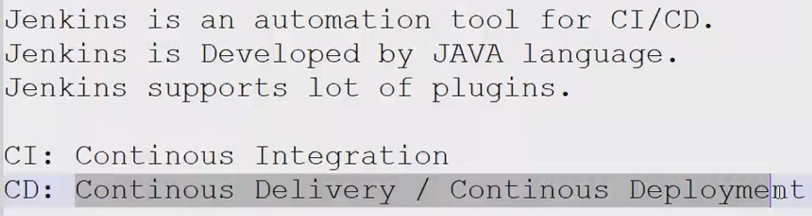
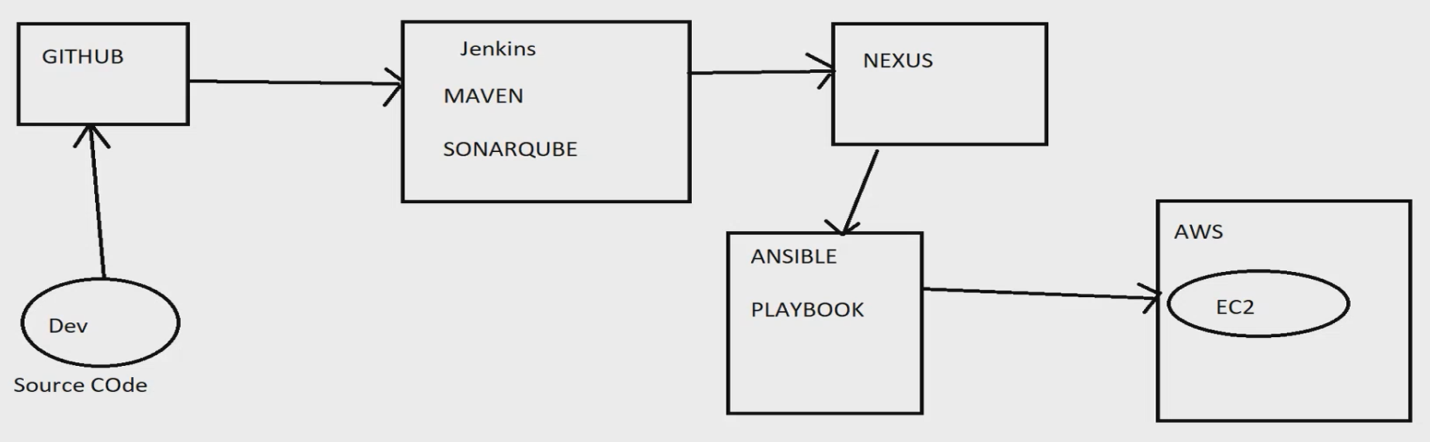
**JENKINS-01**

**DEFINATION:**

**EXPLANATION:**

**Cont. Integration:   
when any commits occurs on github with the help of web-hooks, based upon the event a Jenkins job get triggered which will execute different stages and build a war package and pushed to arti-factory location.   
Cont. Deployment:   
when package get created then it will automatically deploy on the end env.  
Cont. Delivery:   
It makes sure the env is ready for deployment but it will actually not deploy the package**

**Installations:**

Prerequisite to Install jenkins is java.

Download jenkins Repo:

=====================

sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat/jenkins.repo

Import the jenkins key:

======================

sudo rpm --import https://pkg.jenkins.io/redhat/jenkins.io-2023.key

Update ec2:

==========

sudo yum upgrade

Add required dependencies for the jenkins package:

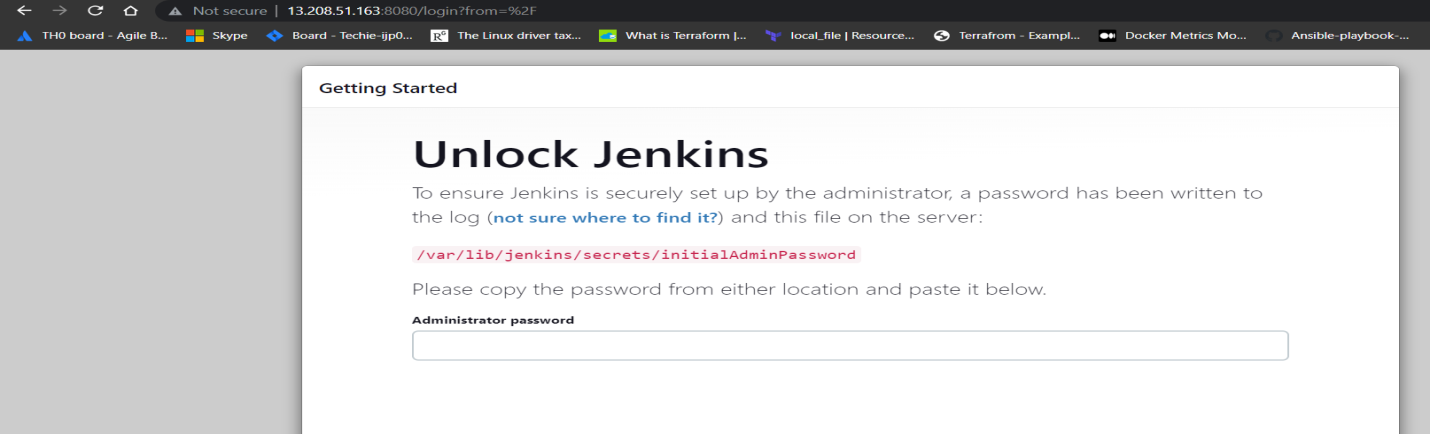
=================================================

sudo amazon-linux-extras install java-openjdk11

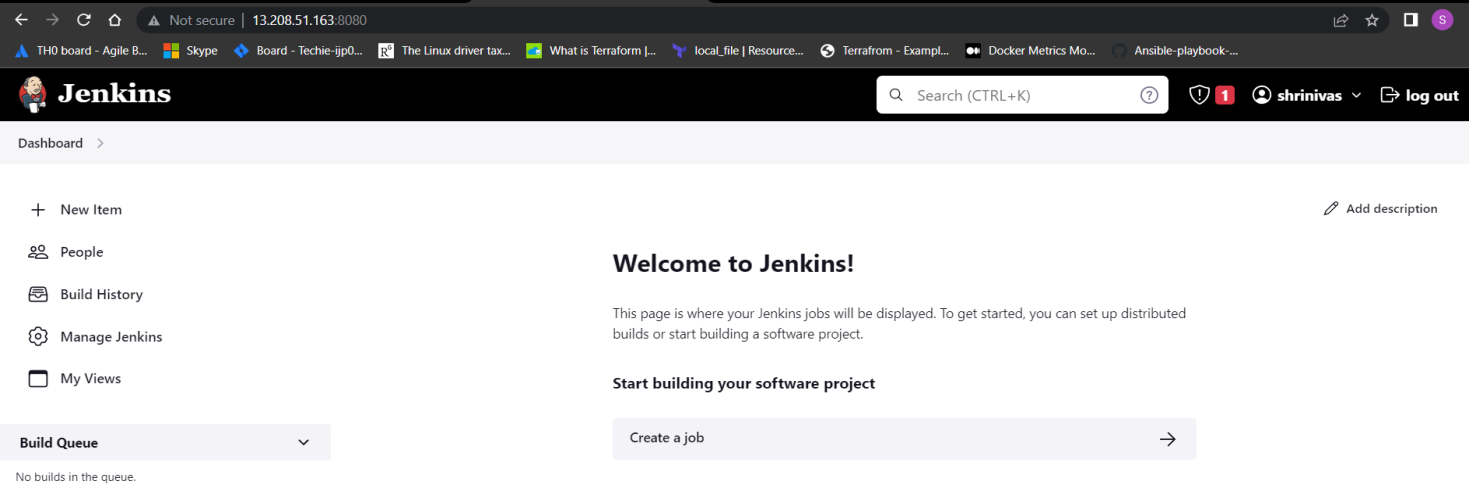
Install Jenkins:

===============

sudo yum install Jenkins



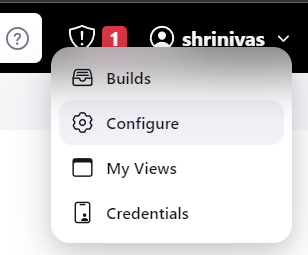
We have to copy the path and get the authenticate passwords,  


**Installation is complete:**

**JENKINS-02**

**To change the password:**

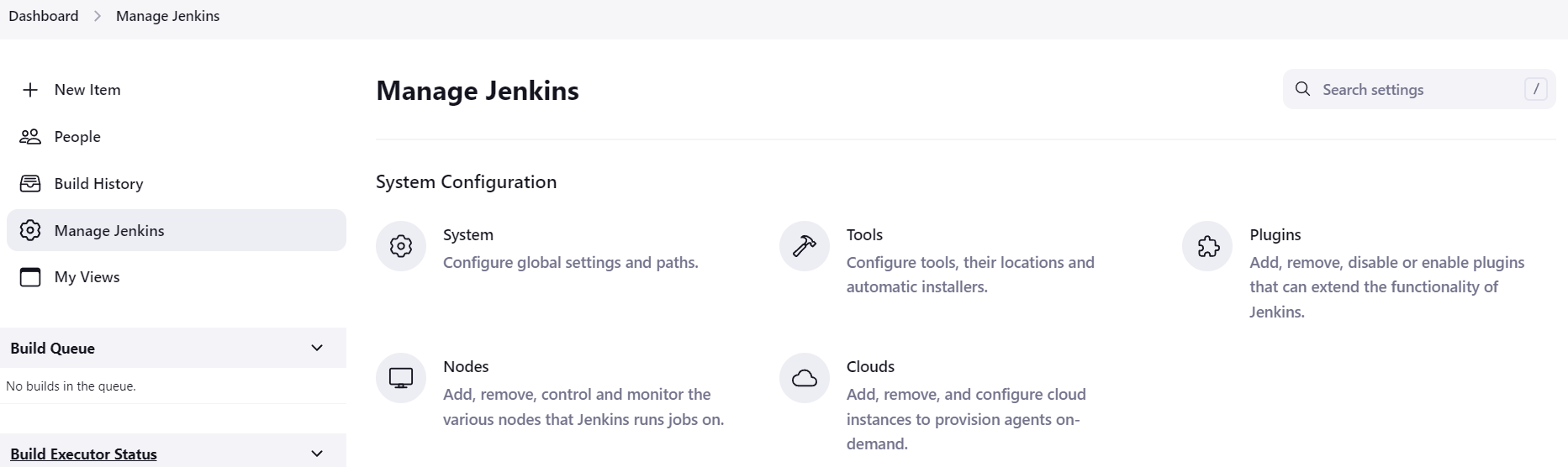
We can change the password by configure section:

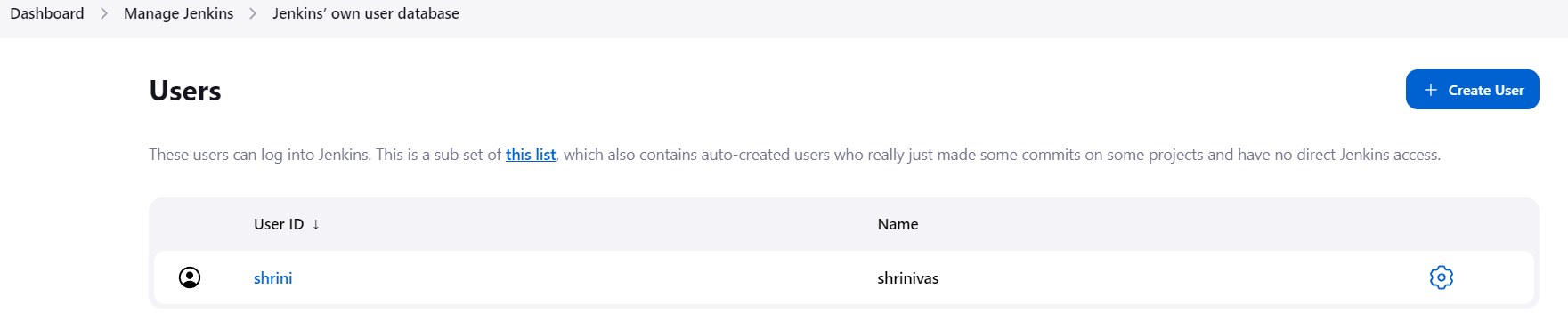


**Provide access of Jenkins (same accn) to our team-mates:**

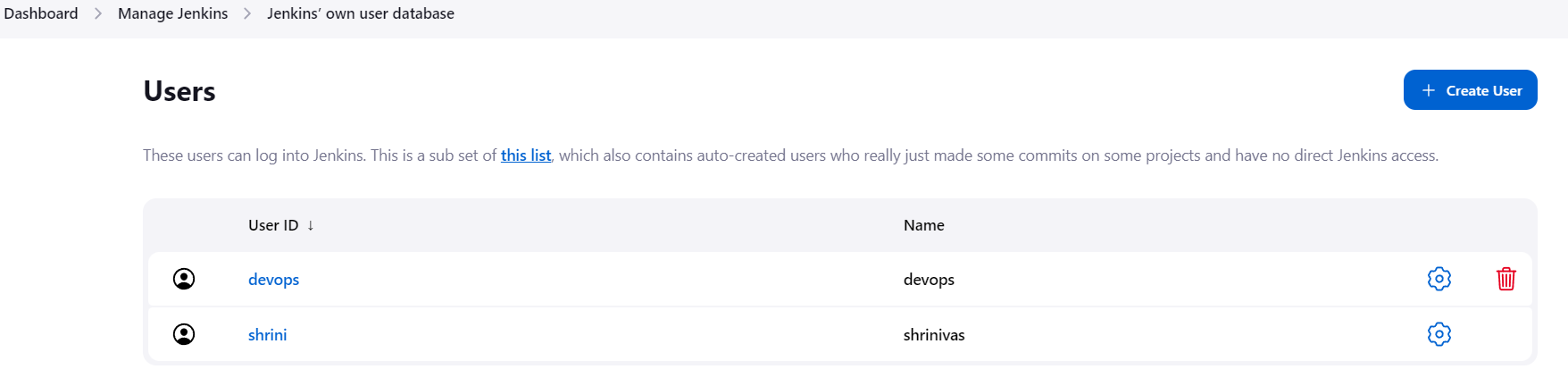
By plugins (role-based) we can provide access to any one

**PLUGINS:**It is pre-defined task which is easing our task with the help of plugin we can automate our task very easily

****

****

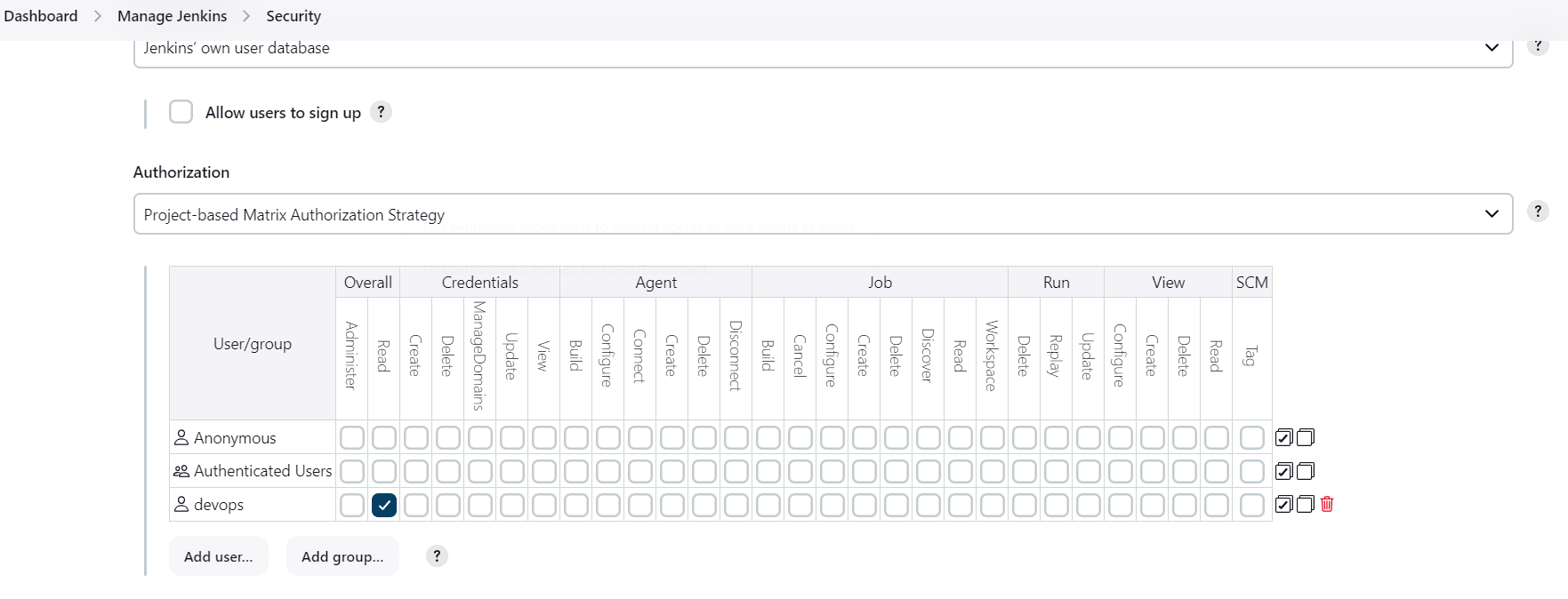
**By creating user we can create another user:**

****

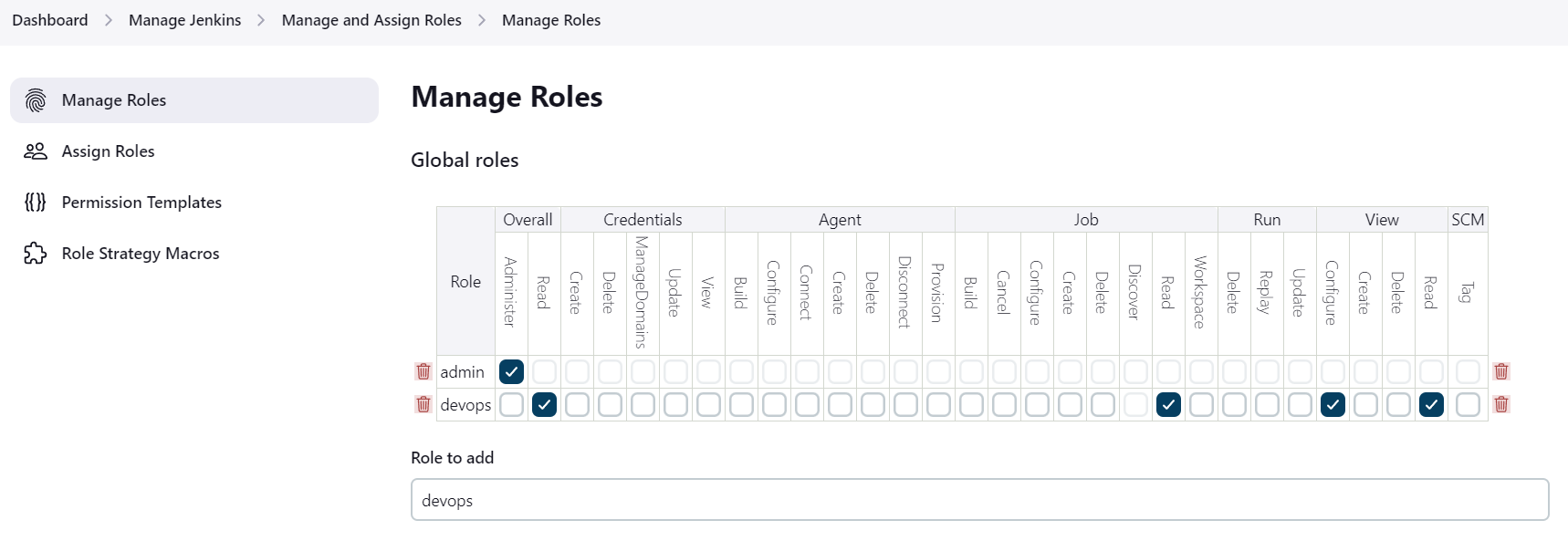
**And by giving credentials of devops user I get logged in:**

****

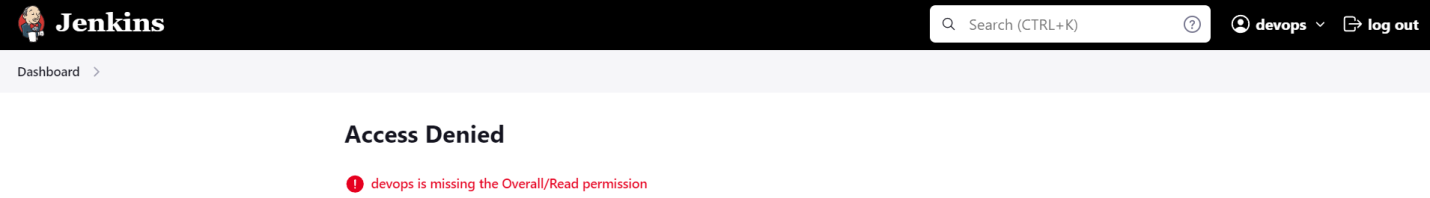
**Now it has all the access of Jenkins service but if we want to provide limited access then:**

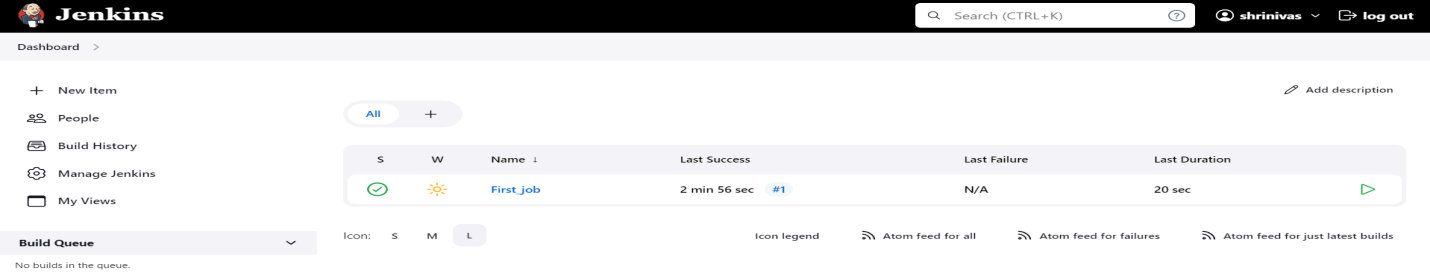
****

**And if we want to give some restrictions to devops user then we can assign some roles:**

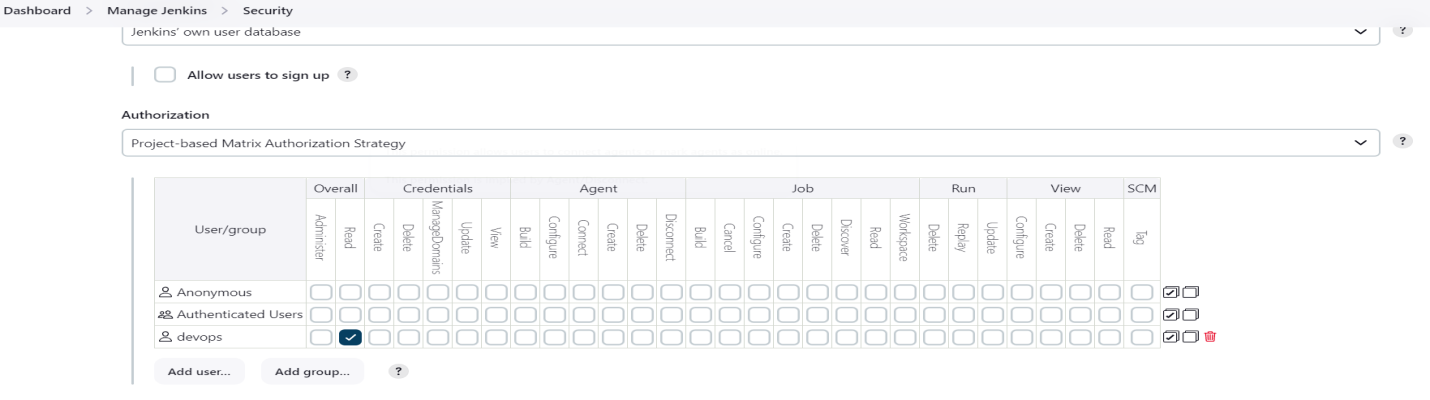
****

**Then if I login it will show access denied due to roles:**

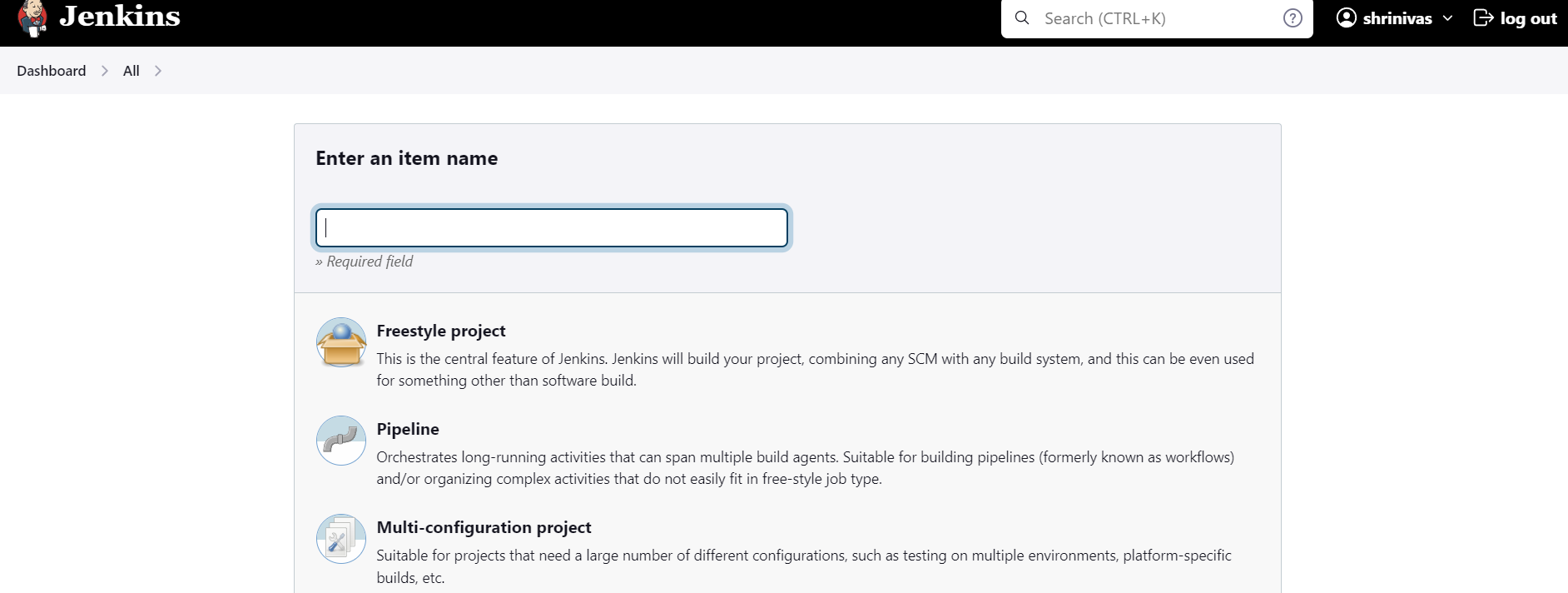
****

****

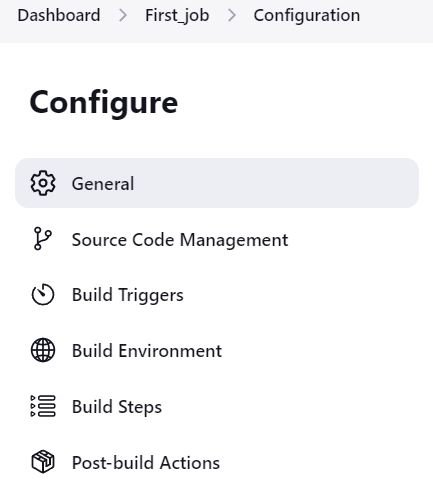
**After giving the roles it will provide the access according to the mention roles to the user**

****

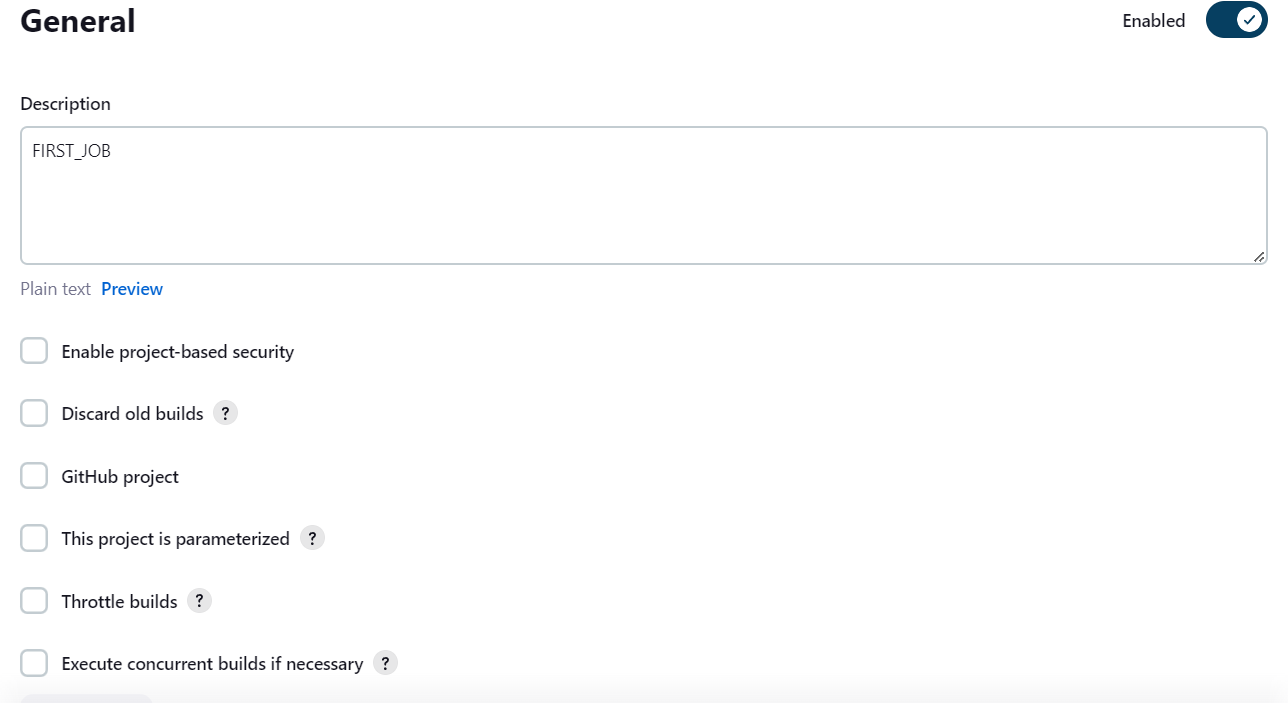
**JENKINS-JOBS:**

**We have different types of jobs it can be installed by plugins:**

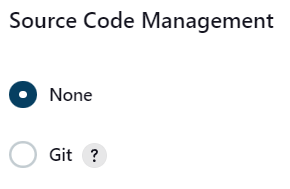
**On free-style we have six stages:**

****

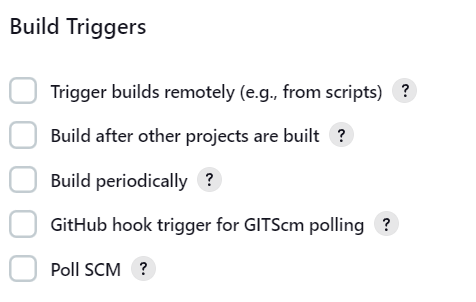
**General:**

****

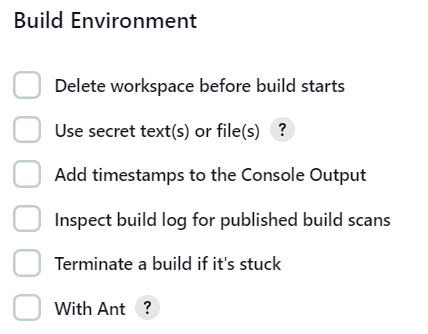
**Source code management:**

****

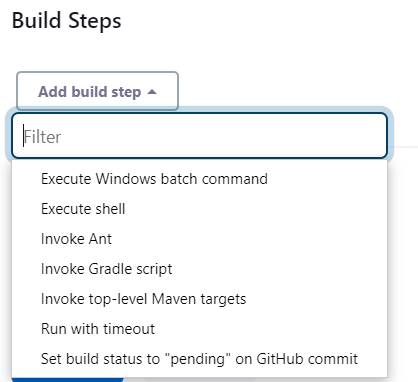
**Build triggers:**

****

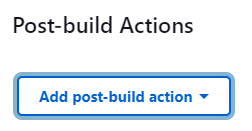
**Build Env:**

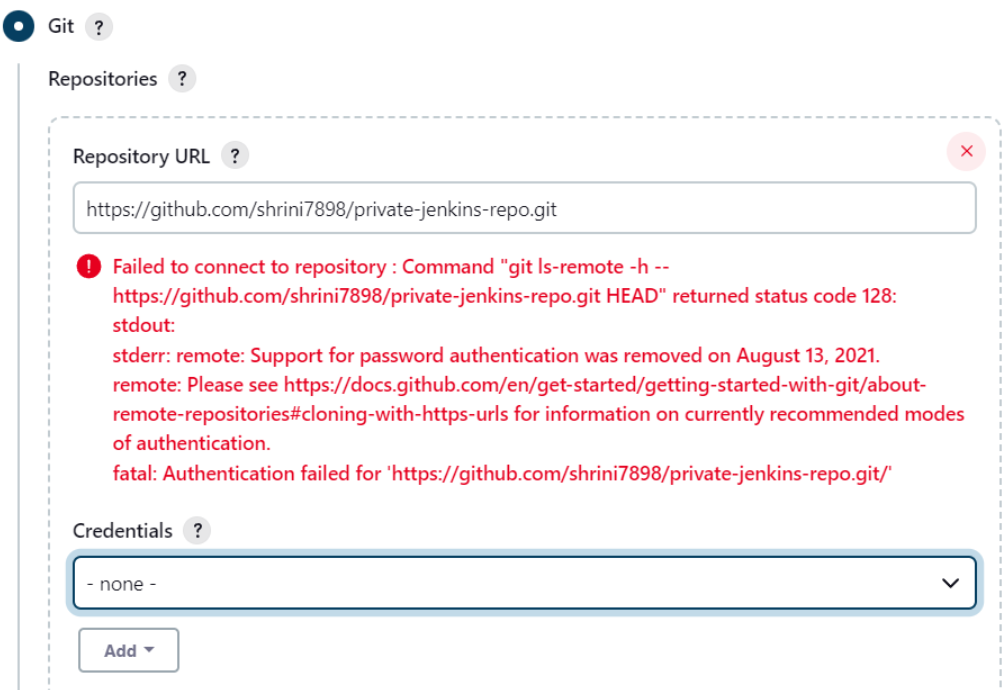
****

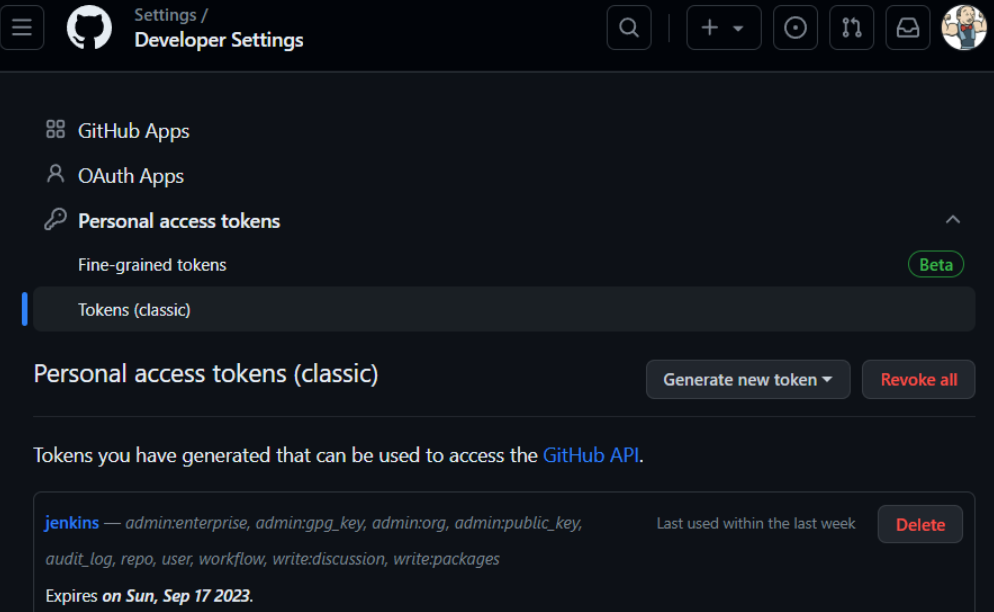
**Build steps:**

****

**Post-build Actions:**

****

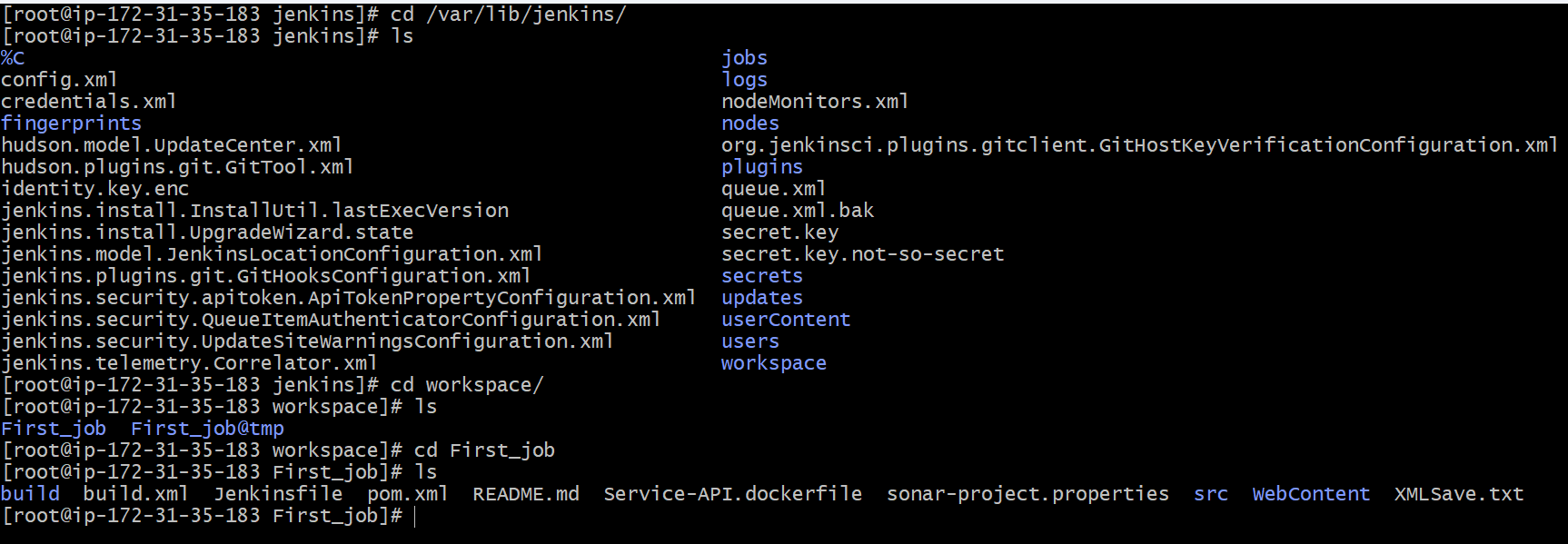
**Configuring private repo in Jenkins:  
If we give the private repo url, it will through error like below,**

**Then we will create personal access token from github accn: After that we can get the access:**

****

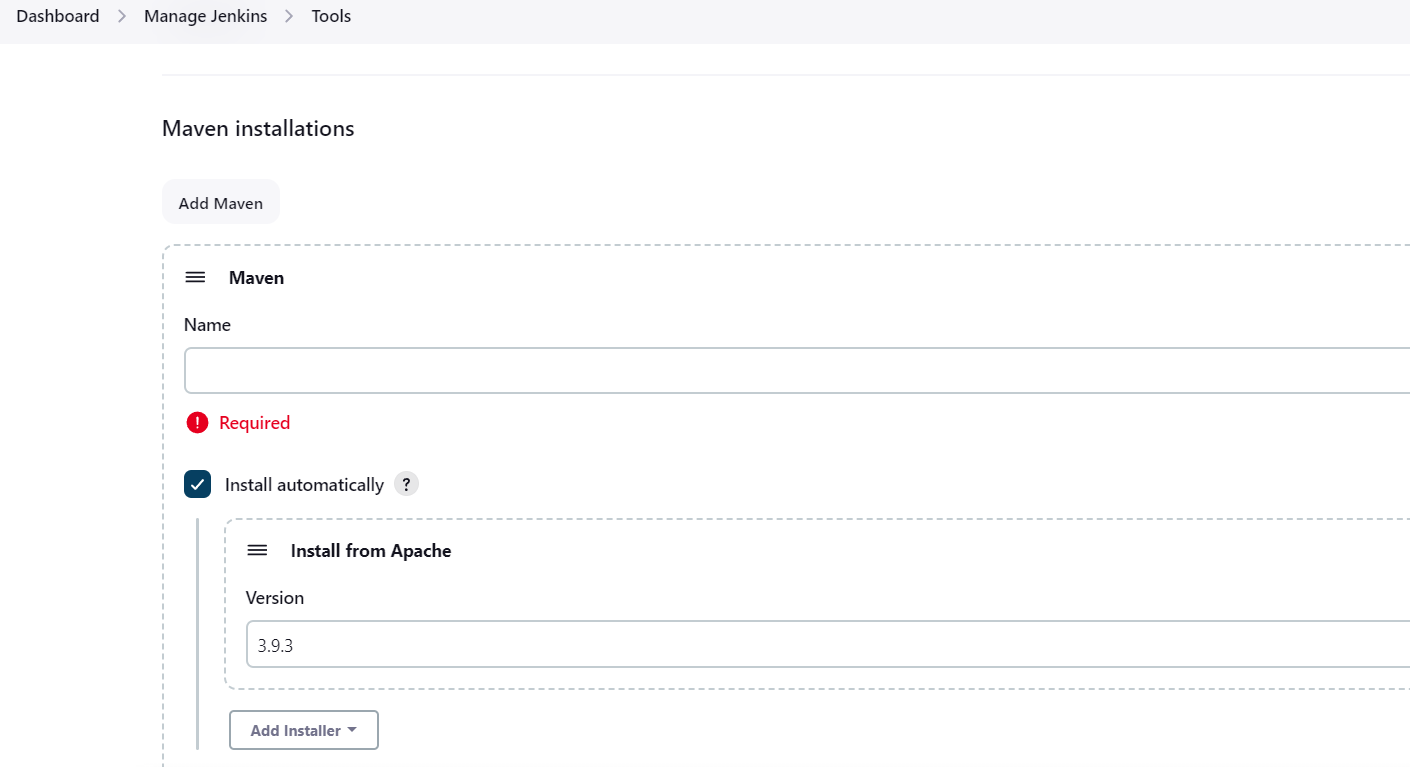
**NOTE: Make sure that git is installed on Jenkins server.**

**After creating the jobs from github URL we can find our workspaces on Jenkins server;**

****

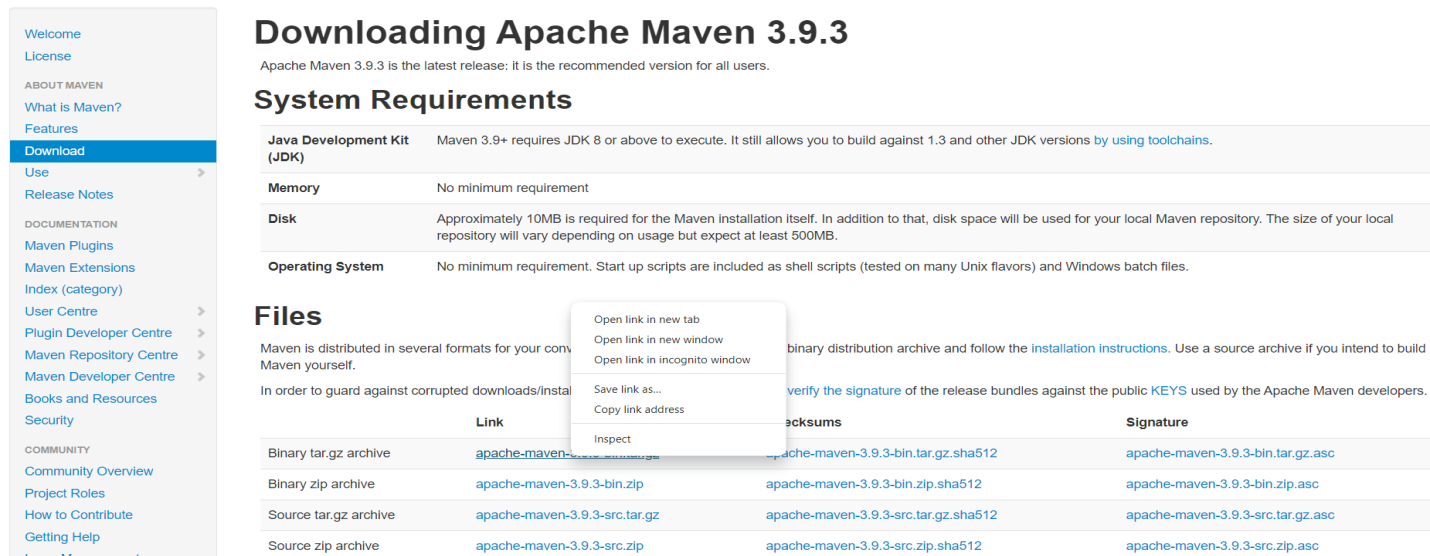
**Now, we will make packages of our job for that we required maven as a build tool we can installed by plugins;**

**By default it gets already installed we can find it out in “tools configuration side”;**

****

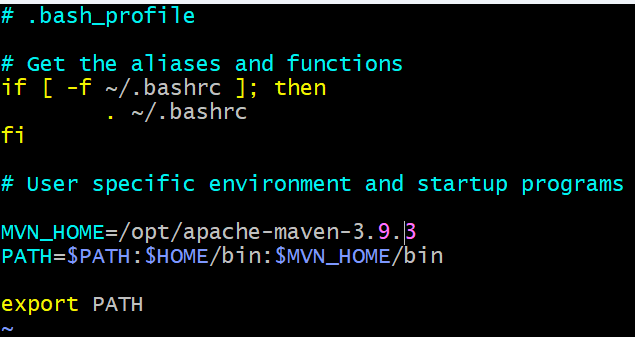
**So we can install maven by two options through plugins and another is from Jenkins server**

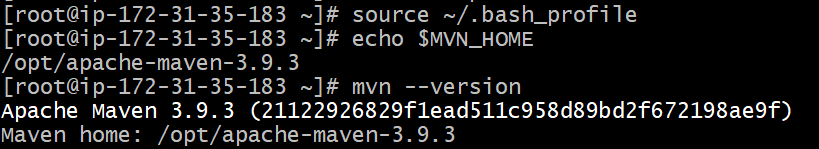
**And also for the best practices we should try to installed in Jenkins server,**

****

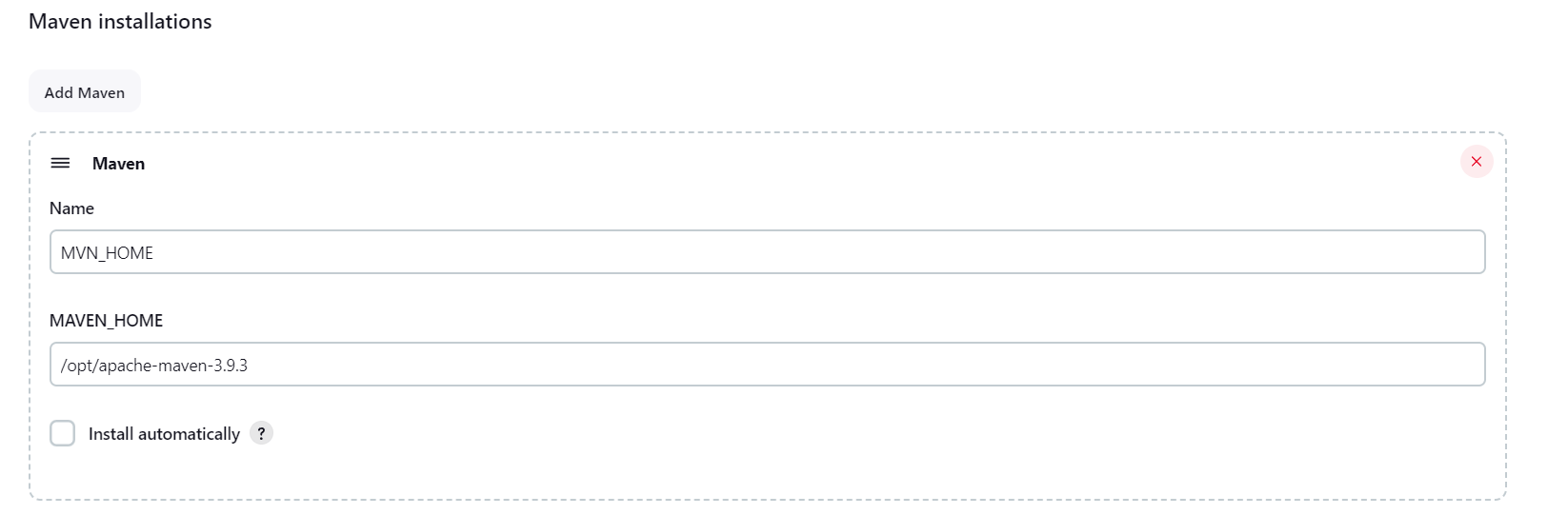
**We will copy the link address and then we can install through wget in “opt” location of our Jenkins server; **

**After this we are installing in opt area but we have to access in anywhere from the server for that we have to configure the global variables in “ .bash\_profiles”**

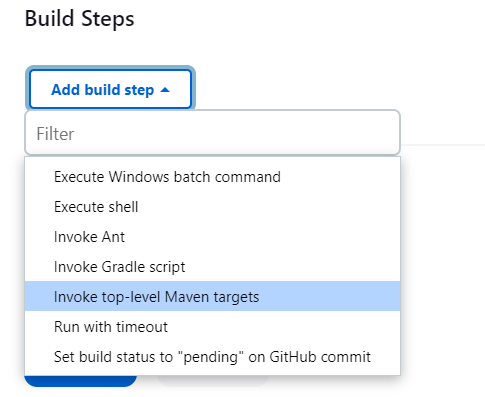
****

**After saving the file we have to restart the file so command will be: source** ~/.bash\_profile****

**Then in Jenkins we can mention the same path which it can easily find and install the maven:**

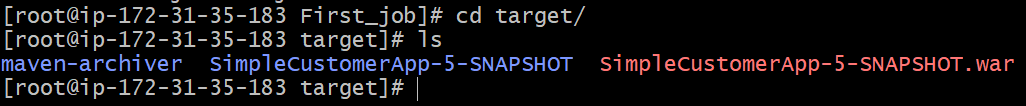
****

**On Jenkins job we take maven by below option:**

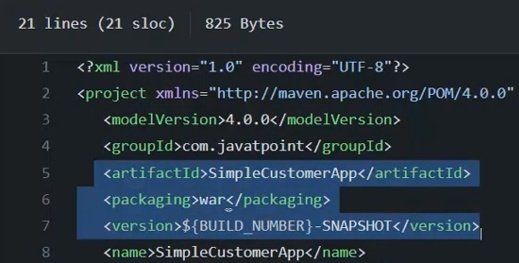


**By clicking build now it get successfully make the war packages by maven and if we check on our workspaces we can find one folder named “target”**

**It will only create when we use any build tools:**

****

**And where it takes the name of “SimpleCustomerApp-5” it is taking by the pom.xml file of git**

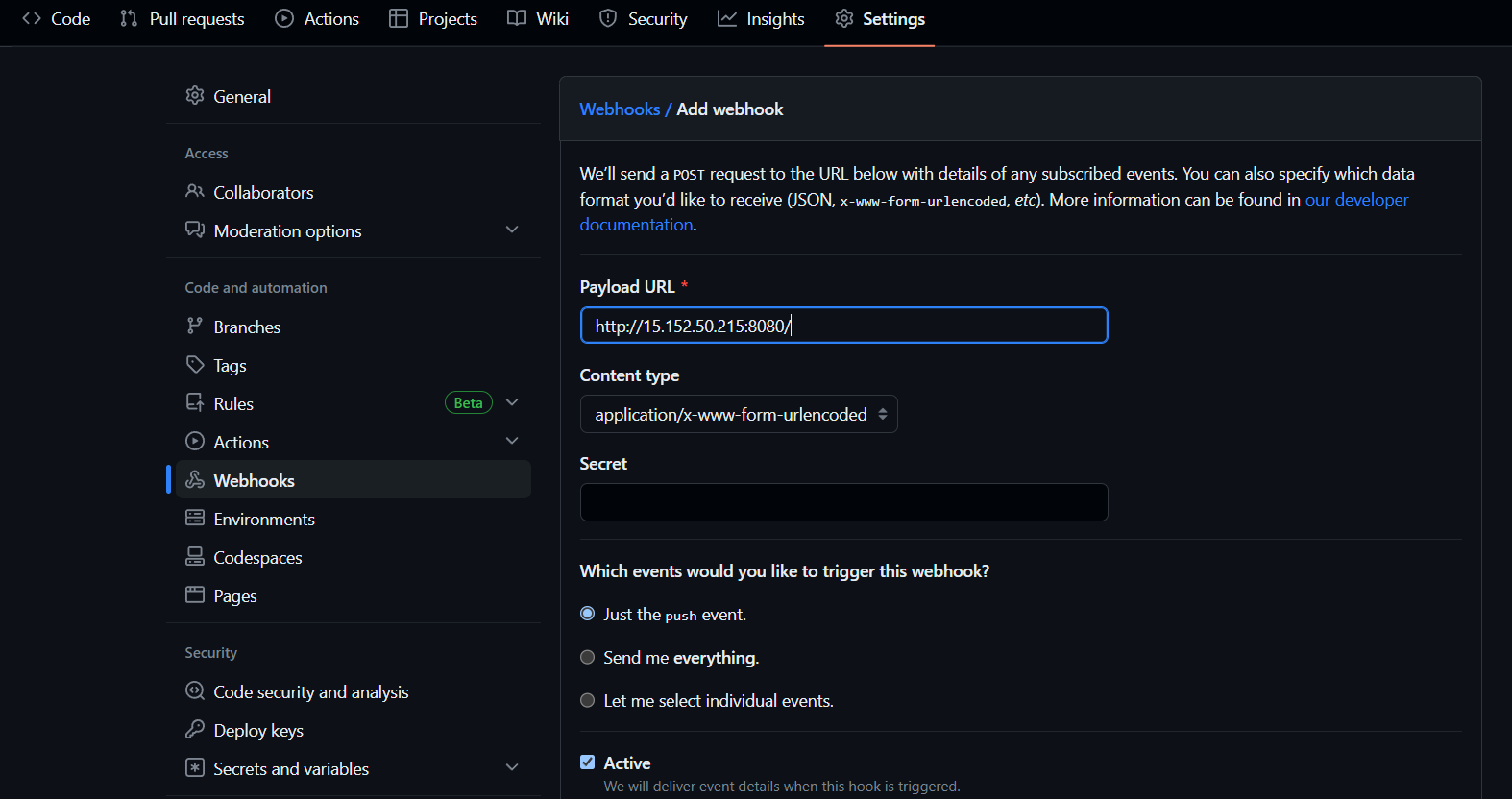
****

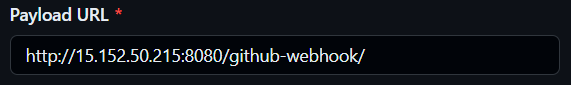
**And pom.xml file will be created by developers**

**Now build we are doing manually but we can configure by web hooks also**

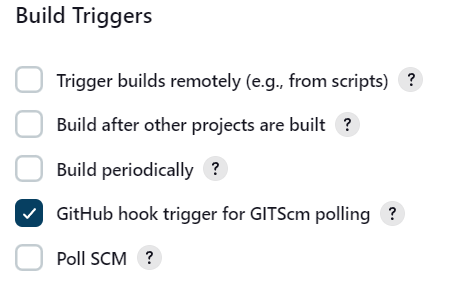
**WEB-HOOKS:**

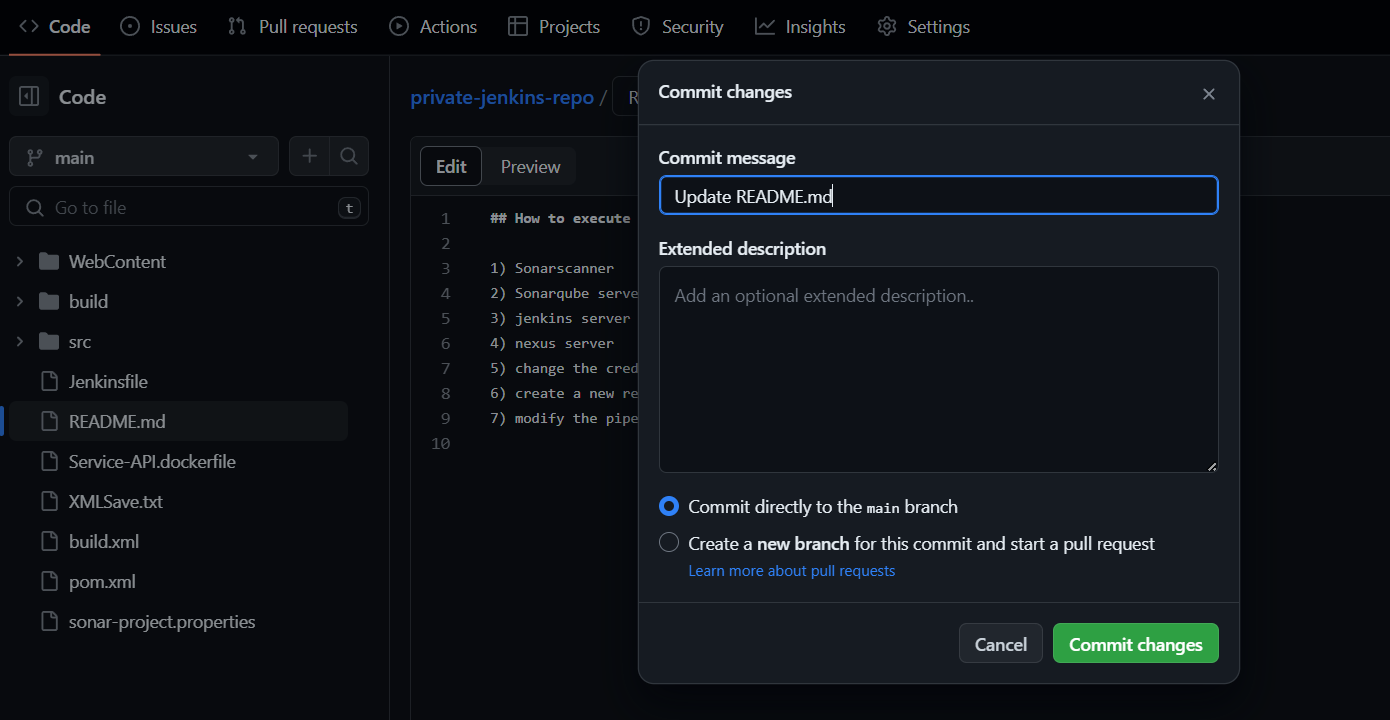
* **Web hooks are a feature that allows you to receive notifications or trigger actions in an external system  
  whenever certain events occur in a Git repository.**
* **Web hooks are commonly used for automating workflows, integrating with external services, or triggering CI/CD pipelines.**

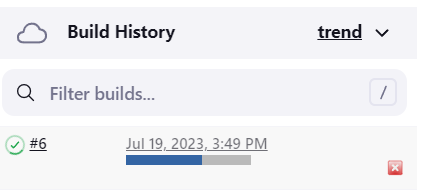
**We will configure web hooks to our repo by the URL of our Jenkins,**

**We have to mention the Payload URL like these;**

**Then in build triggers option of our project we have to enable “hook trigger”**

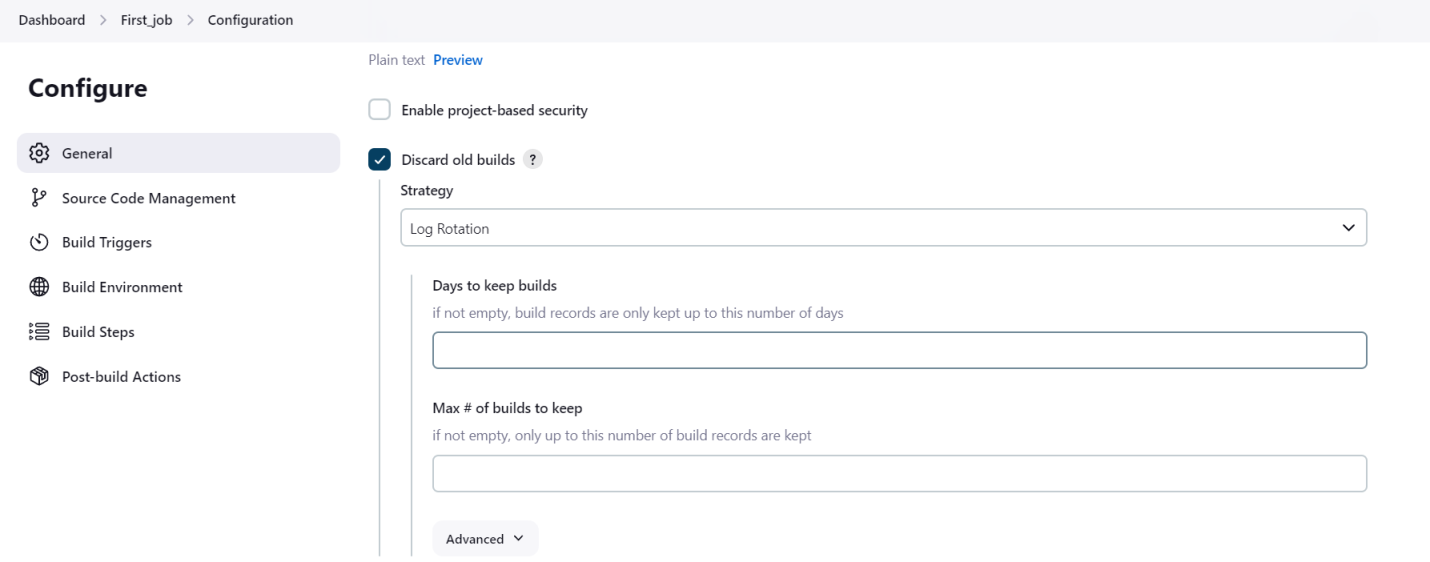
****

**And then if we make any changes then it will trigger and build the job:**

****

**In every changes it get triggered and build the job so and it will create so many jobs which are not required so we can do the “log rotations” in which it will delete the old jobs which are not required;**

**DISCARD OLD BUILDS:**

****

**And, if now check our build history it will only show the latest “3” because we mentioned the “log rotation to show only “3”**

**🡨-------------------SHRINIVAS-----------------🡪**