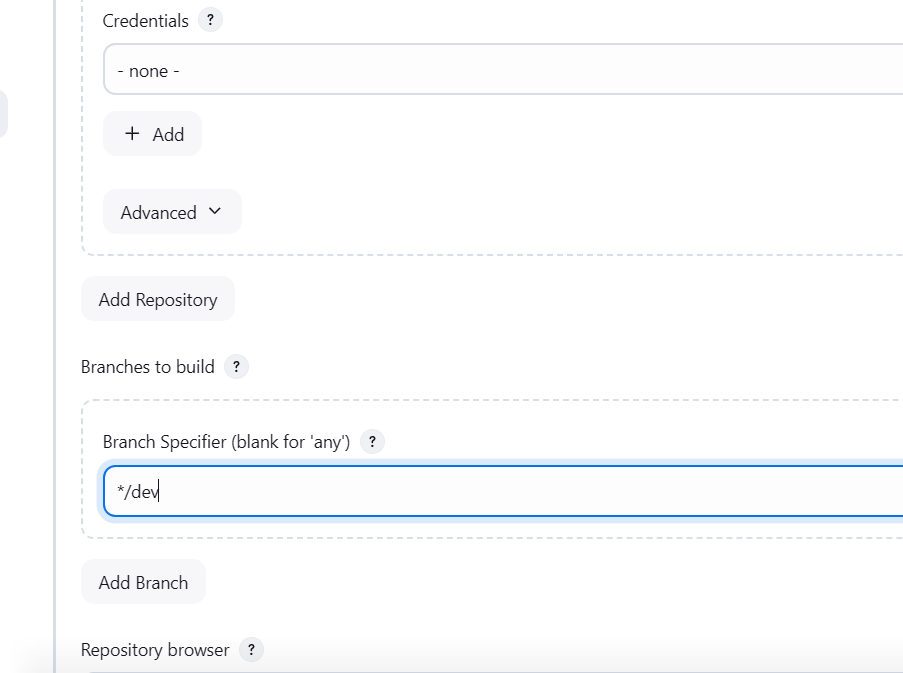


1. New window will open 🡪 goto source code management

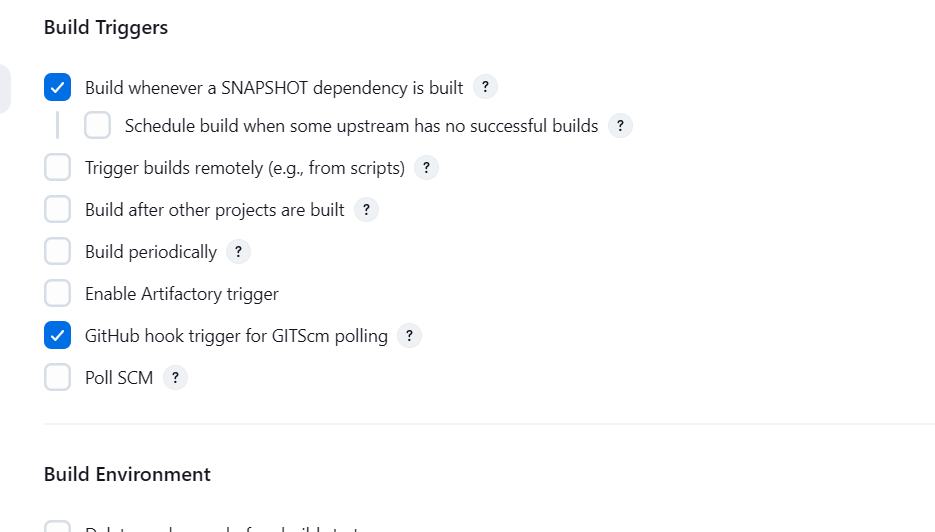


J. Select Git as Source code management 🡪 provide dev branch URL in this block.

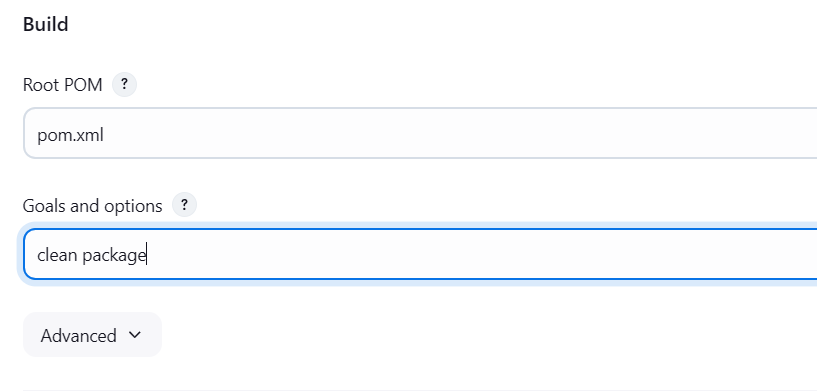




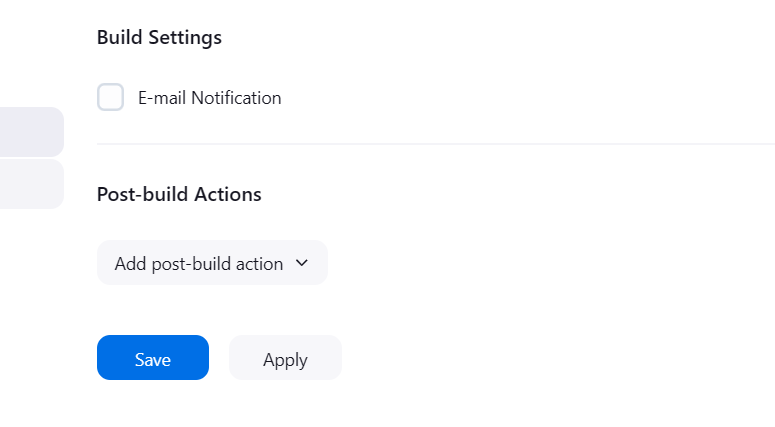
Goto BUILD TRIGGER 🡪 Select “GitHub hook trigger for GITSCM polling



Goto Build 🡪 you can see Root POM is “POM.xml” and we need provide Goals and Options like clean package



Save changes



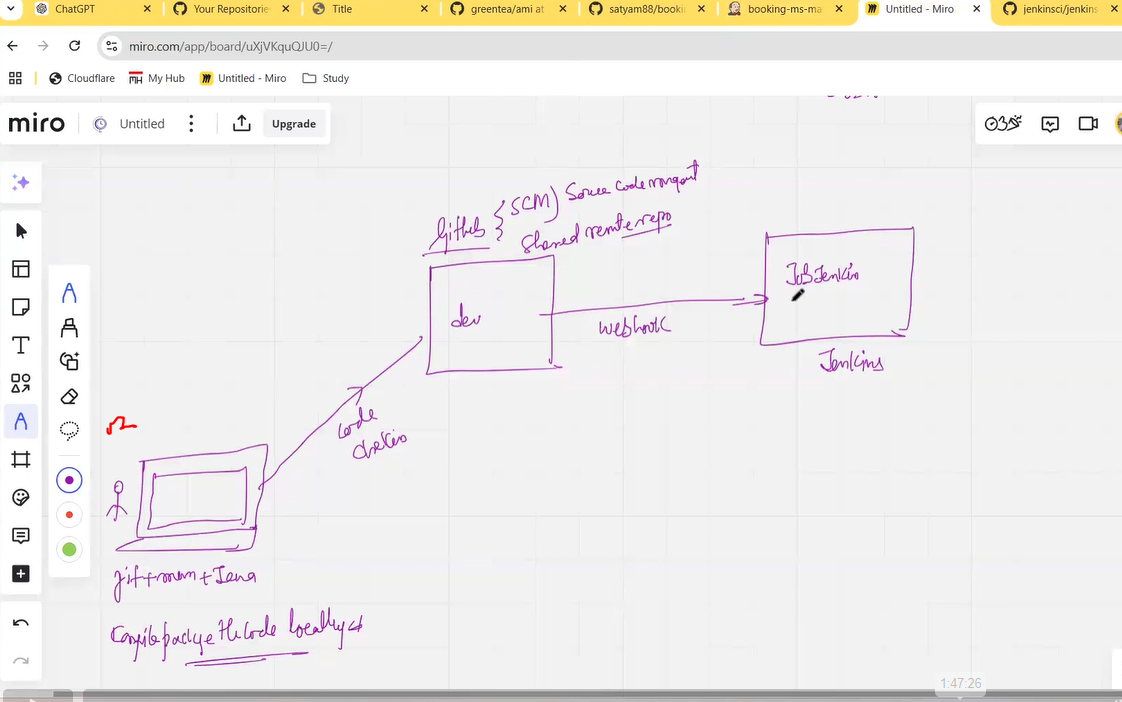
Open IntellijIDEA to make some changes in booking-ms project. So that we need to perform below commands

Check branch you are present 🡪 git branch

If you are in another branch you need to come in dev branch 🡪 git checkout dev

To sync (remote dev and local dev ) dev branch :🡪 git pull origin dev

So latest changes pull from remote dev branch to local branch



Add class file or do some changes in POM.xml or whatever changes required you can do. Compile, package and run the code locally with below command

mvn clean compile

mvn clean package

mvn spring-boot:run

and try to launch URL

After that push the changes to remote repository.

git add - - all

git commit -m “changes”

git push origin future/bug

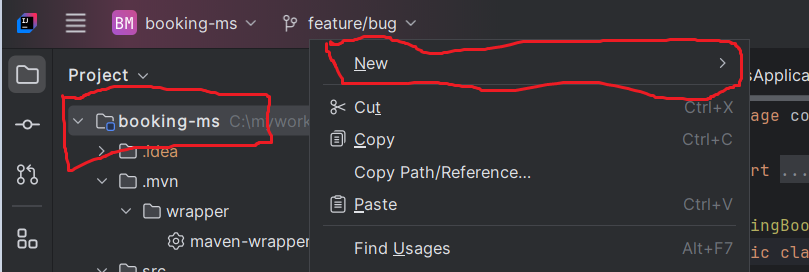
login on github 🡪 check the new branch is appear 🡪 create pull request from feature/bug to dev branch

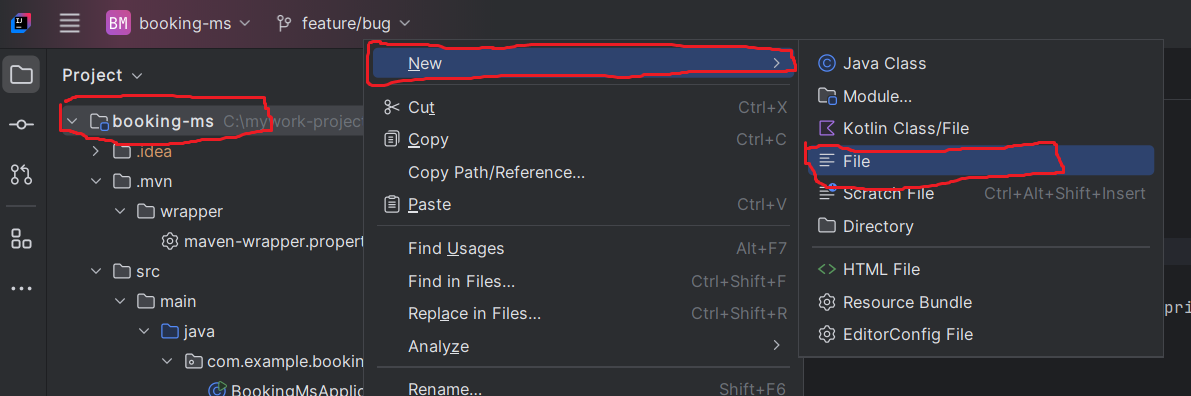
once this merge request gets complete, new job will trigger on Jenkins because we have created webhook for dev branch.

PIPELINE jobs:

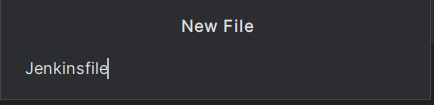
1. Create pipeline file in intellij Idea for specific project

Open ItellijIdea 🡪 Right click on project (booking-ms) 🡪 new window will pop up 🡪 click on New button 🡪 select file

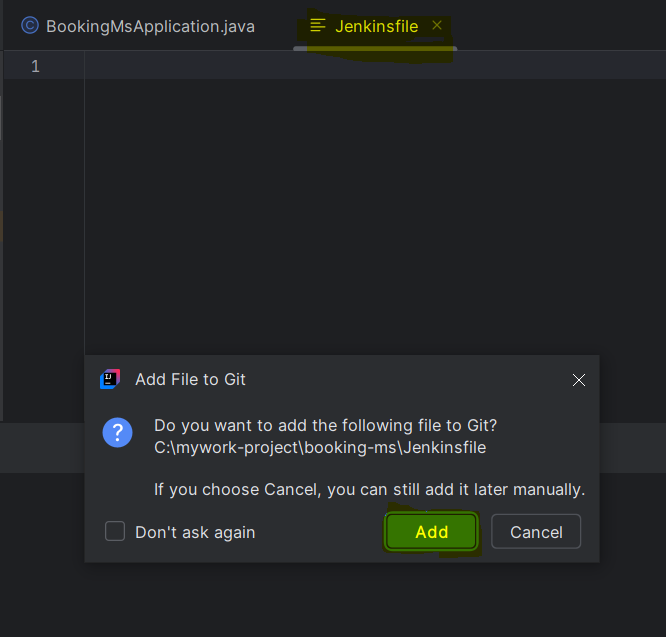




New window will pop-up 🡪 provide file name **“Jenkinsfile”**



**New window will pop up to add file in Git**



**Add below code in Jenkinsfile**

**pipeline {**

**agent any**

**options {**

**buildDiscarder(logRotator(numToKeepStr: '5', artifactNumToKeepStr: '5'))**

**}**

**tools {**

**maven 'mvn\_3.9.9'**

**}**

**stages {**

**stage('Code Compilation') {**

**steps {**

**echo 'Compiling the code...'**

**sh 'mvn clean compile'**

**}**

**}**

**stage('JUnit Execution') {**

**steps {**

**echo 'Executing JUnit tests...'**

**sh 'mvn test'**

**}**

**}**

**stage('Code Packaging') {**

**steps {**

**echo 'Packaging the code...'**

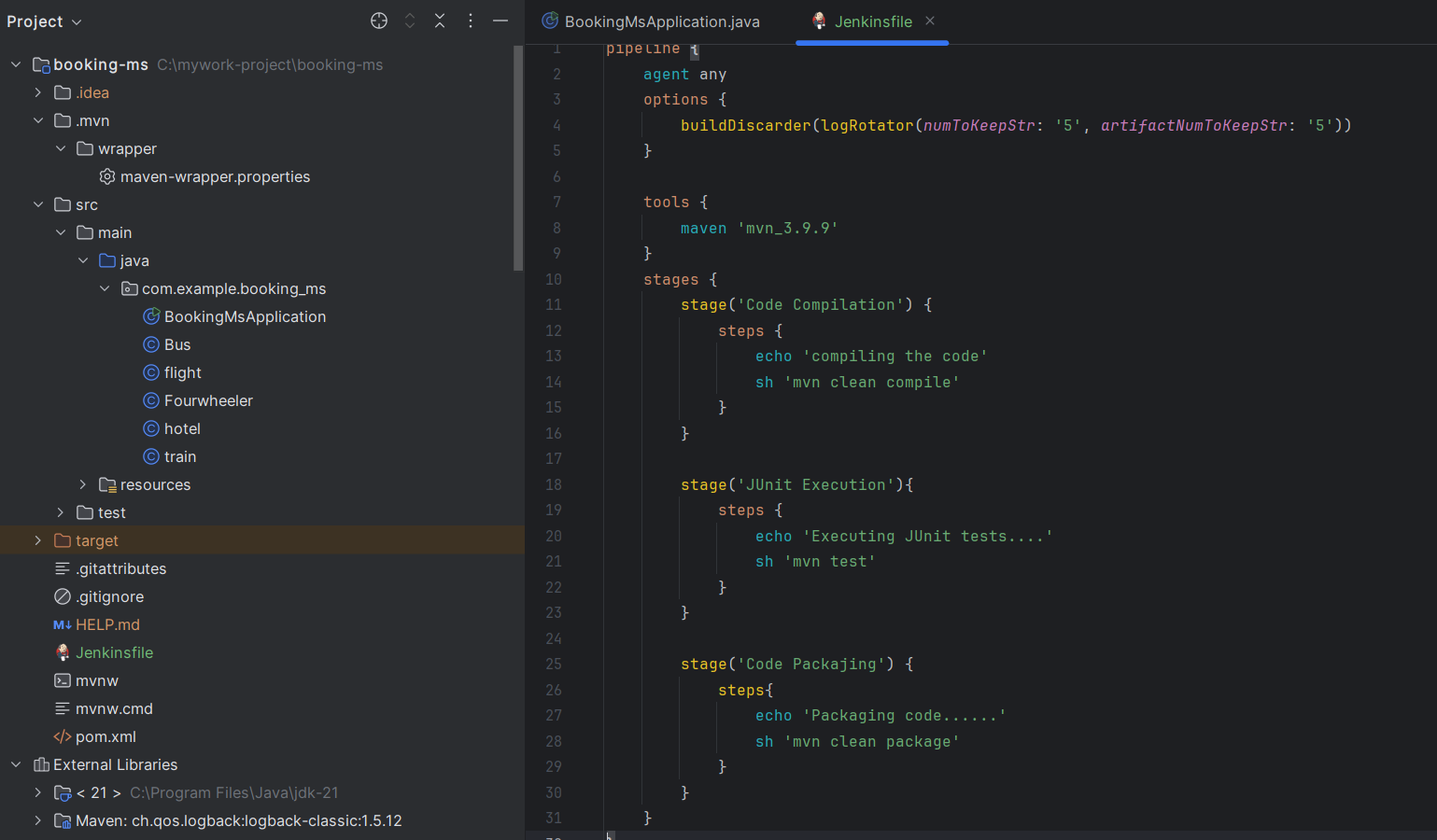
**sh 'mvn package'**

**}**

**}**

**}**

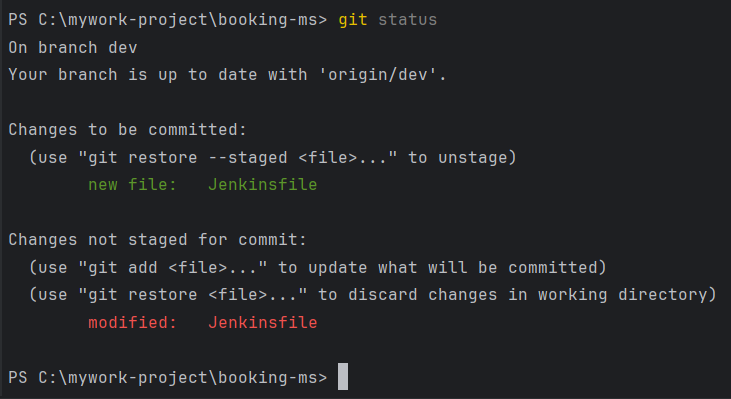
**}**



**Save the file and push the changes to dev remote branch.**

**Check how many files are changed with below command**

**git status**

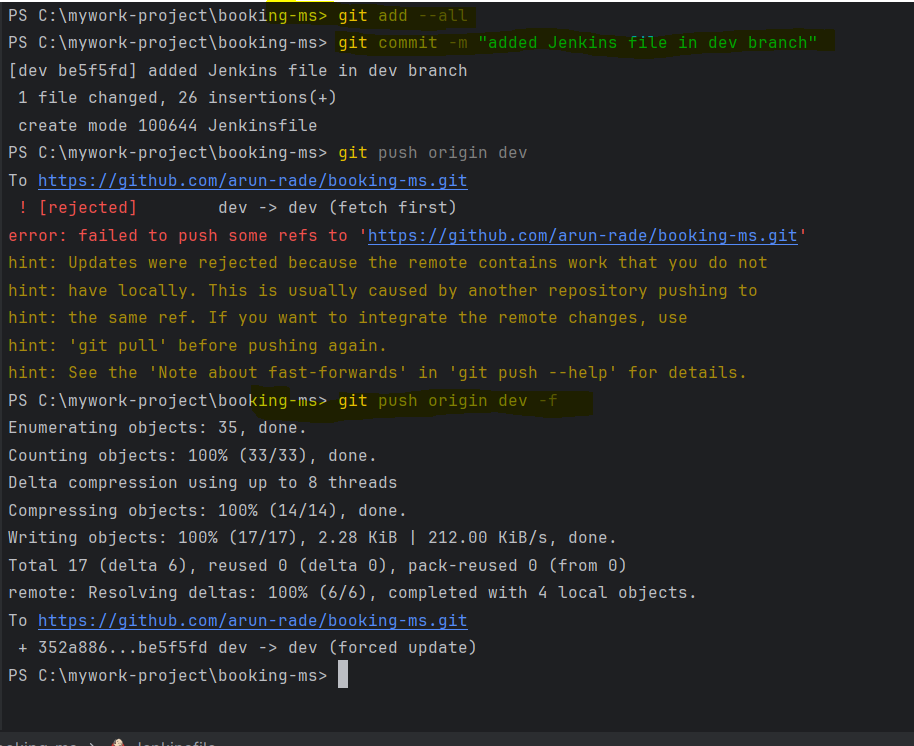


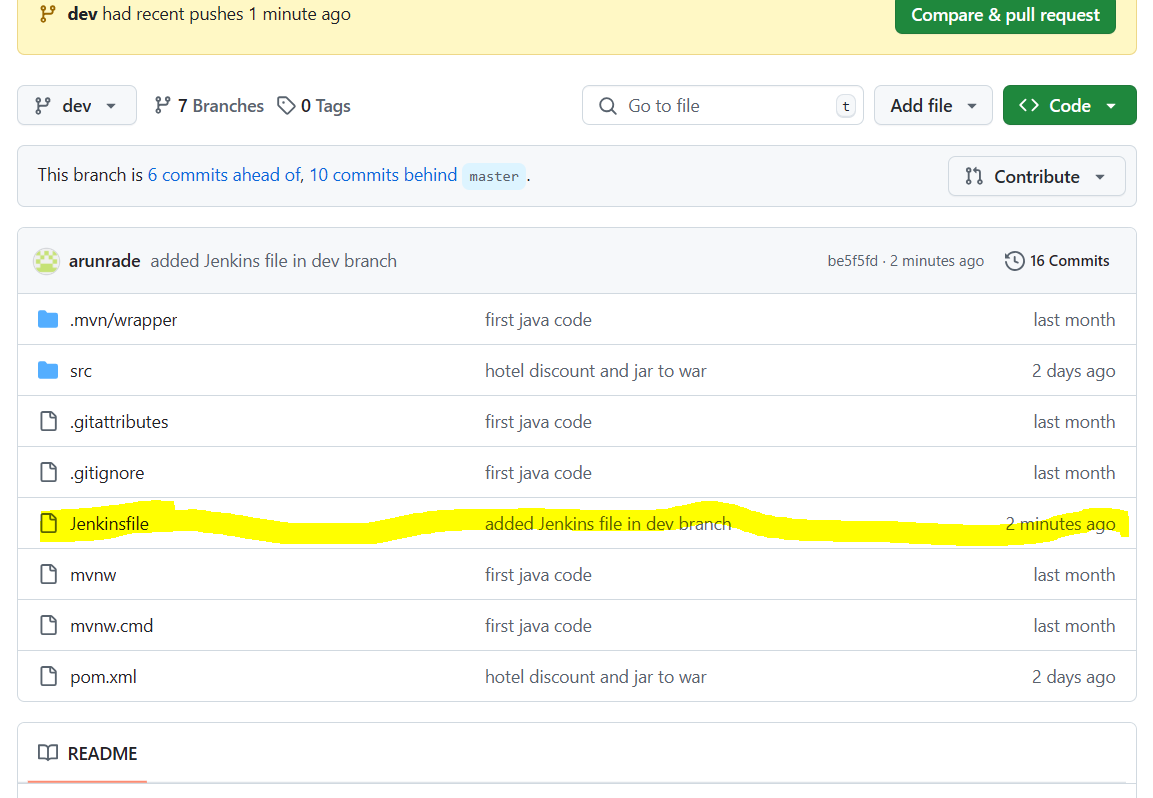
**Fire below commands to push the changes to dev remote branch**

**git add –all**

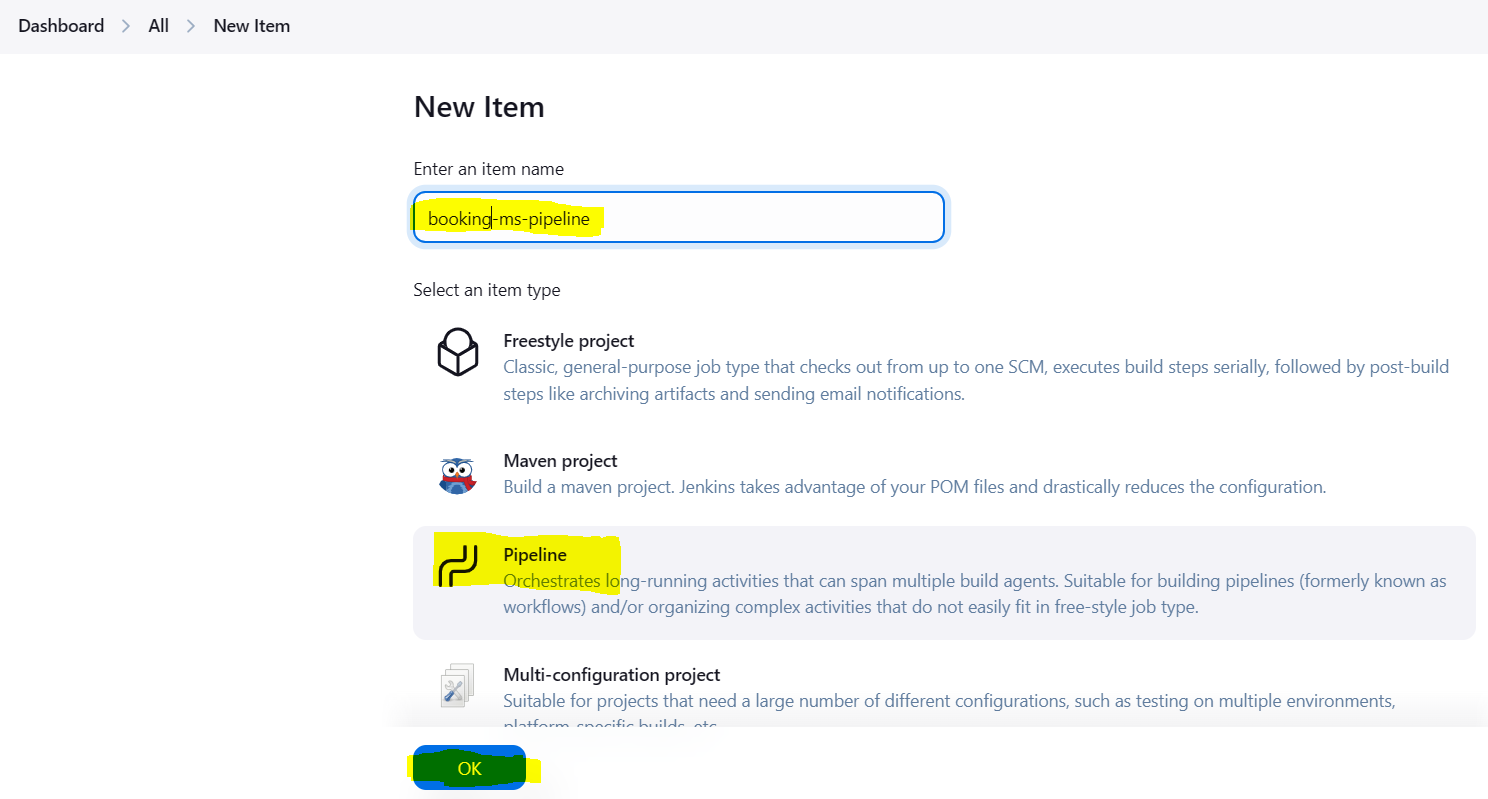
**git commit -m “added JenkinsFile”**

**git push origin dev 🡪 to push changes to dev remote branch**

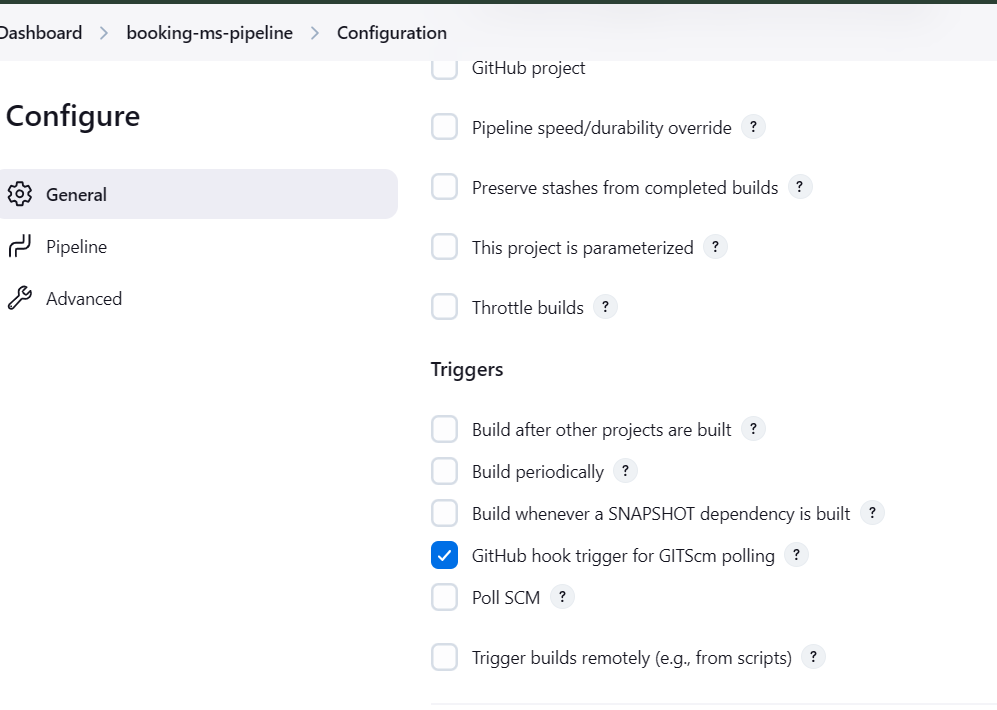




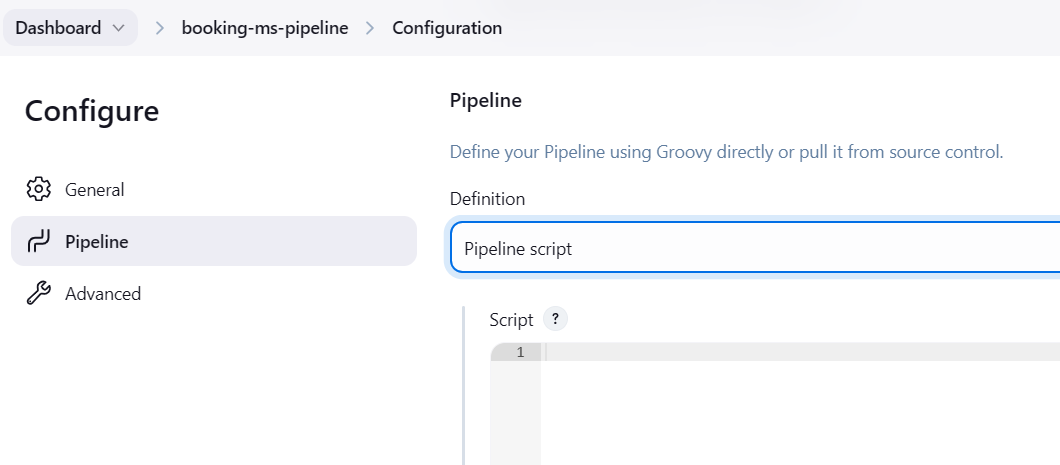
**Now login on Jenkins 🡪 click on New Item 🡪 provide name 🡪 select pipeline project 🡪 click ok to save the project**



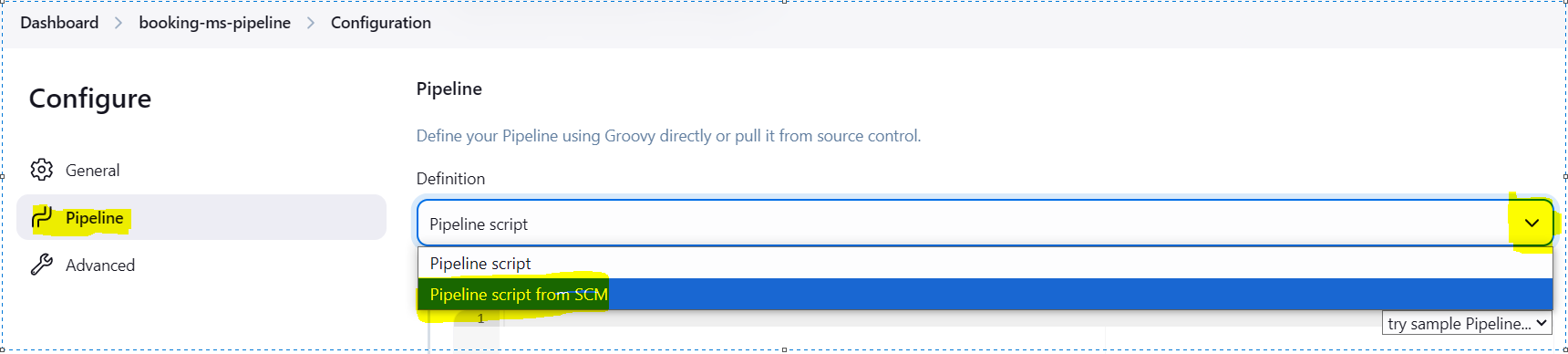
**Goto Pipeline**

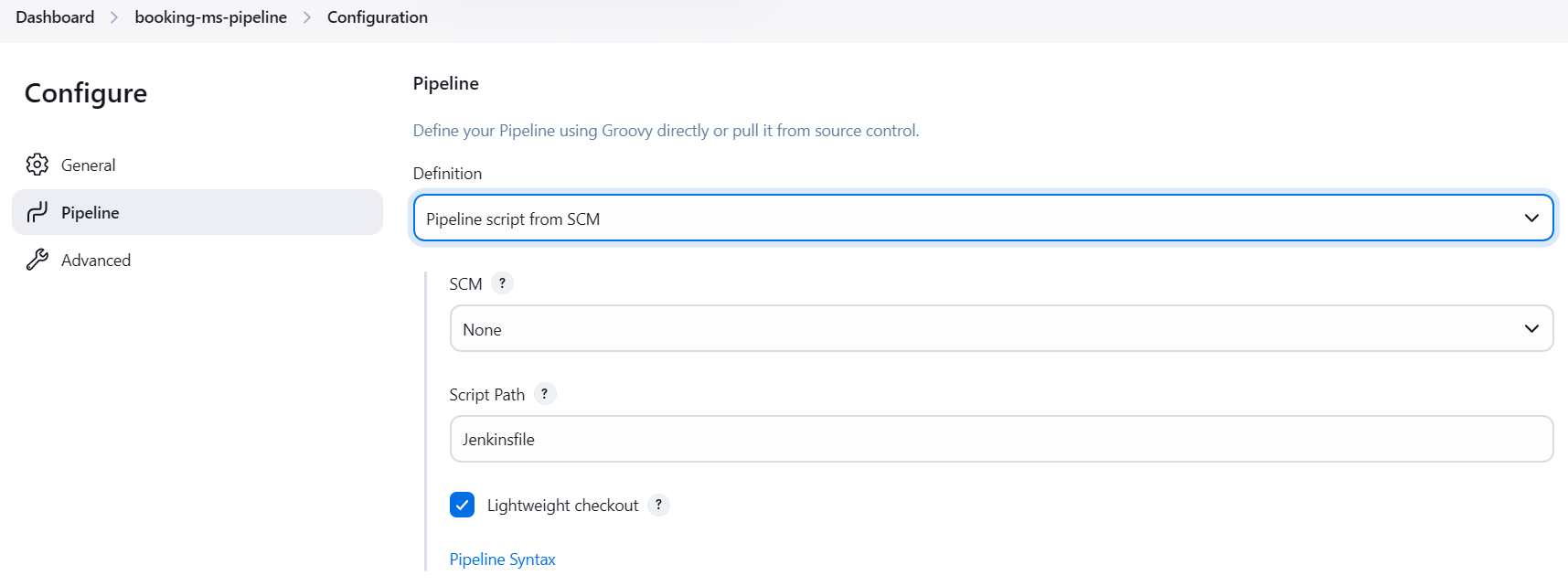


**Webhook addition is mandatory here without this Jenkins job will not trigger**

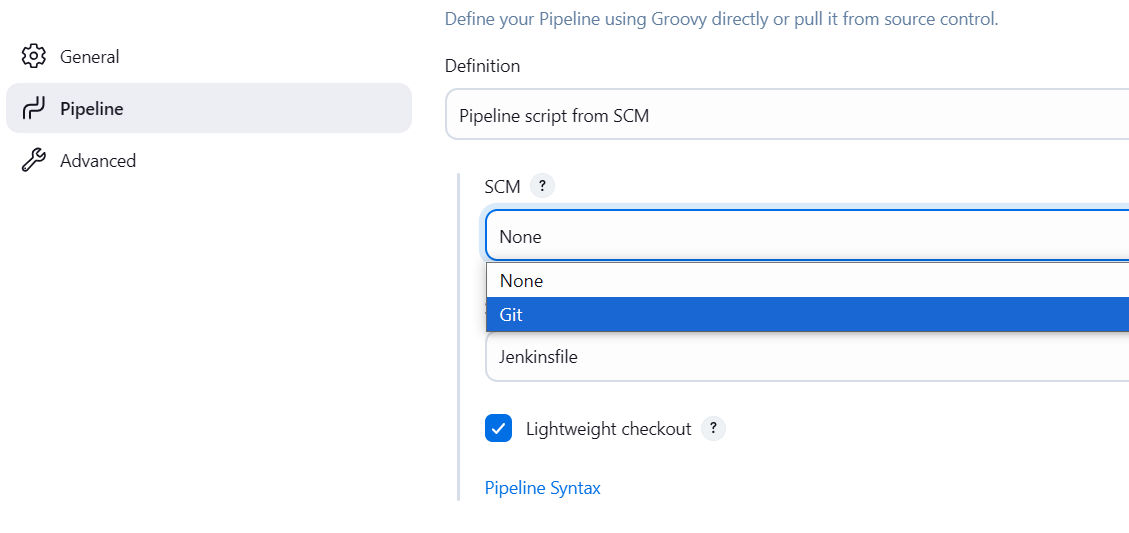


**To select definition as “pipeline script from SCM” click on dropdown option and select pipeline script from SCM.**



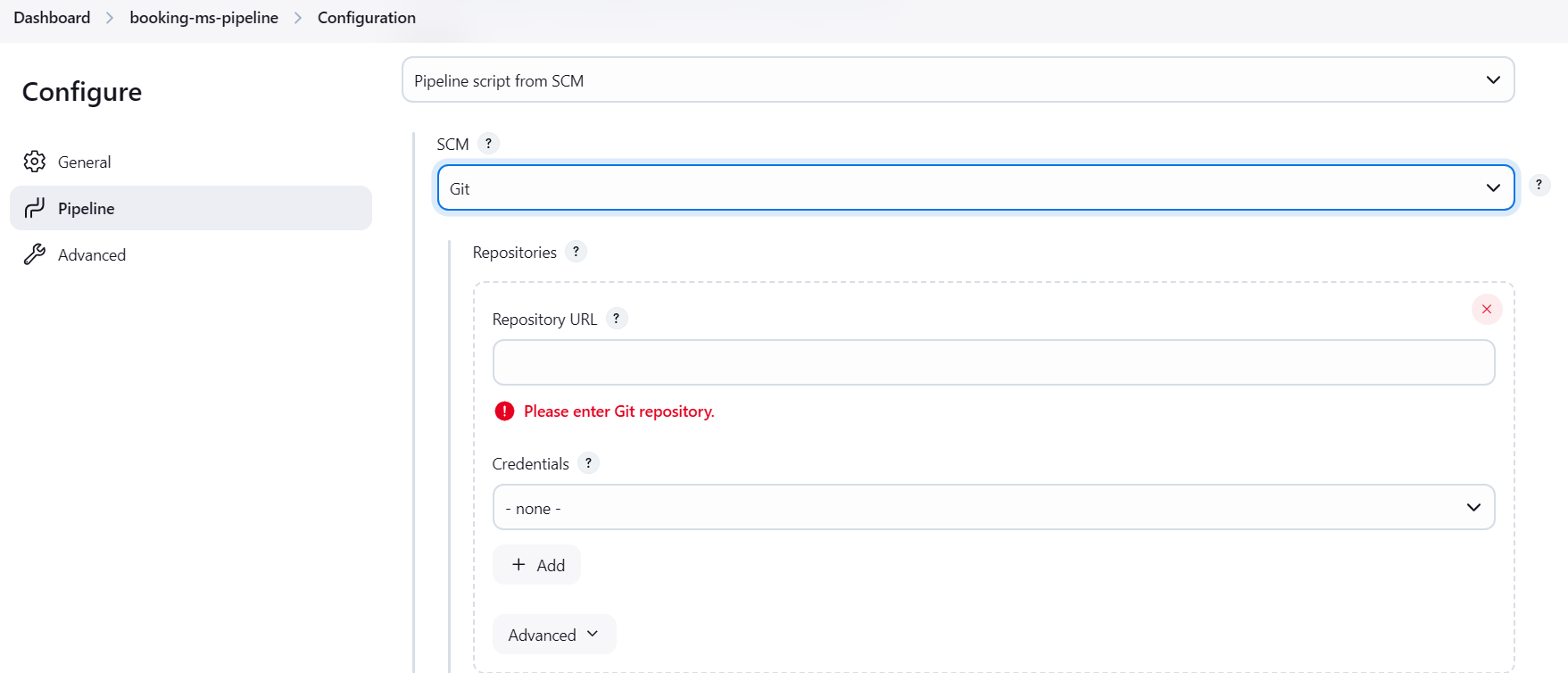


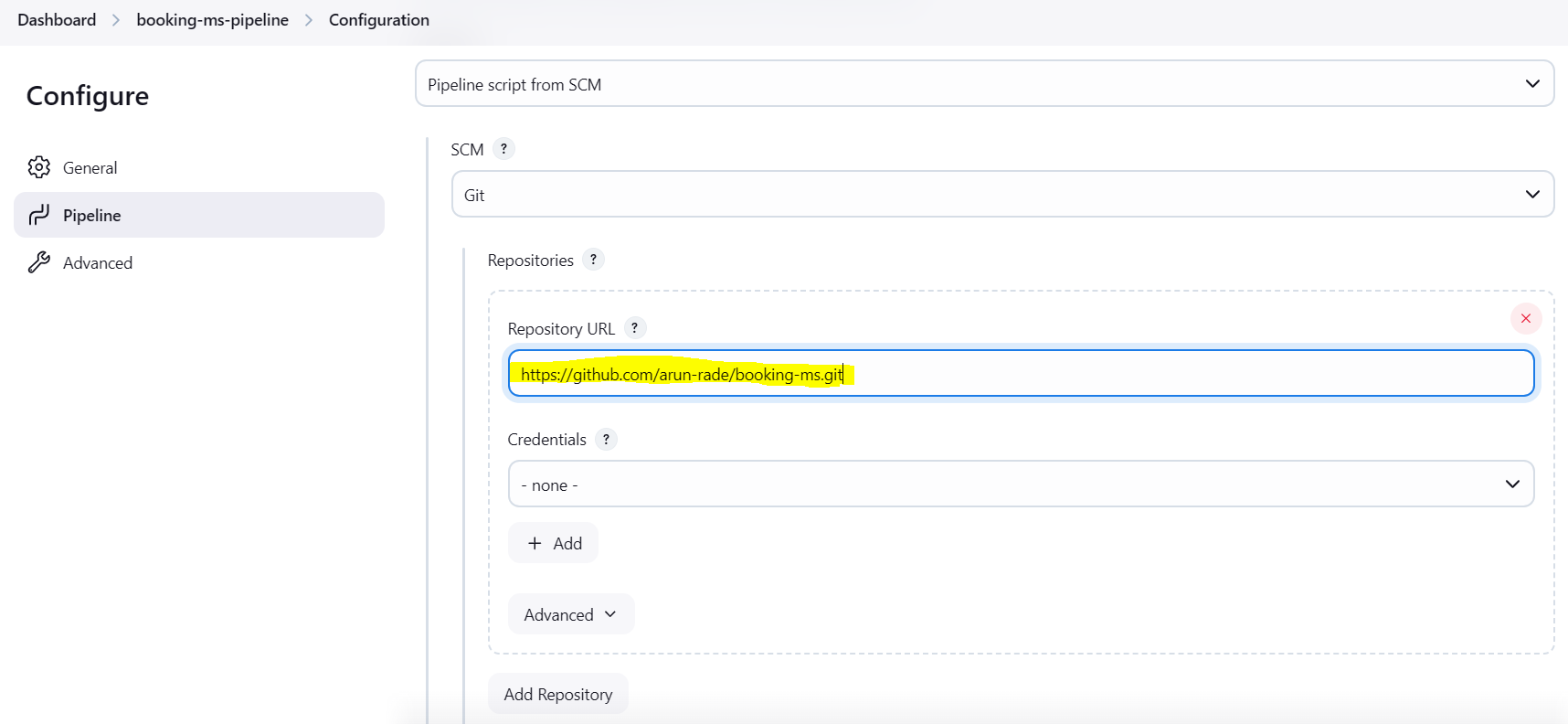
**Select git as SCM**

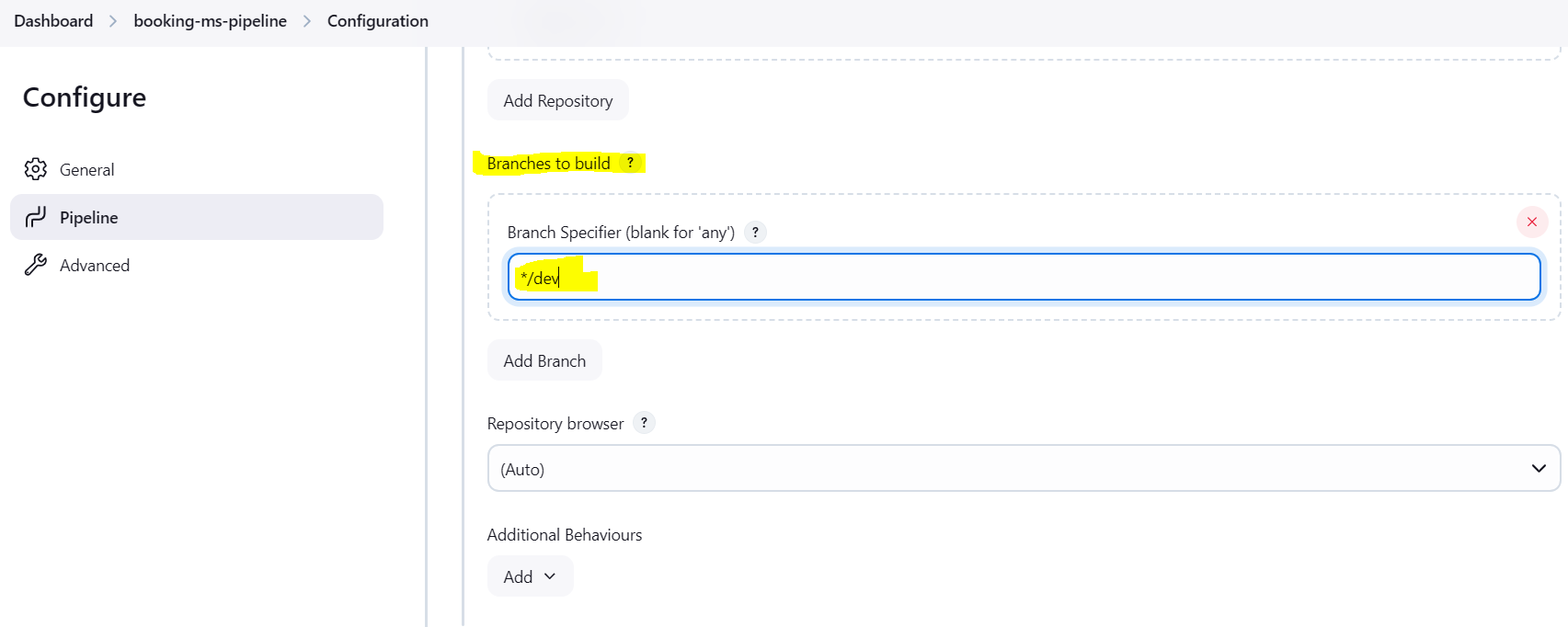


**Add remote repository dev branch URL under “Repository URL”**

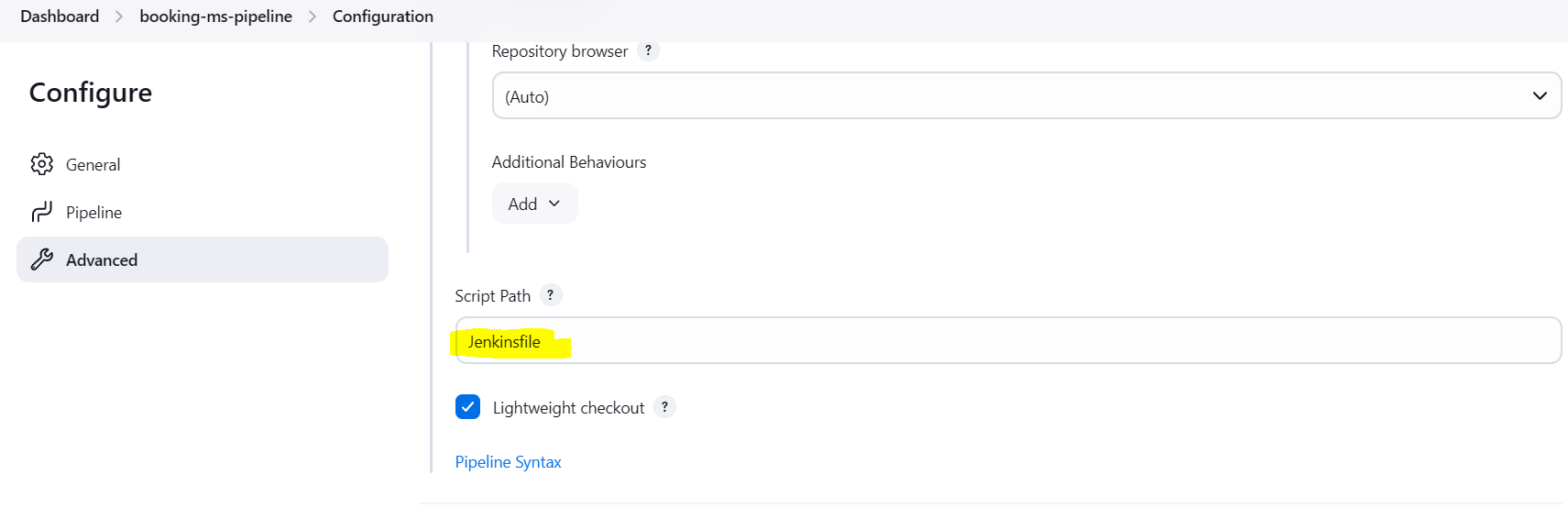
**Also mention branch name where you want to make changes and Jenkins job will trigger.**



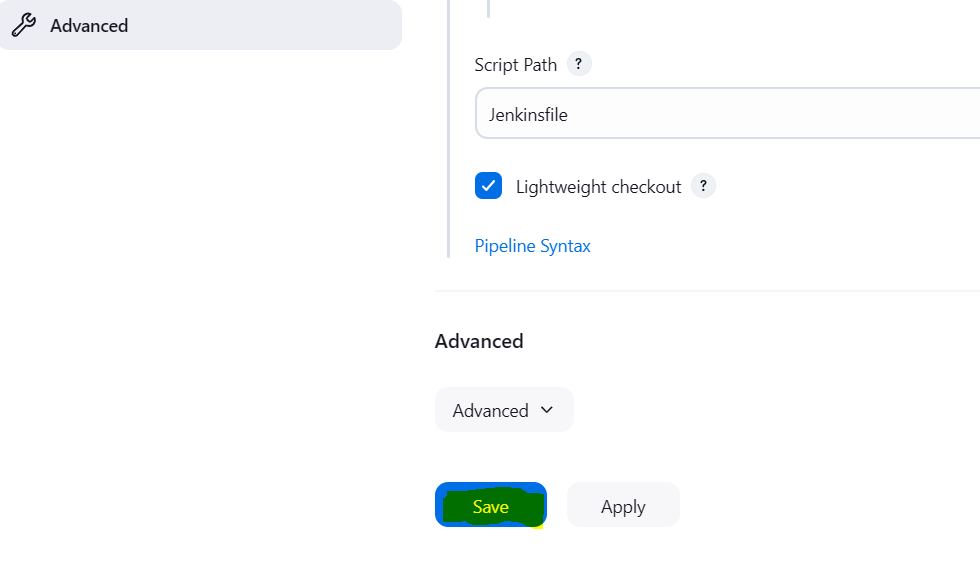




**Jenkins job will take commands or instruction from Jenkisfile which is mentioned in Advanced 🡪 script path**



**Save the changes**



**Once this project setup gets complete. Go to IntellijIdea and any java file class file for e.g. I have added vacation.java file**

**Save the changes and run the code locally**

**mvn clean compile**

**mvn clean test**

**mvn clean package**

**mvn spring-boot:run**

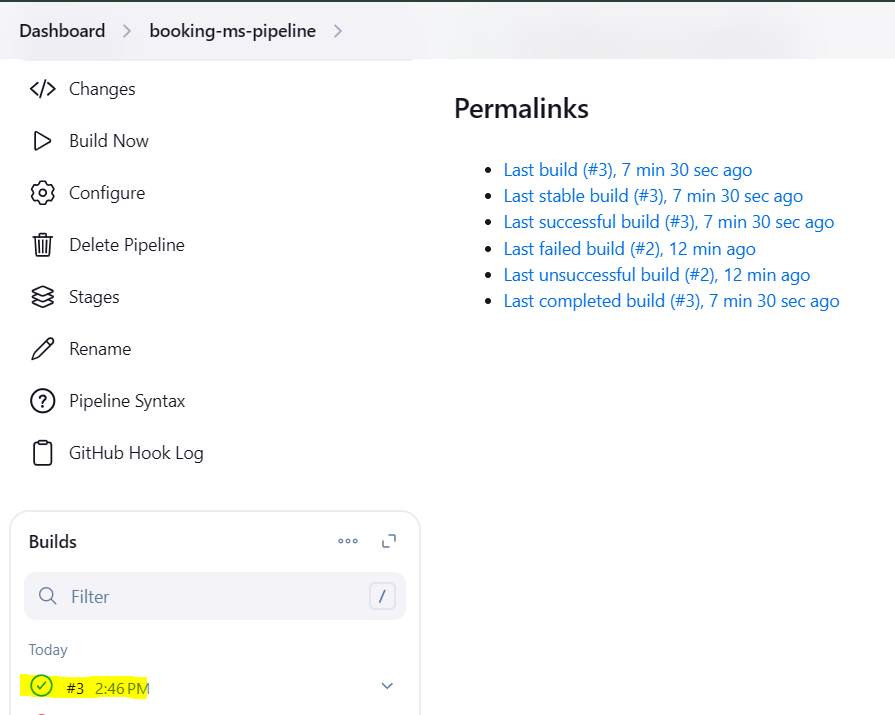
**once this gets success, we need to push the code to remote repository dev branch**

**git add –all**

**git commit -m “new java code file added”**

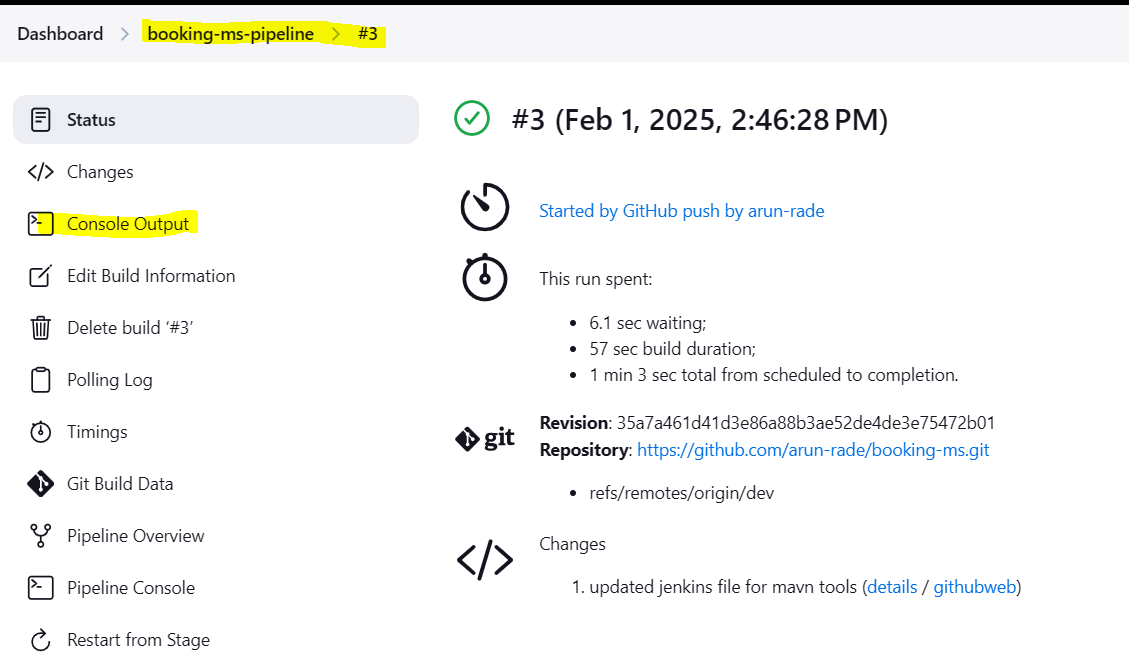
**git push origin dev**

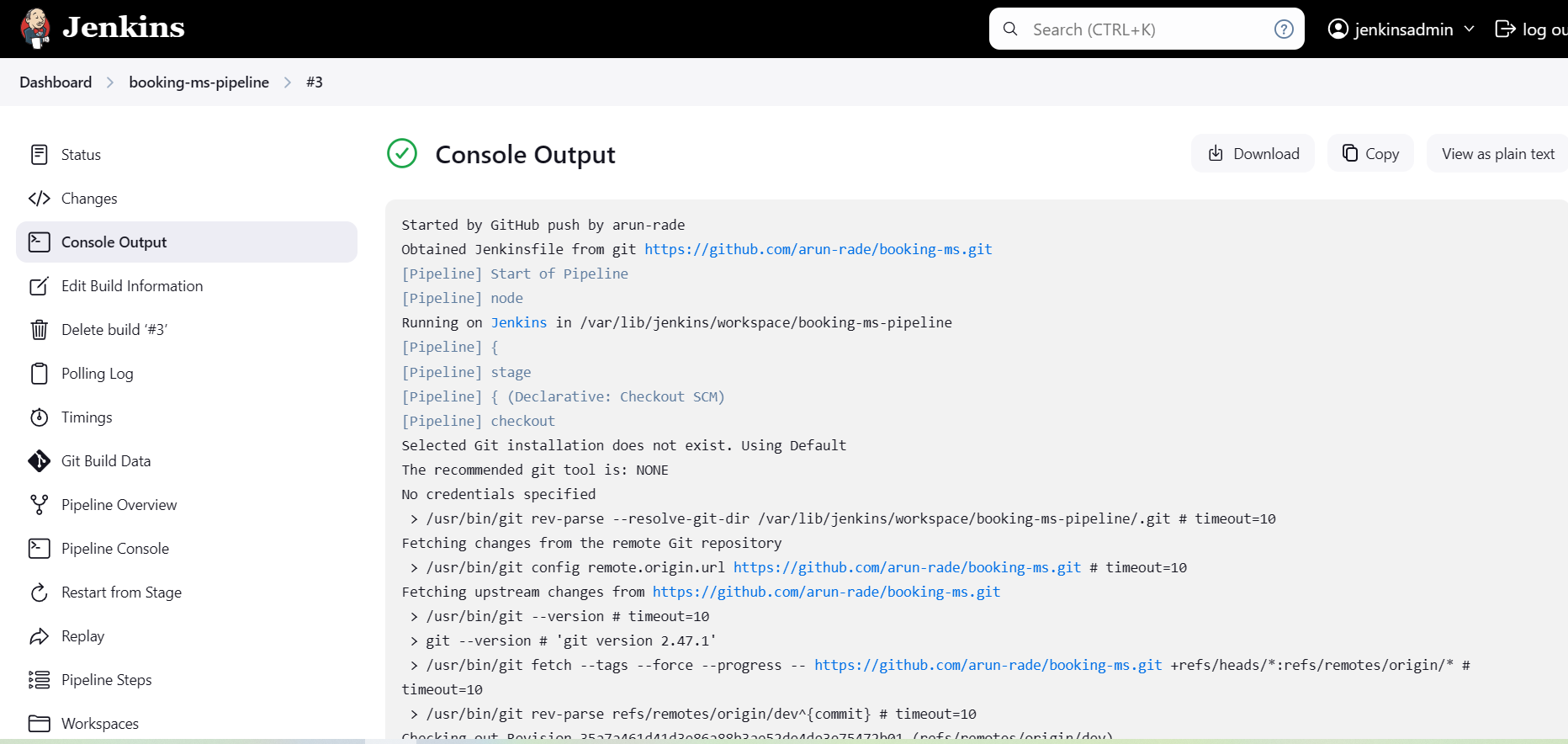
**Once build pushed to remote dev repository Jenkins job will trigger automatiocally.**



**You can check the console output for the same.**

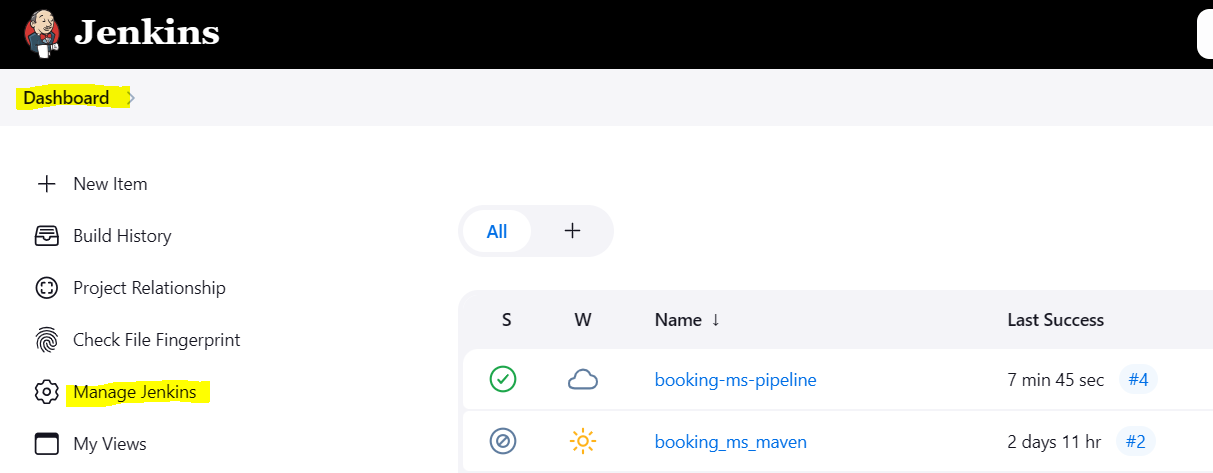
**Click on the success build and click on Console Output**



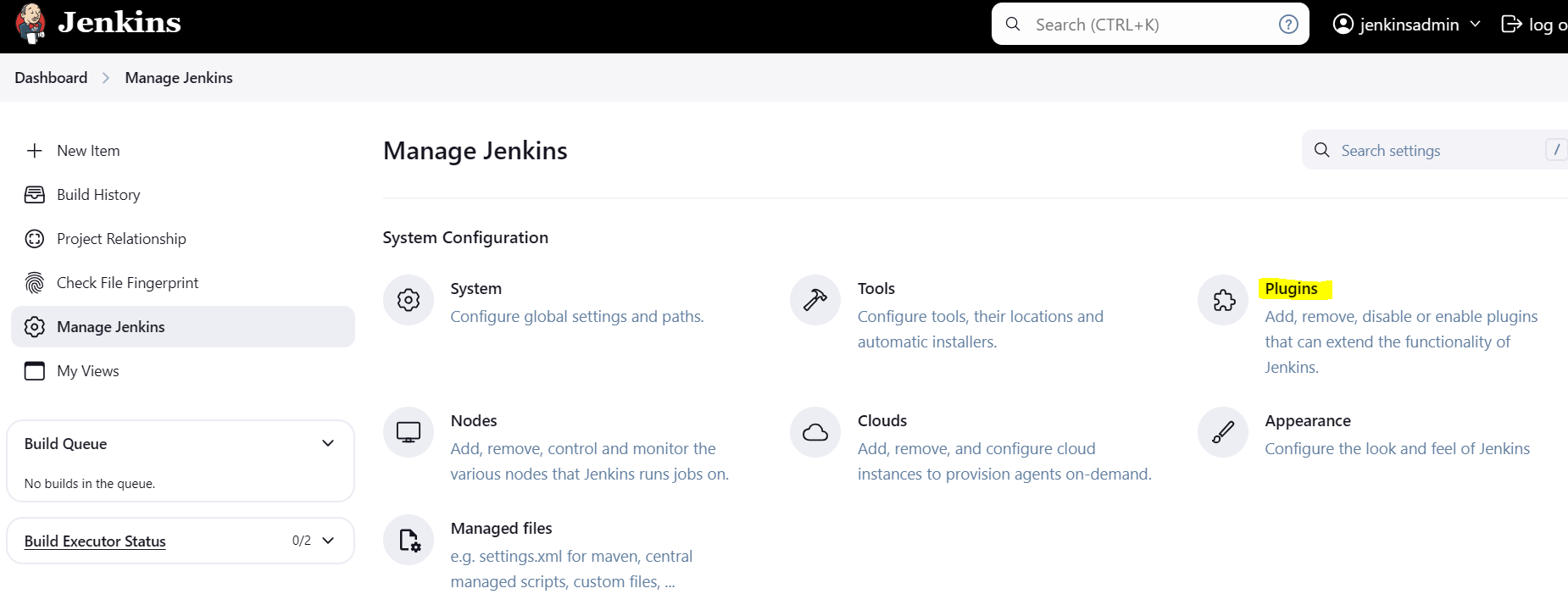


**To add stages in Jenkins job please follow below steps**

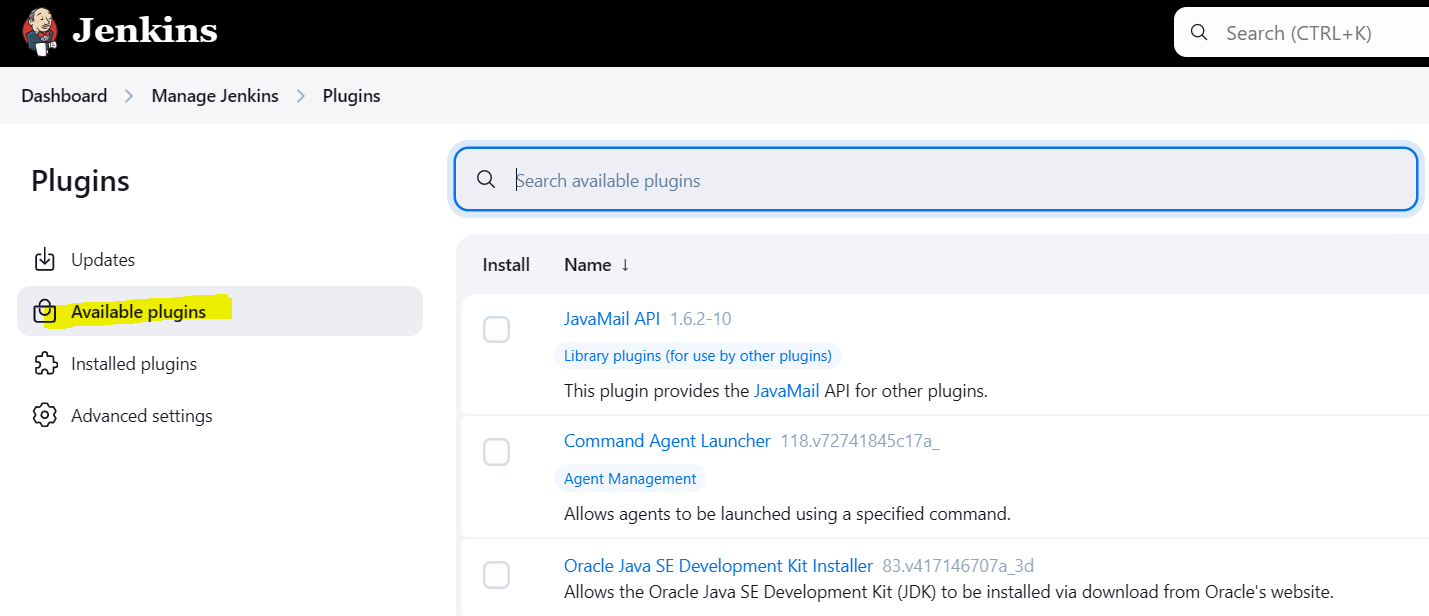
**Click on dashboard 🡪 Click on Manage Jenkins**



**Click on Plugins**



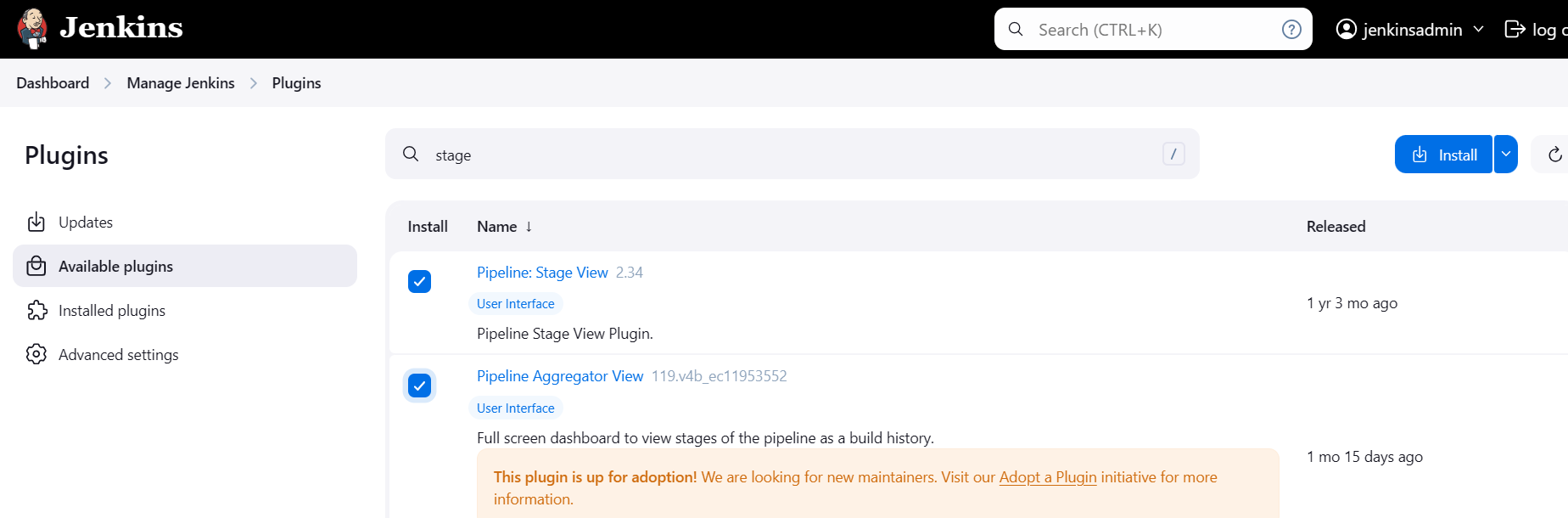
**Click “Available Plugins”**



**Search for Stages and select below plugins**

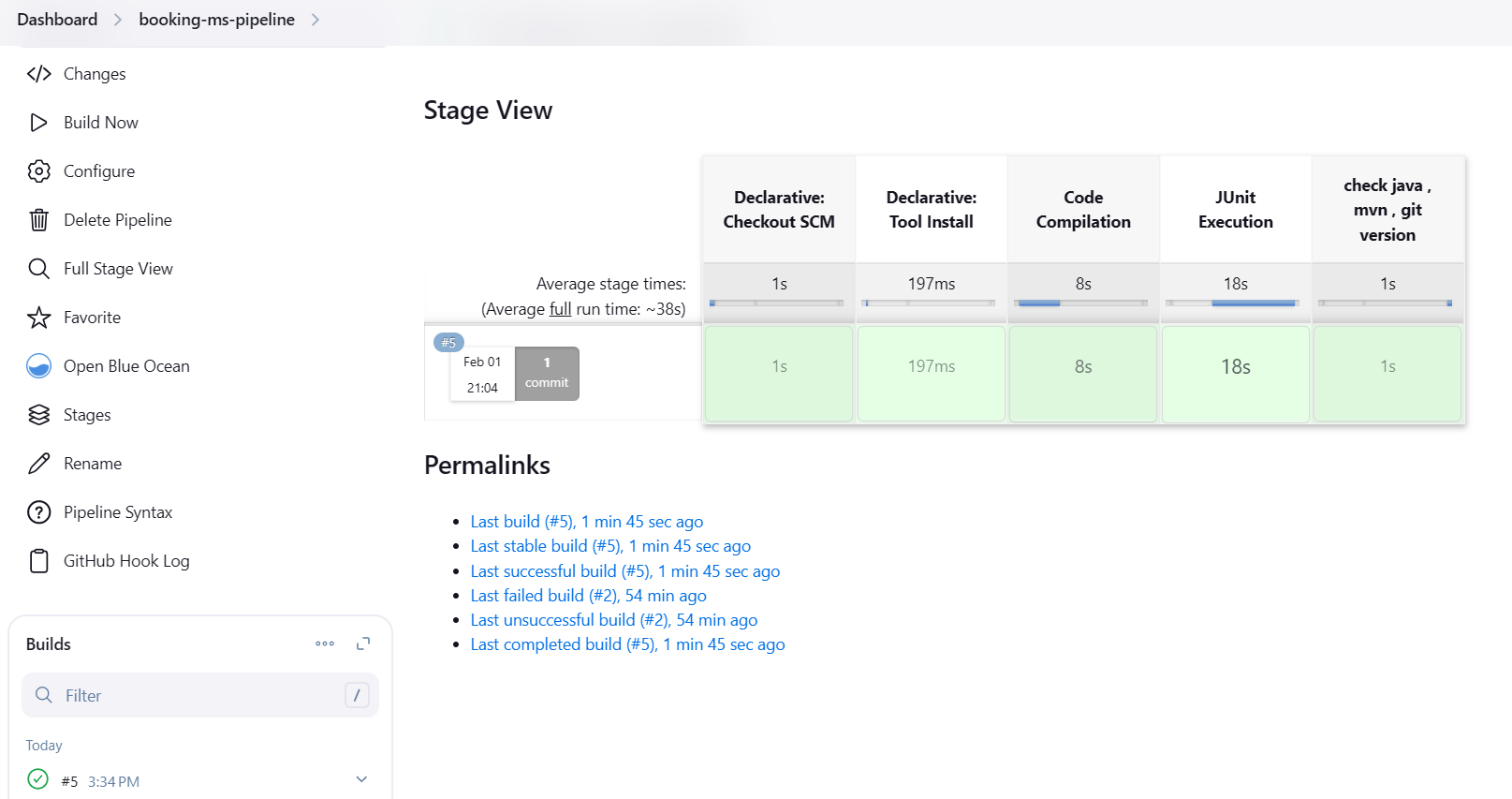
**Pipeline Stage: View**

**Pipeline Aggregator View**



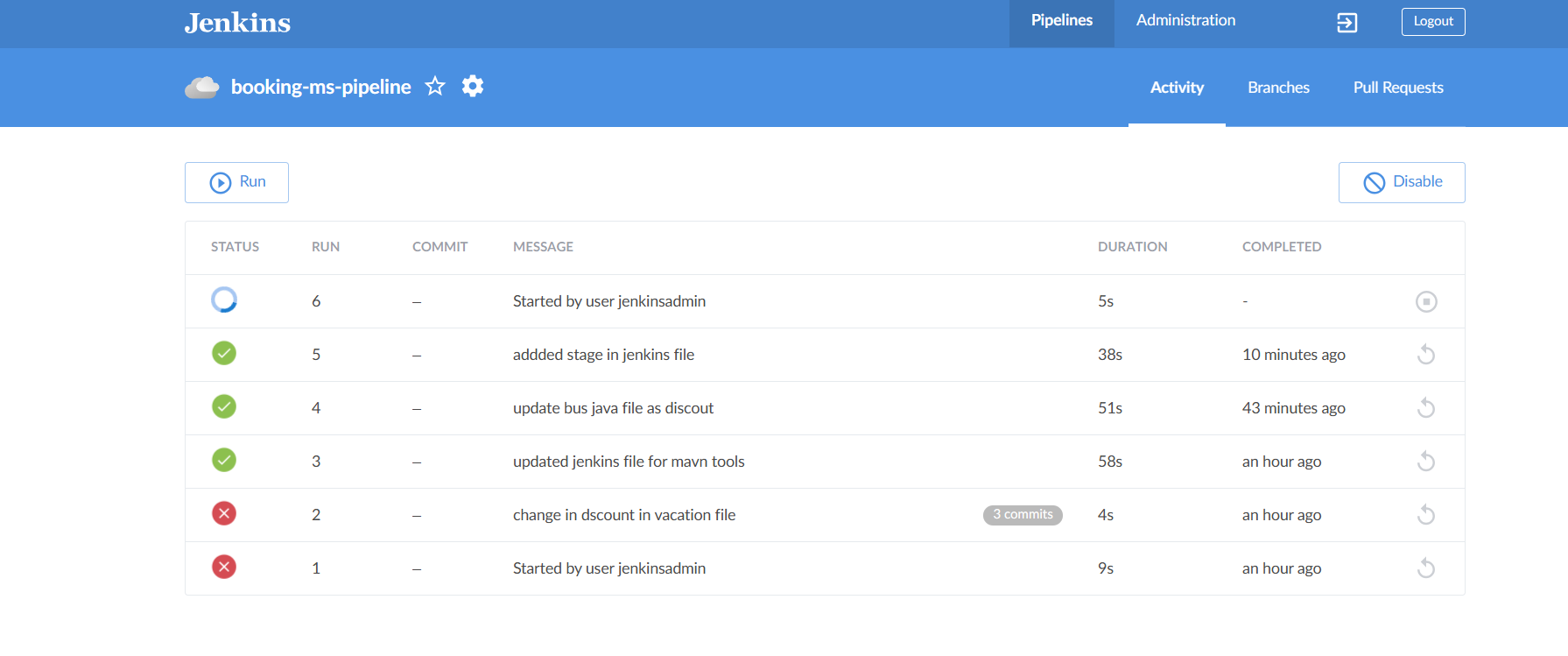
**Also install Blue Ocean plugins (Select all available Blue Ocean Plugins)**

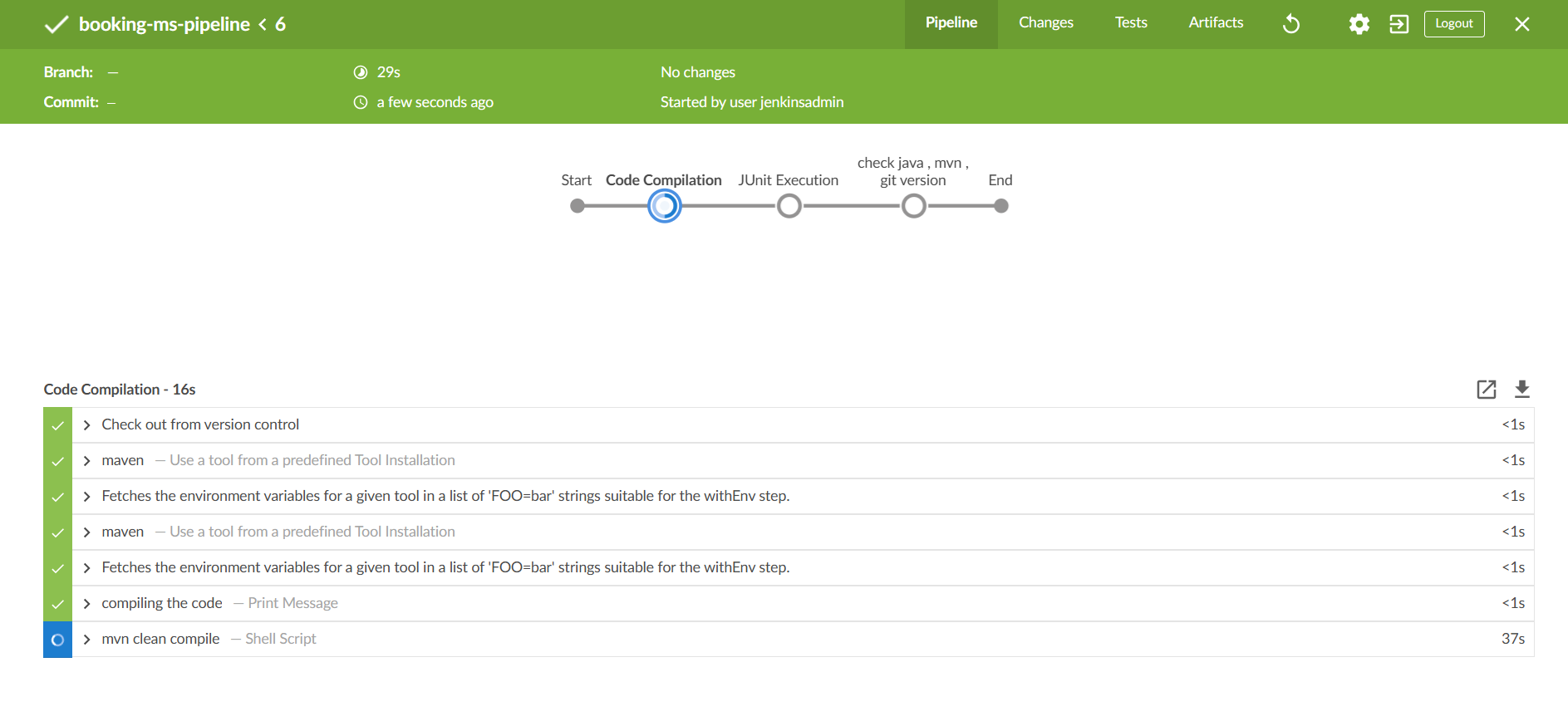
**So next Jenkins job run below stages we can see as per Jenkins file stages**



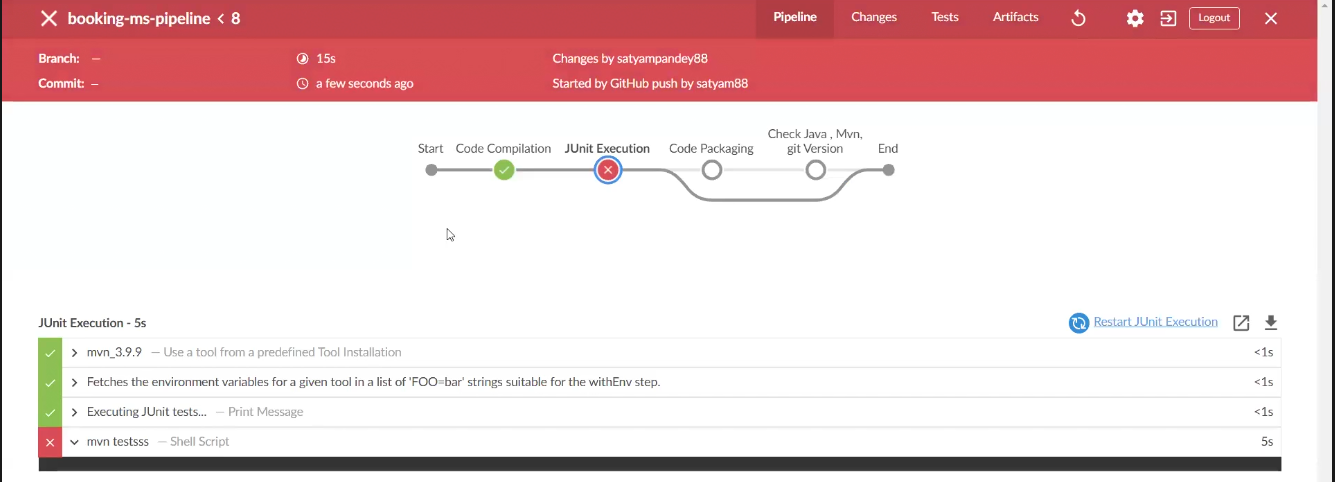
**Blue Ocean View**

**stage('check java , mvn , git version') {  
 steps {  
 echo 'version check the code'  
 sh 'git --version; java --version; /opt/apache-maven-3.9.9/bin/mvn --version'  
 }  
}**



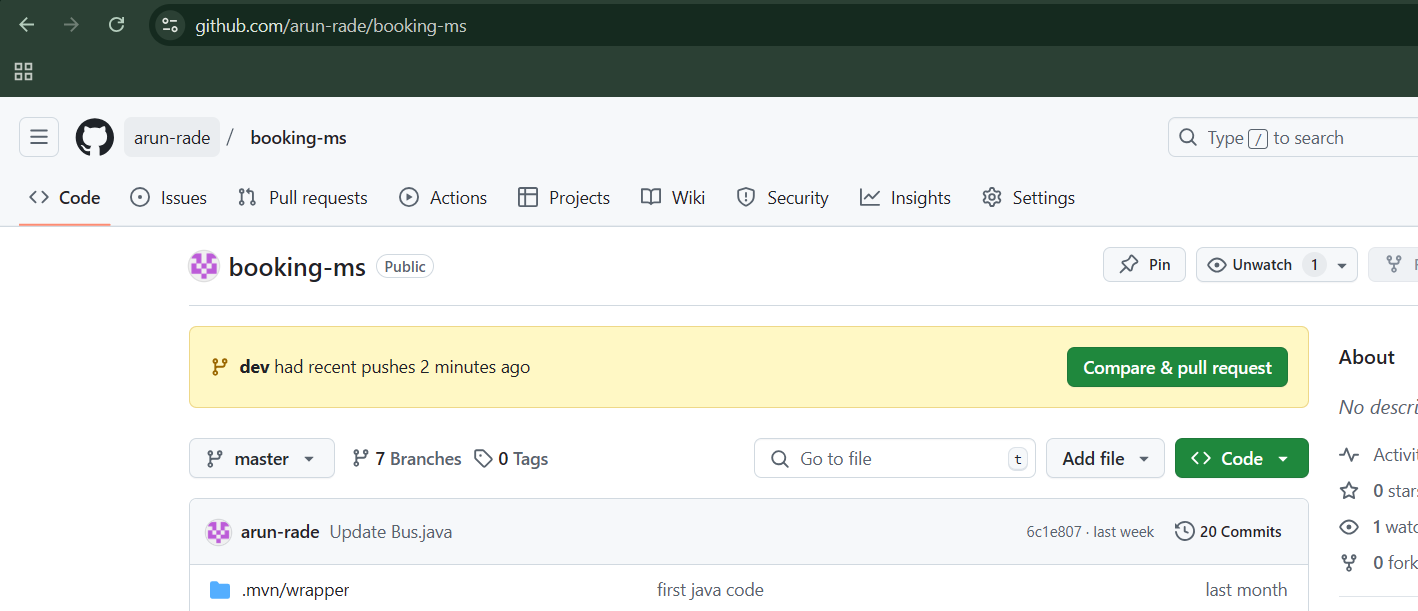


**If any stage get failed, next steps will be skipped**



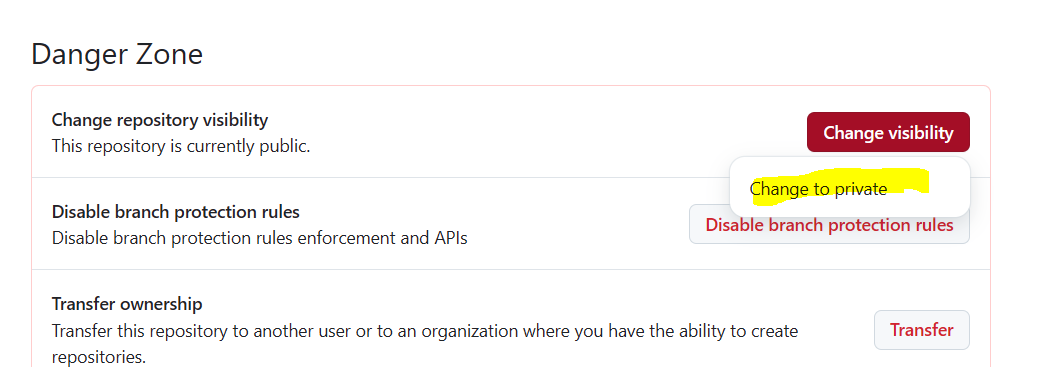
**Make repository as private to add credentials.**

**Goto git hub and make repository to private**

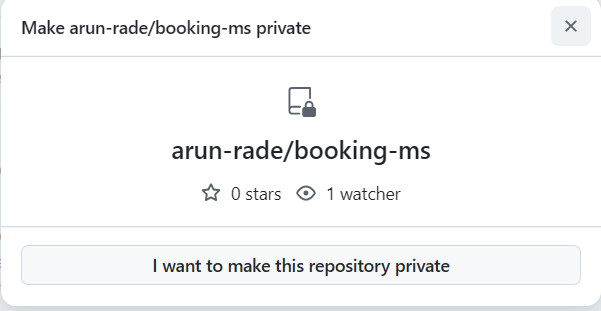


**Click on setting button**

**Goto Danger Zone and click on Change visibility button and select change to private**

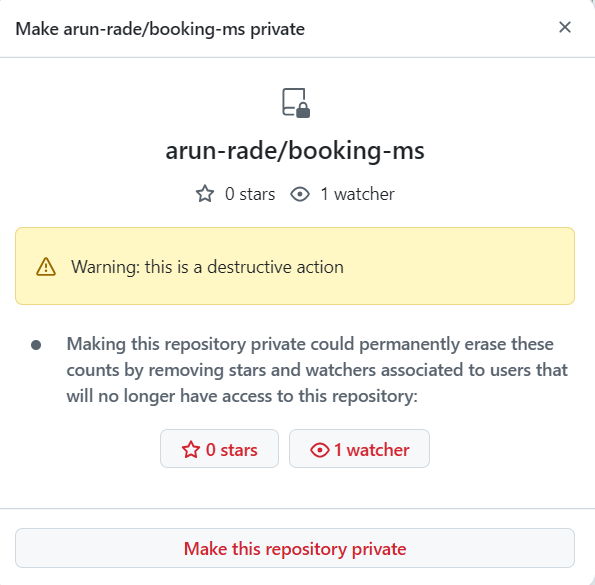


**Click on “I want to make this repository private”**

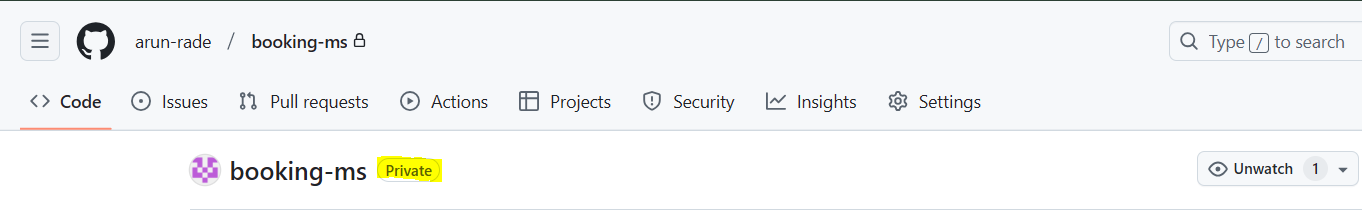


**Read and accept the terms and condition**

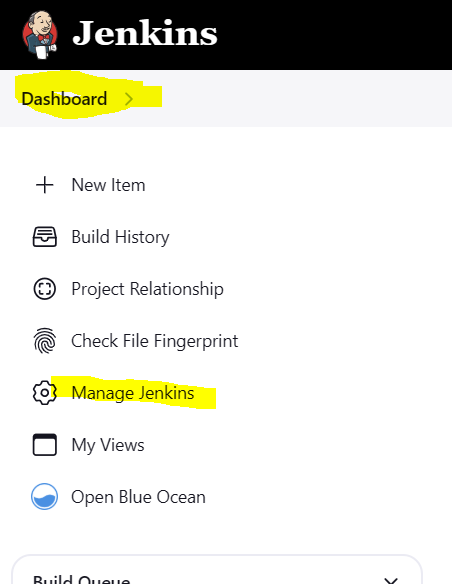
**And click on make this repository private**

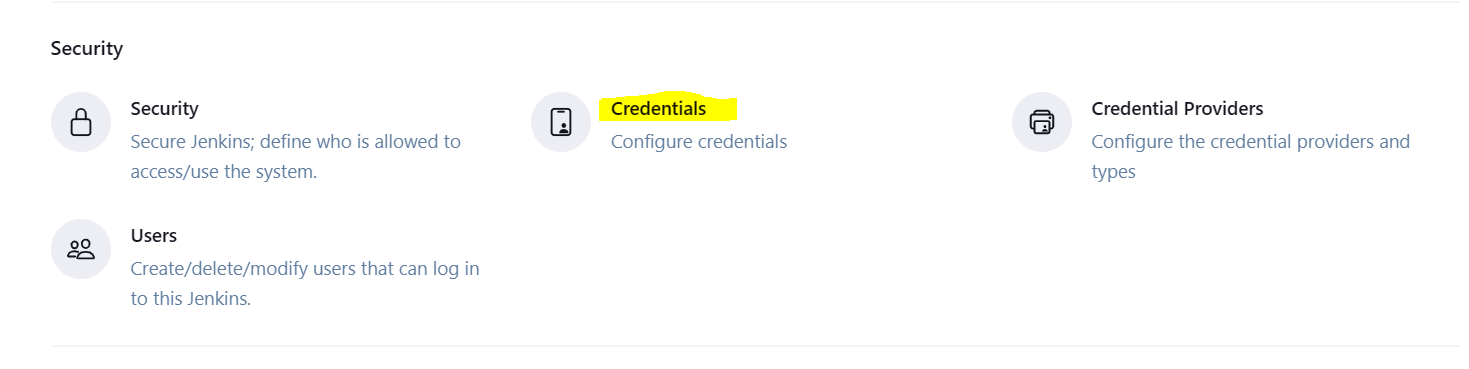


**You can now repository is private**

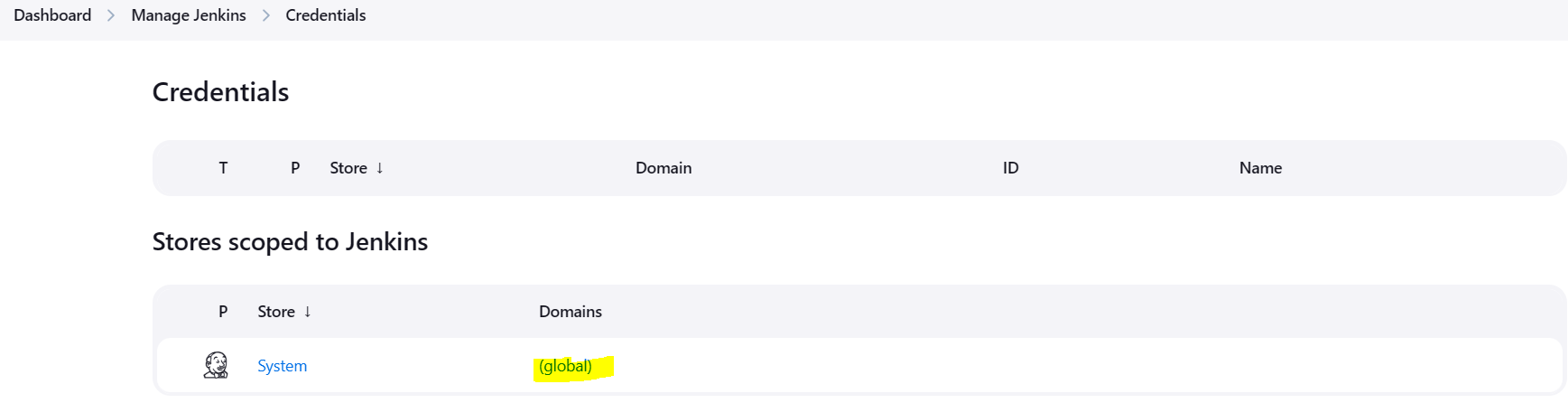


**To add the credential, go to Jenkins 🡪 click on Manage Jenkins 🡪 credential**

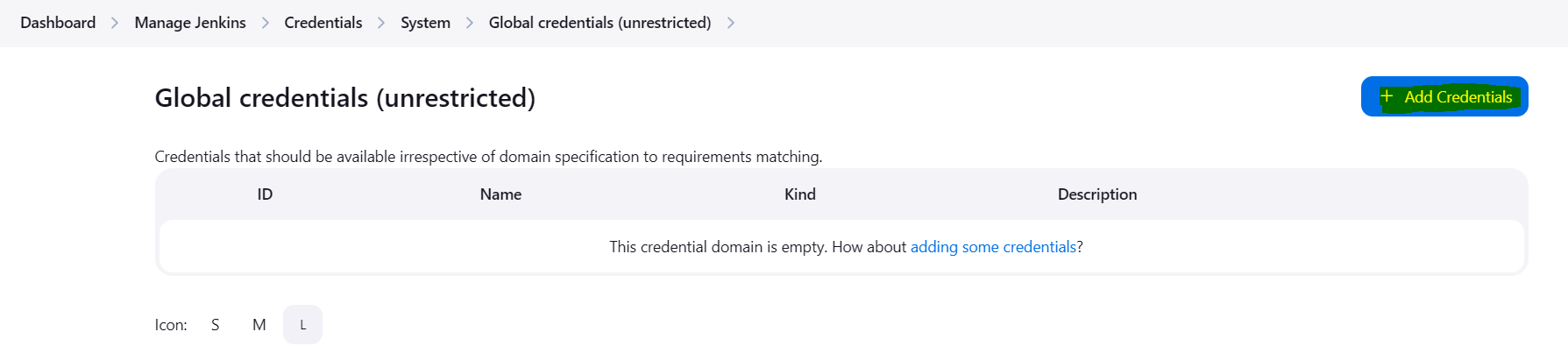


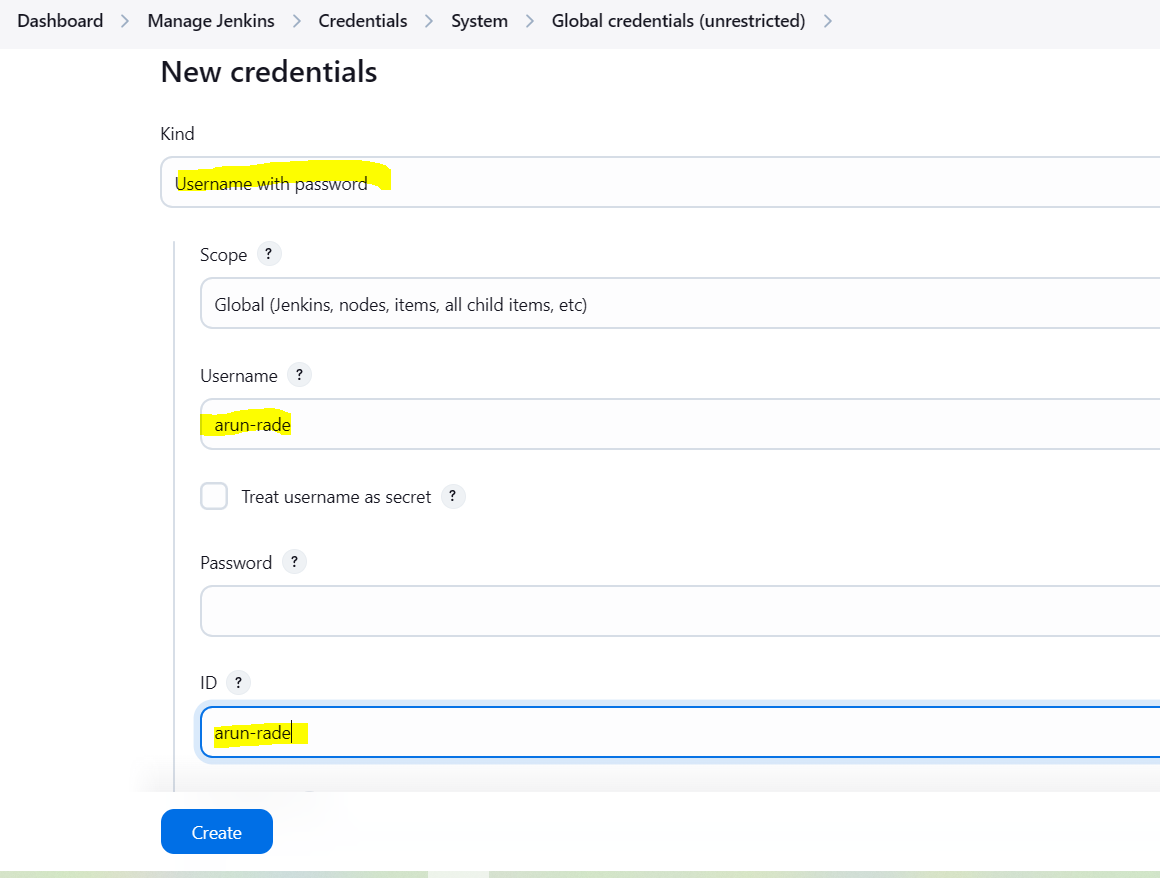


**To add credential, click on global**

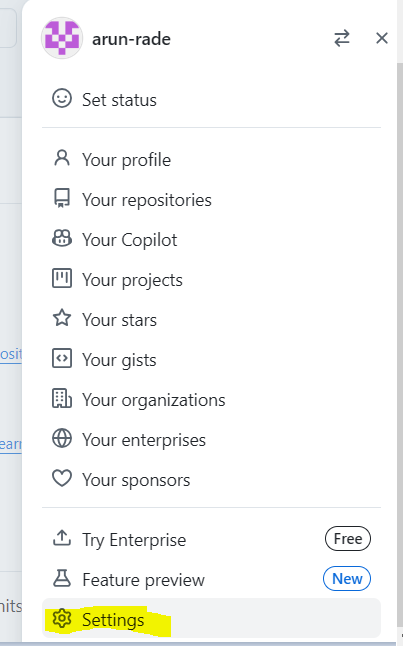


**Click on +Add Credential to add the credential**

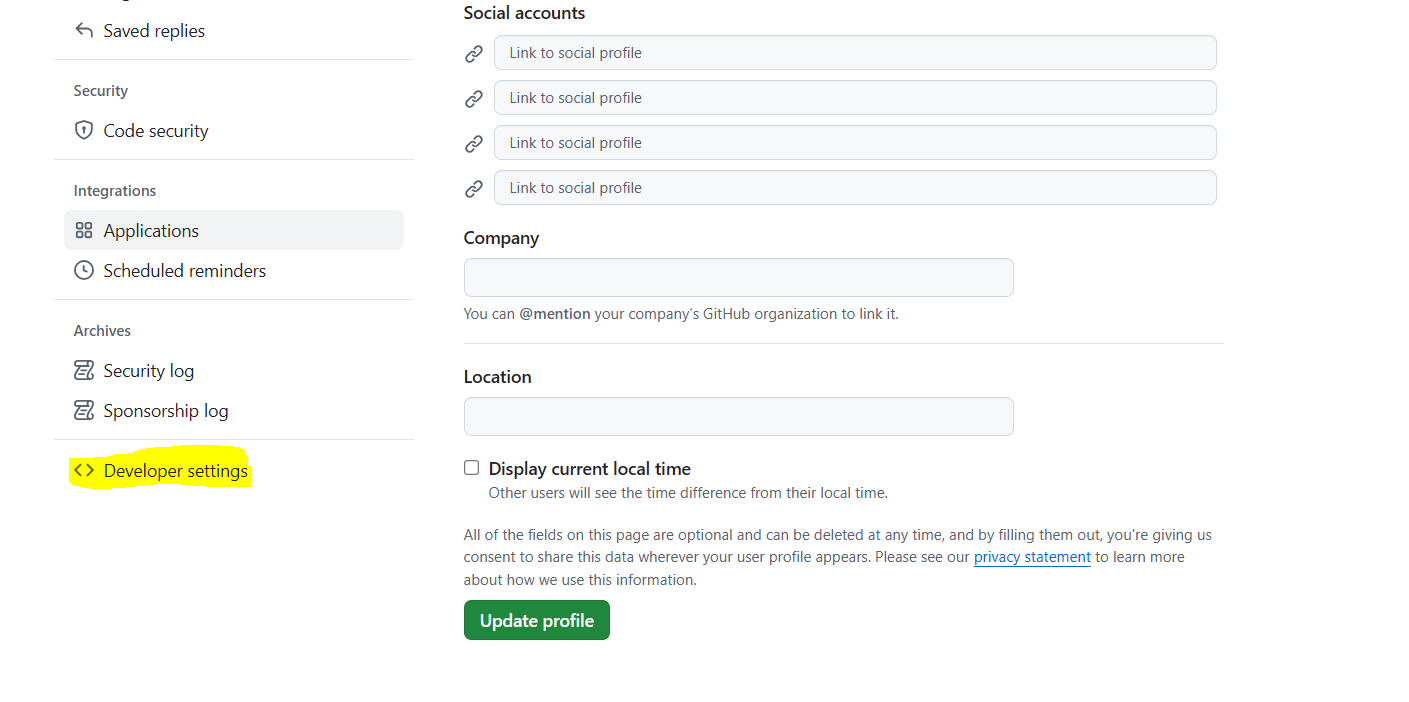




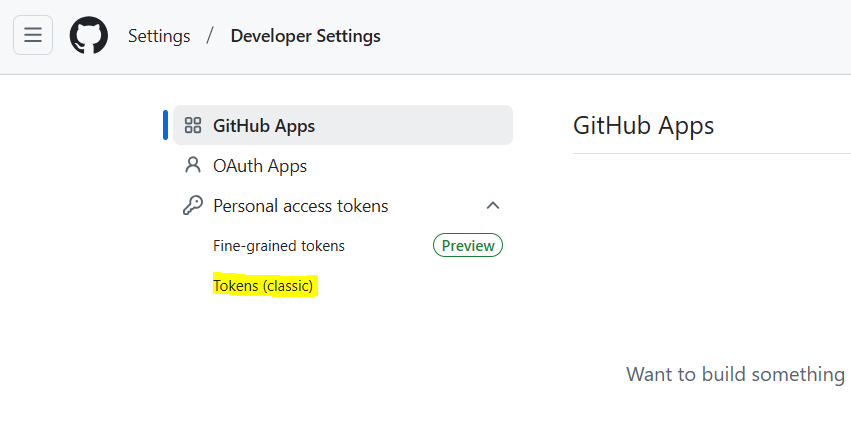
**To create password, we need create token. So we need to login on github 🡪 click on your name - username 🡪click on settings**

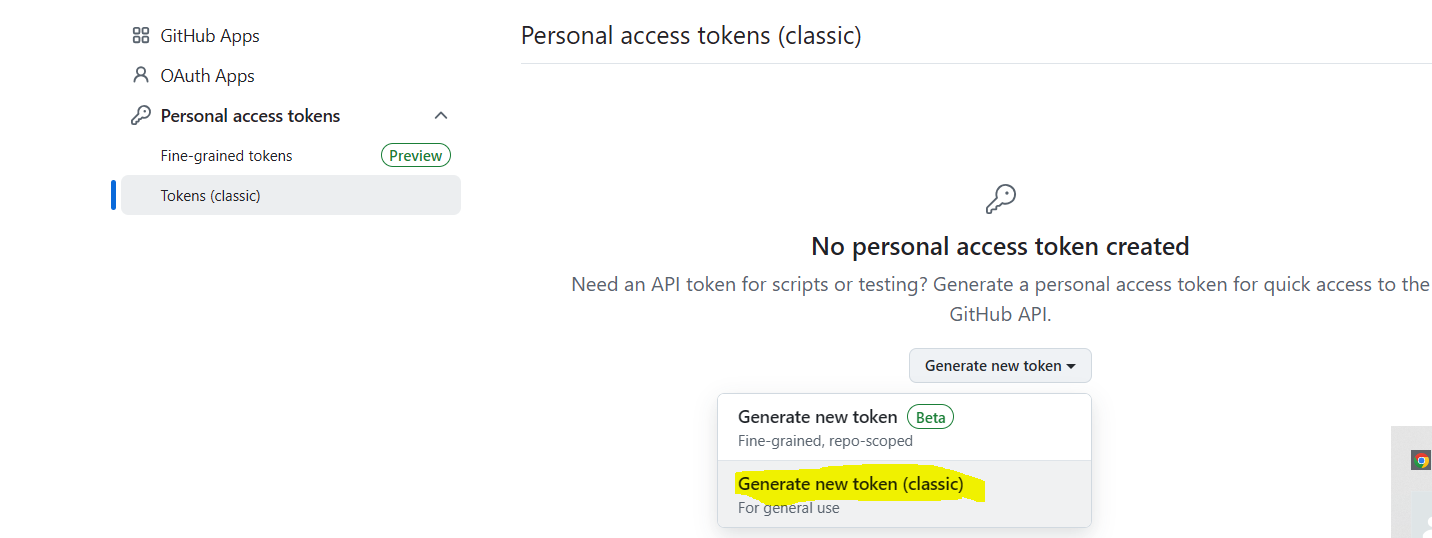


**Click on Developer setting**



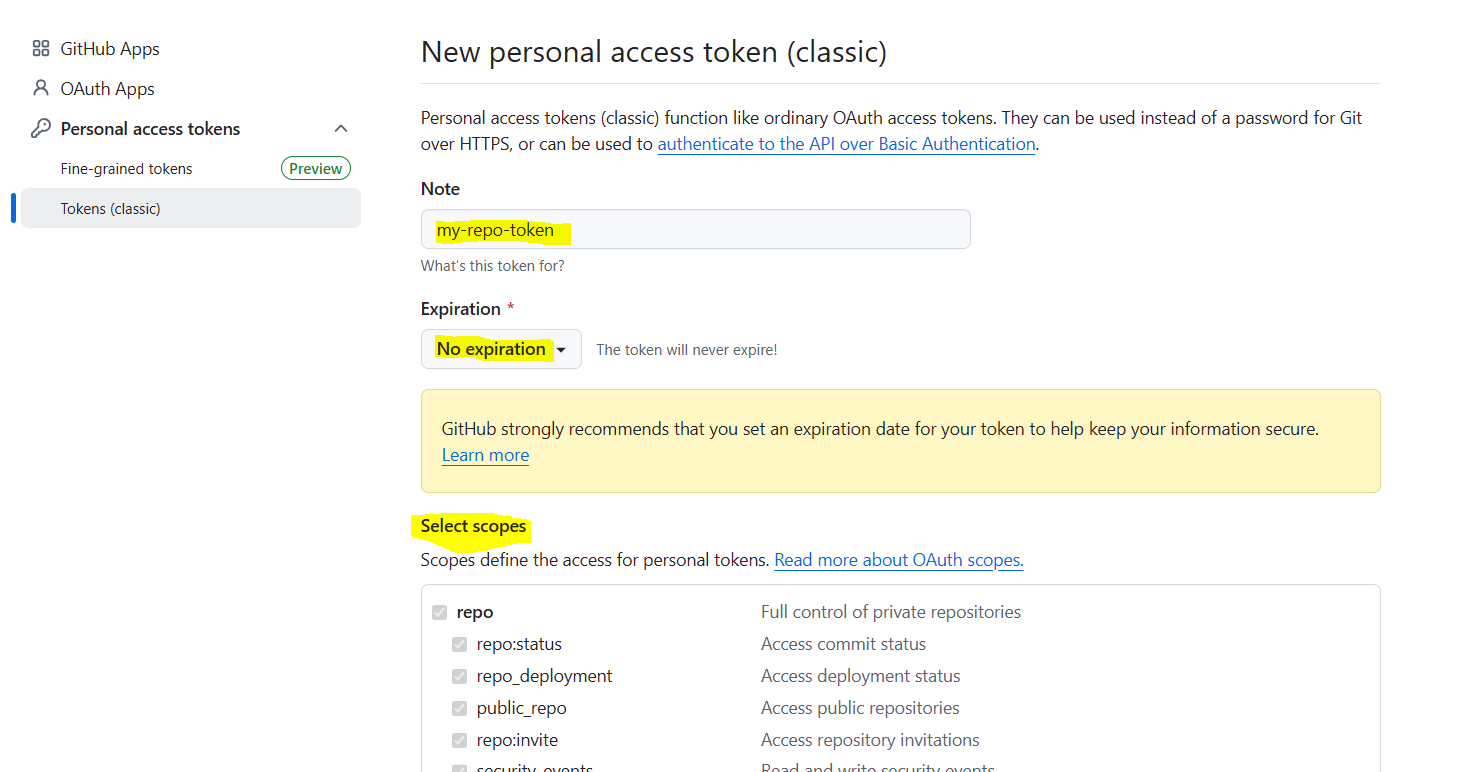
**Click on drop down button near Personal Access token and click on Tokens (Classic) 🡪 click on Generate new token 🡪 click on generate new token (classic)**

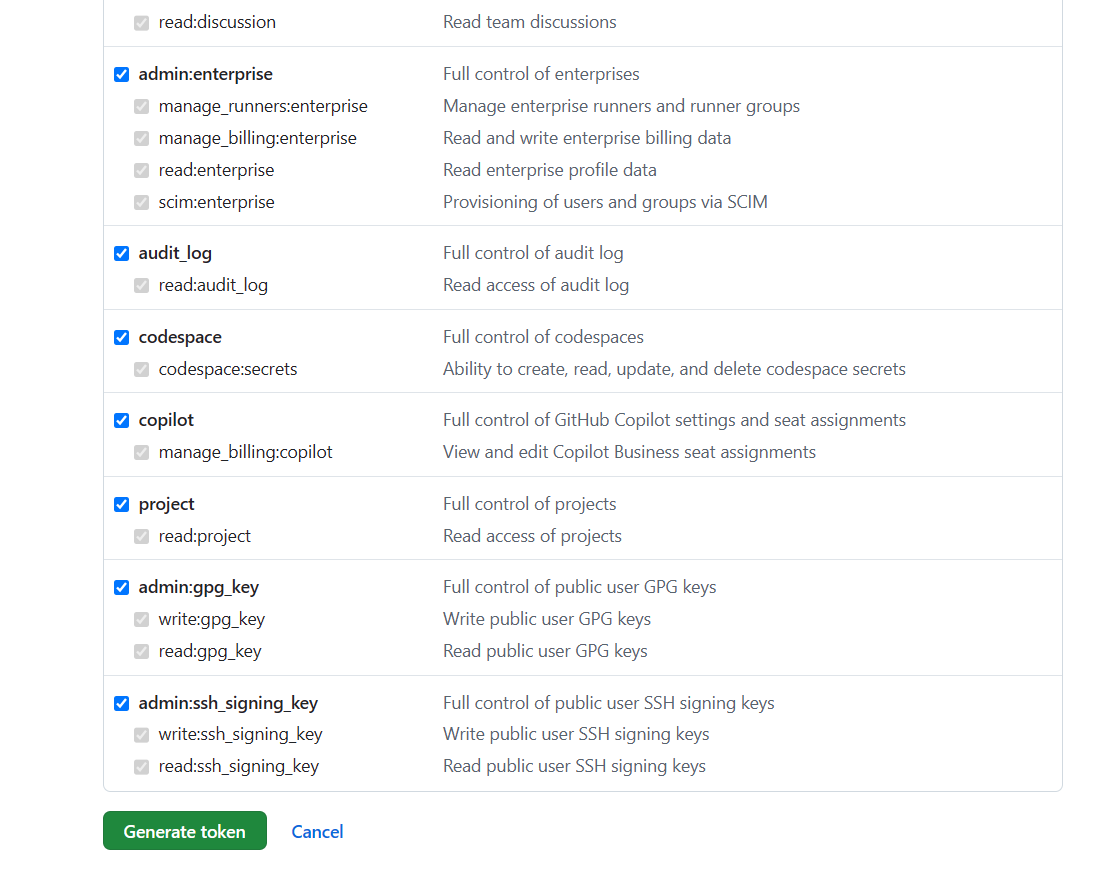




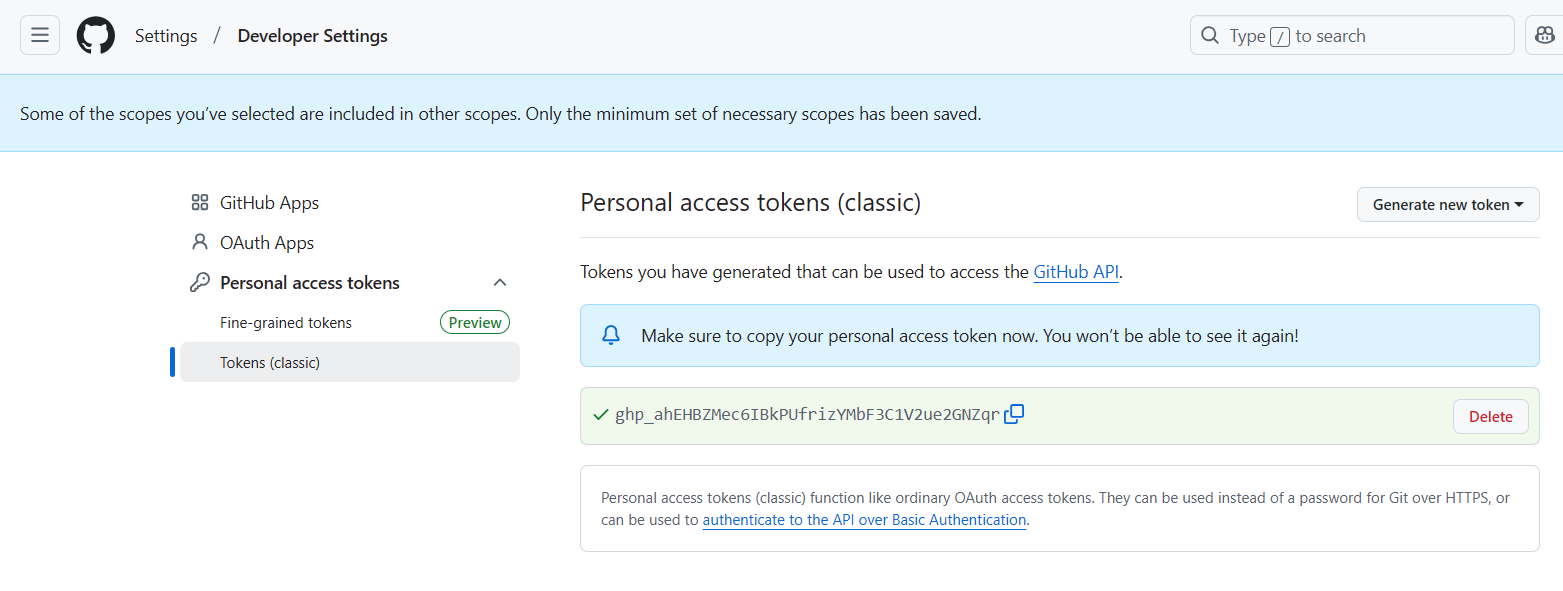
**New window will open, so we can provide name of token, set No expiration and select all scopes**

**And click on generate token**

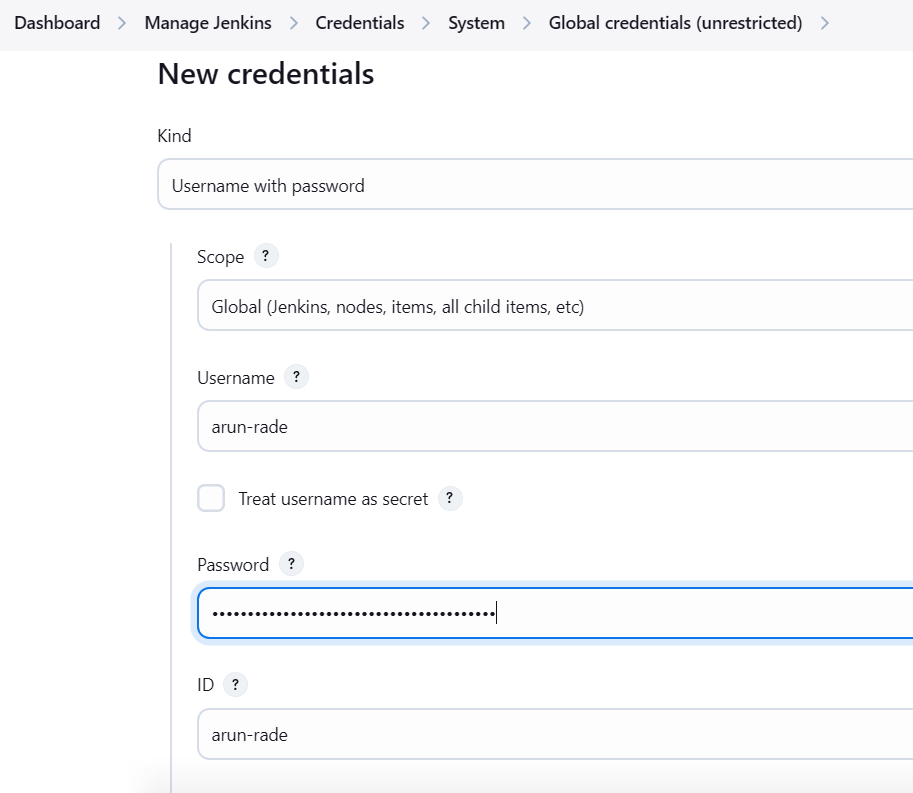




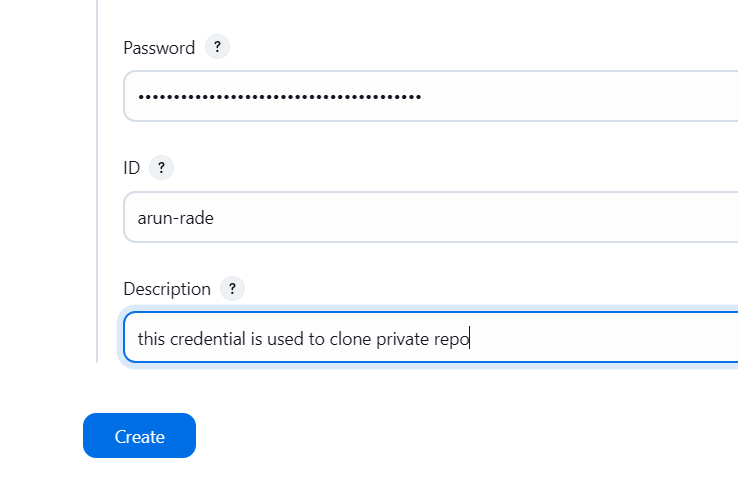
**You can see new token is generated as per below screen shot.**



**Copy this token and paste on Jenkins credential page.**

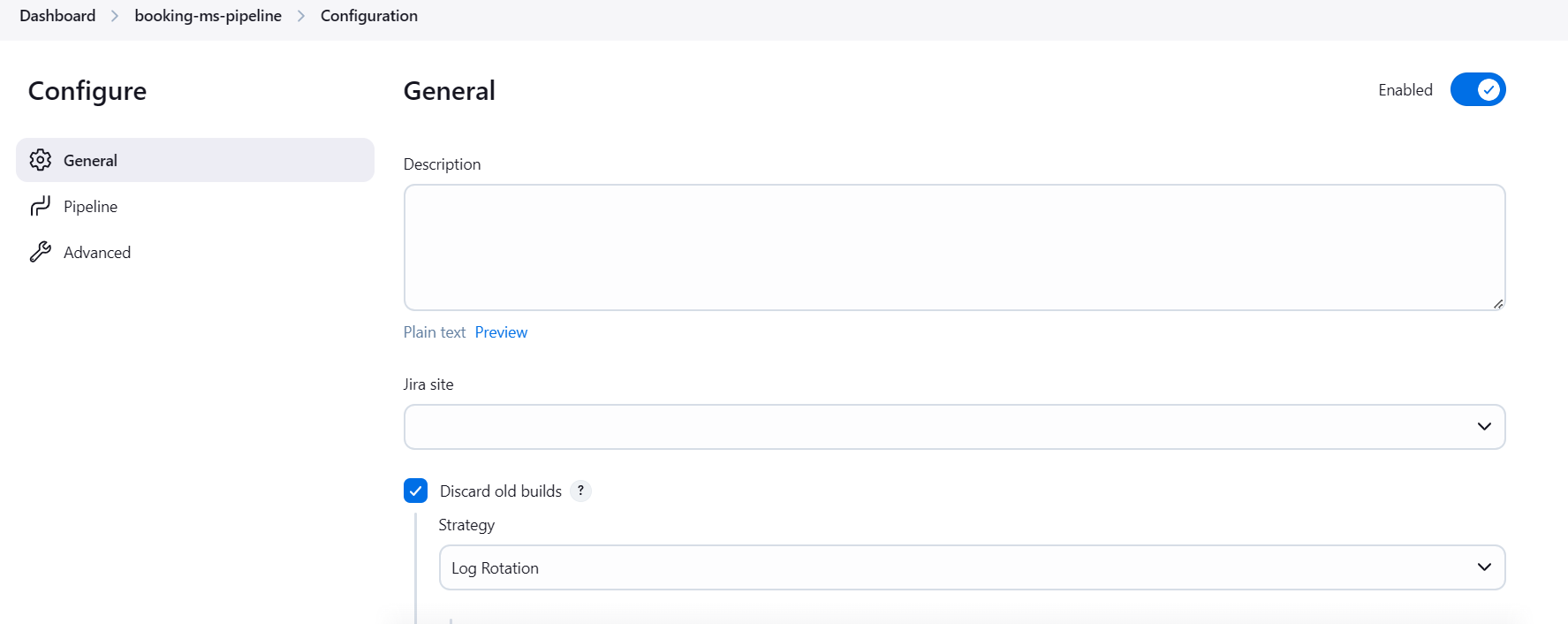


**Provide some description and click on create**



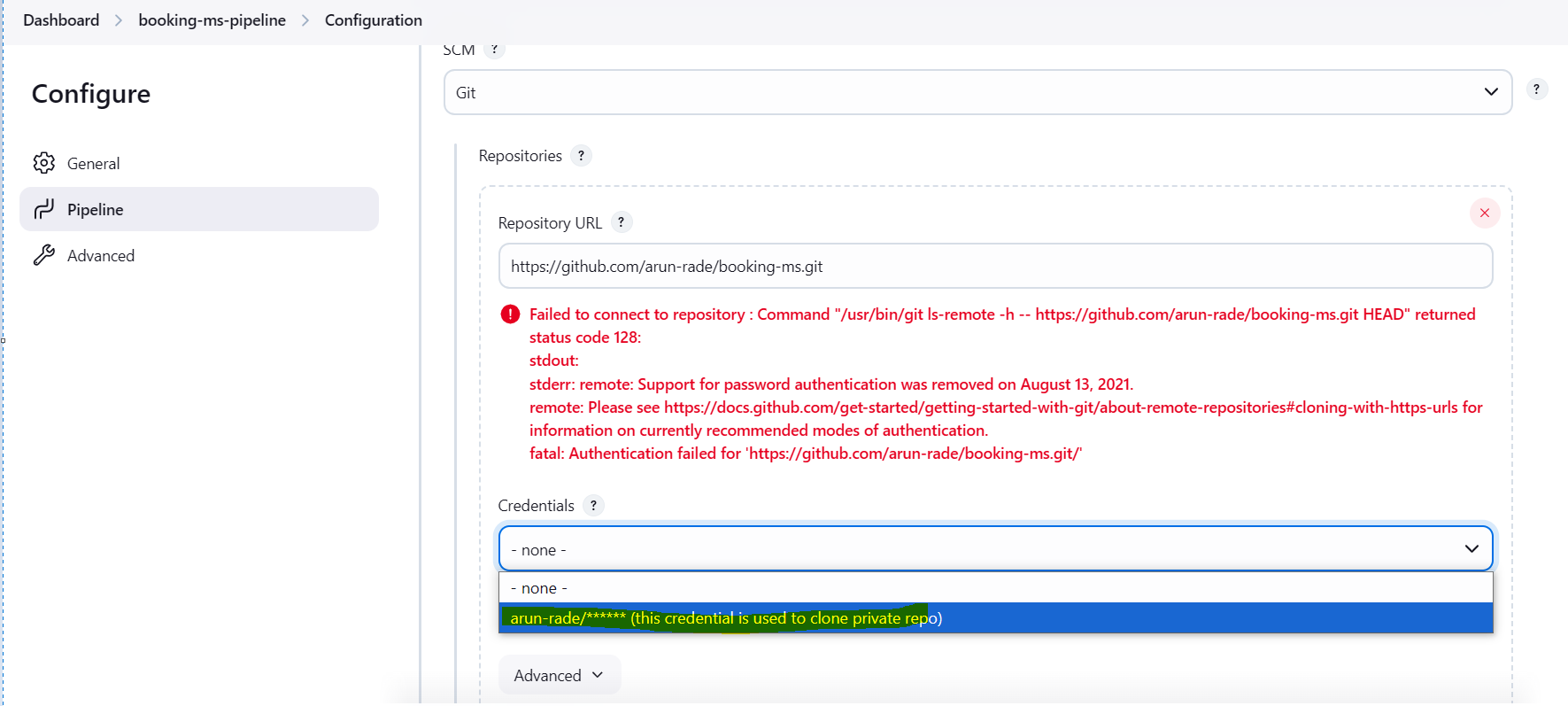
**Now we need to add this credential in Jenkins job i.e. booking-ms-pipeline**

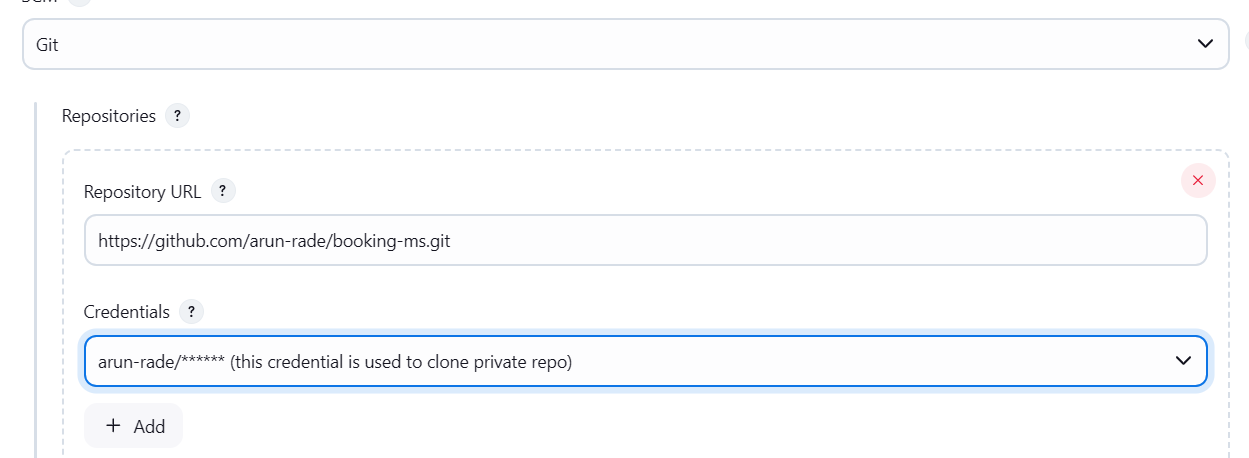
**Click on Dashboard 🡪click on project booking-ms-pipeline --? Click on configure**



**You can see error under SCM 🡪 where we mentioned repository path 🡪 click on drop down of credential**

**You can new credentials are created. Select that credentials.**





**Save the changes.**

**Trigger any commit in dev branch so this job will trigger.**

**Parametrized job:**

**Go Jenkins 🡪 click on Dashboard 🡪 click on New Item**

