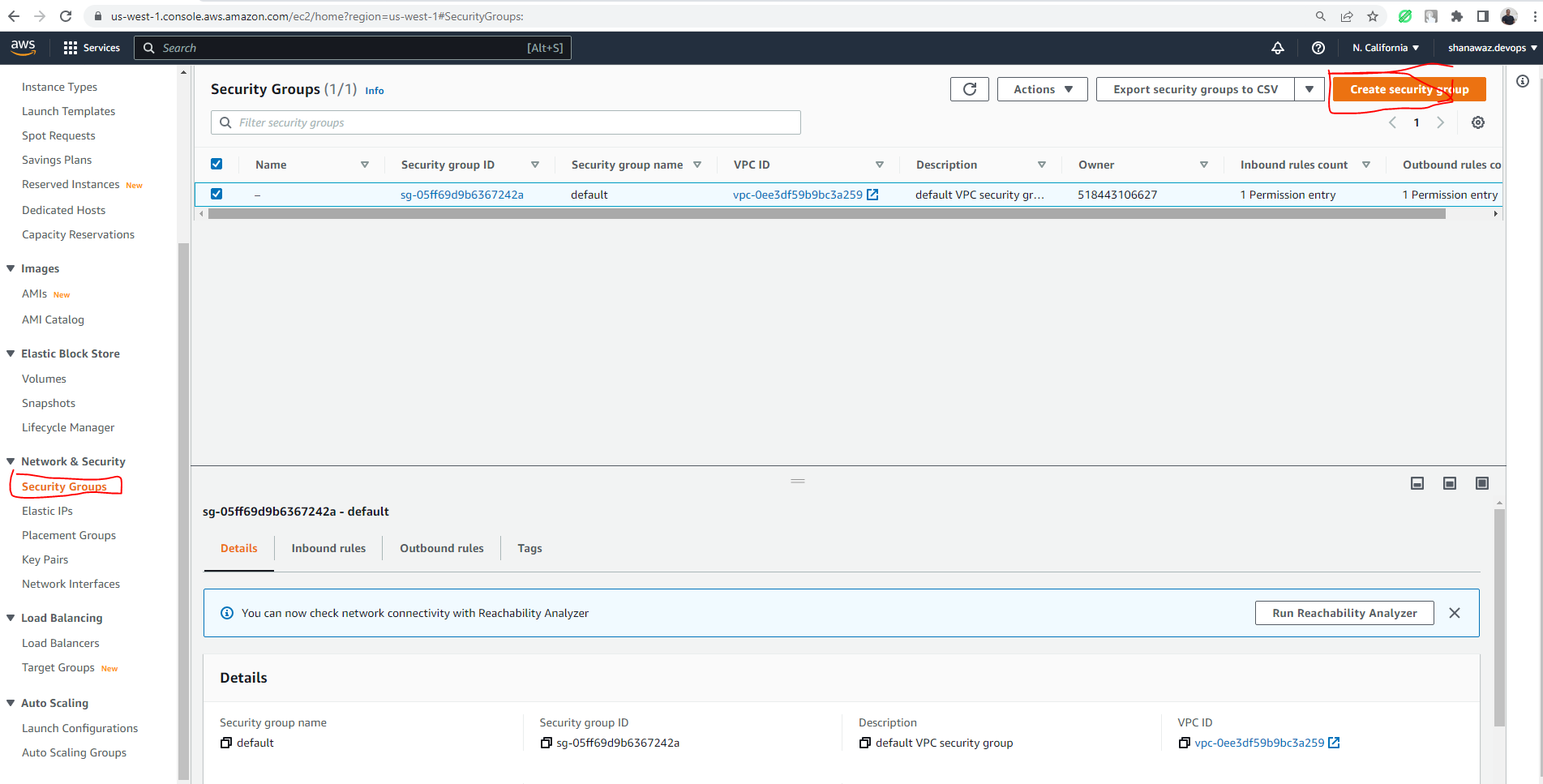
**CONTINOUS INTEGRATION**

Security Group Configuration for Jenkins, SonarQube, Nexus

1. Login to amazon aws, on the left side click on Security groups and click on create security group.



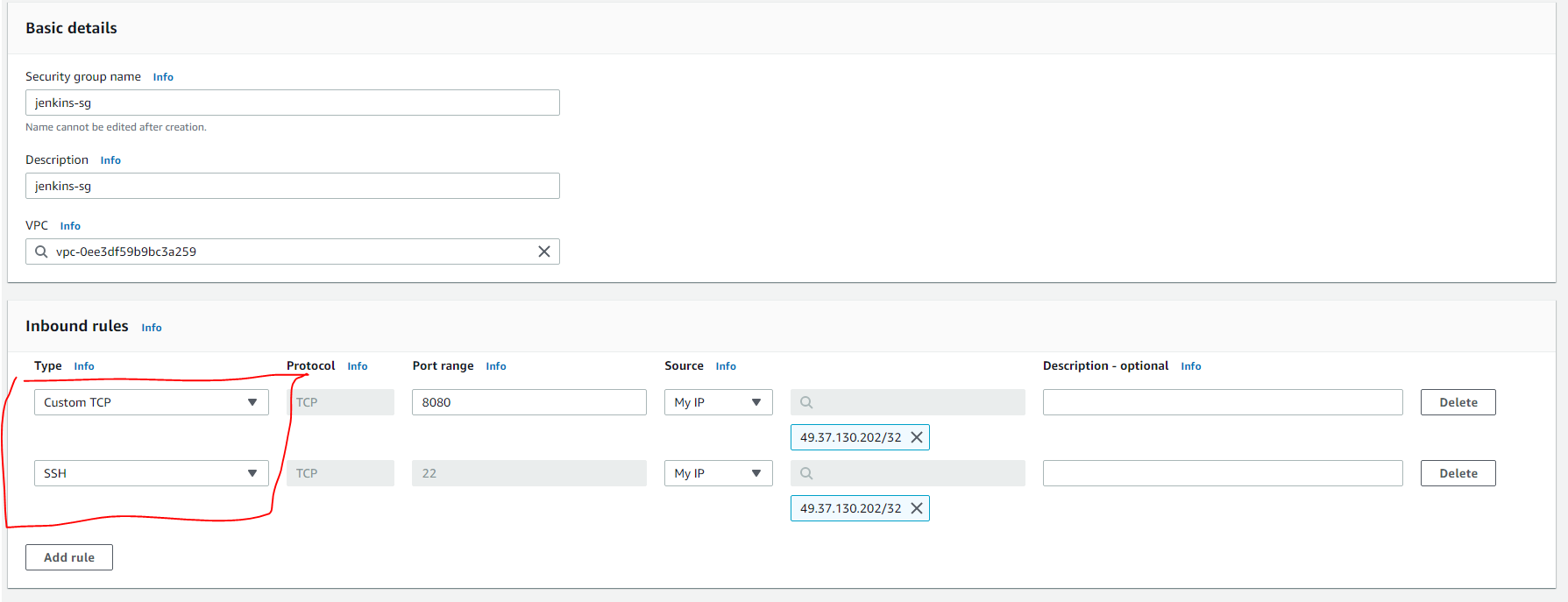
1. Jenkins run on port 8080

Security group Name: Jenkins-sg

Description: Jenkins-sg

**Inbound rules**

Click on Add rule: Configure below rules



1. Nexus run on port 8081

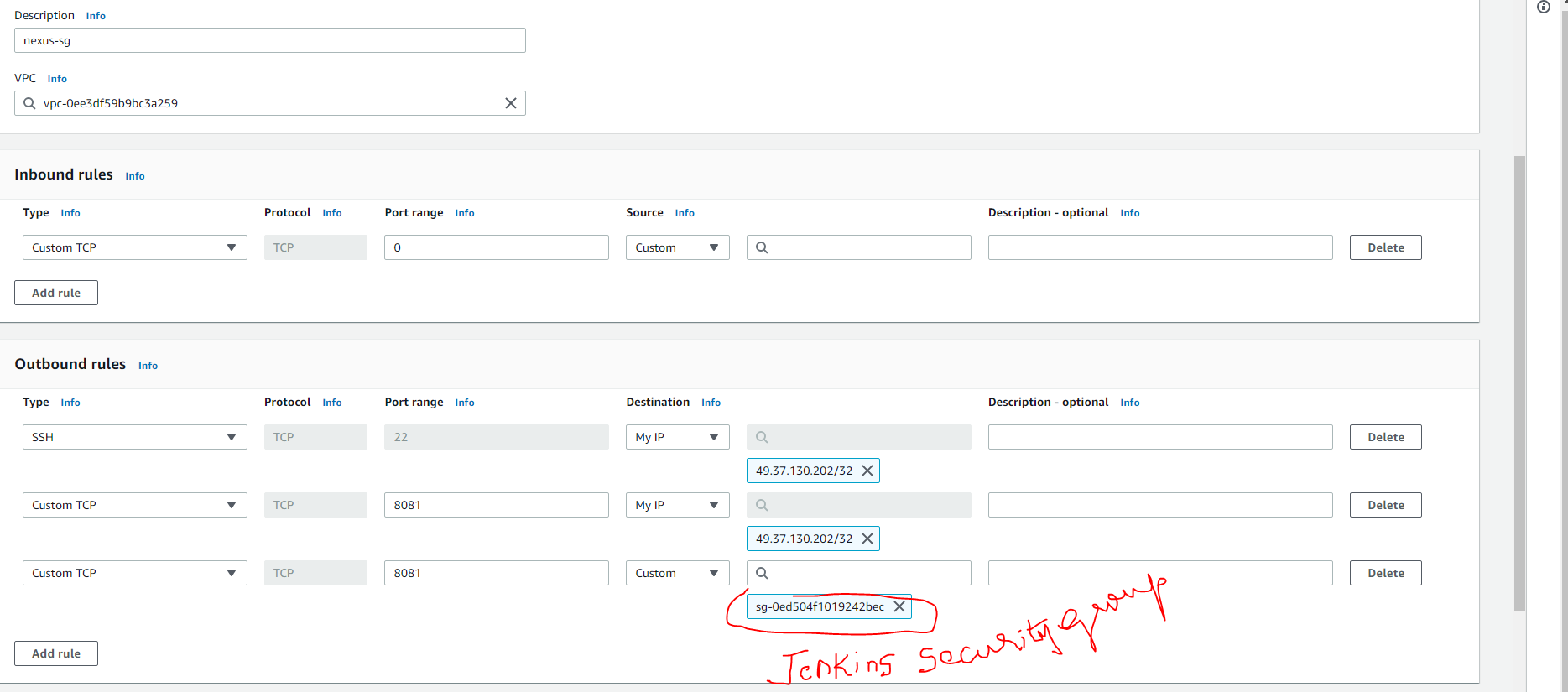
Security group Name: nexus-sg

Description: nexus-sg

**Inbound rules**

Click on Add rule: Configure below rules and click on create security group.

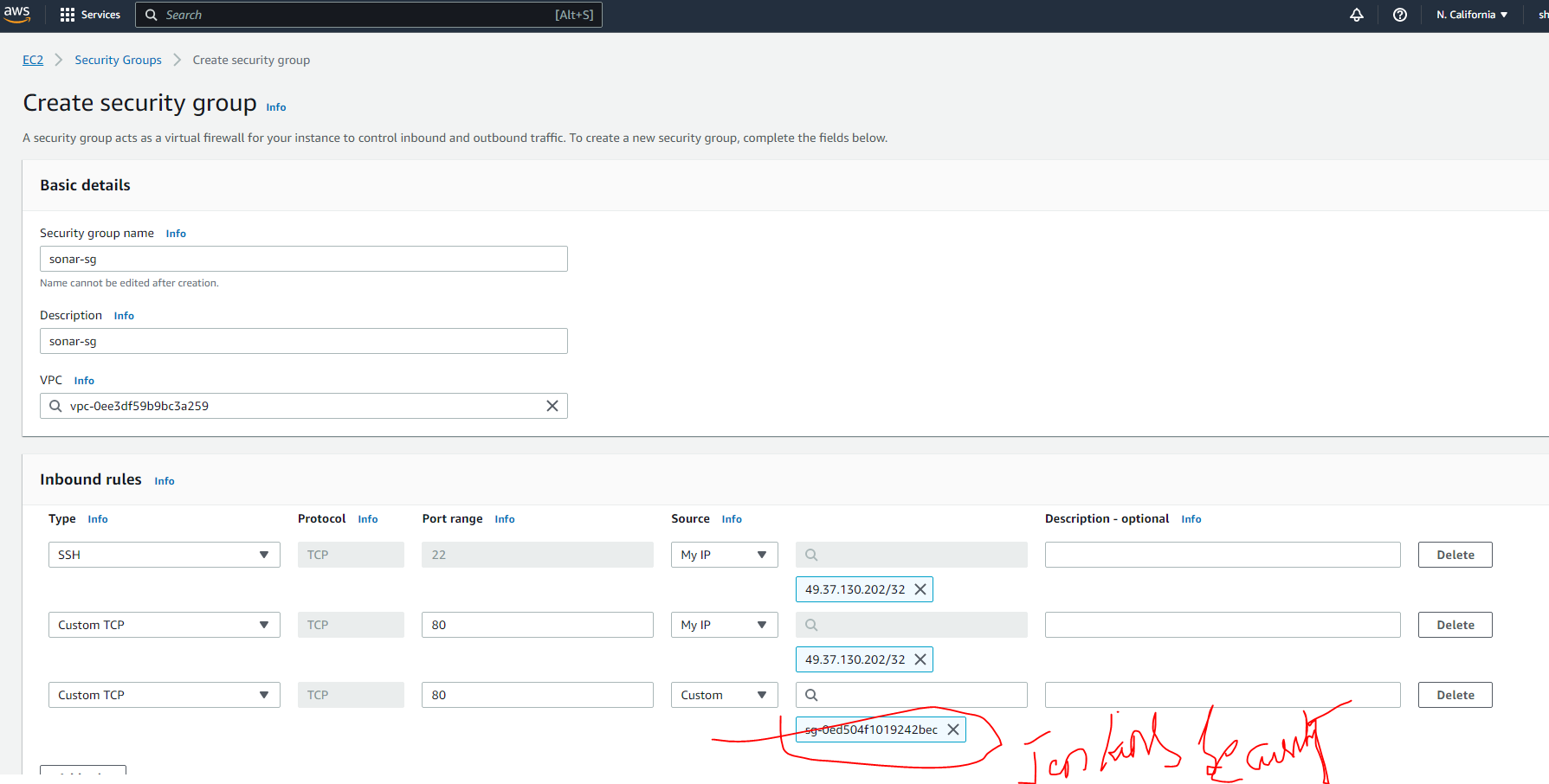
**It can download dependencies from nexus service and upload artifact that why we select jenkins-sg in the source**



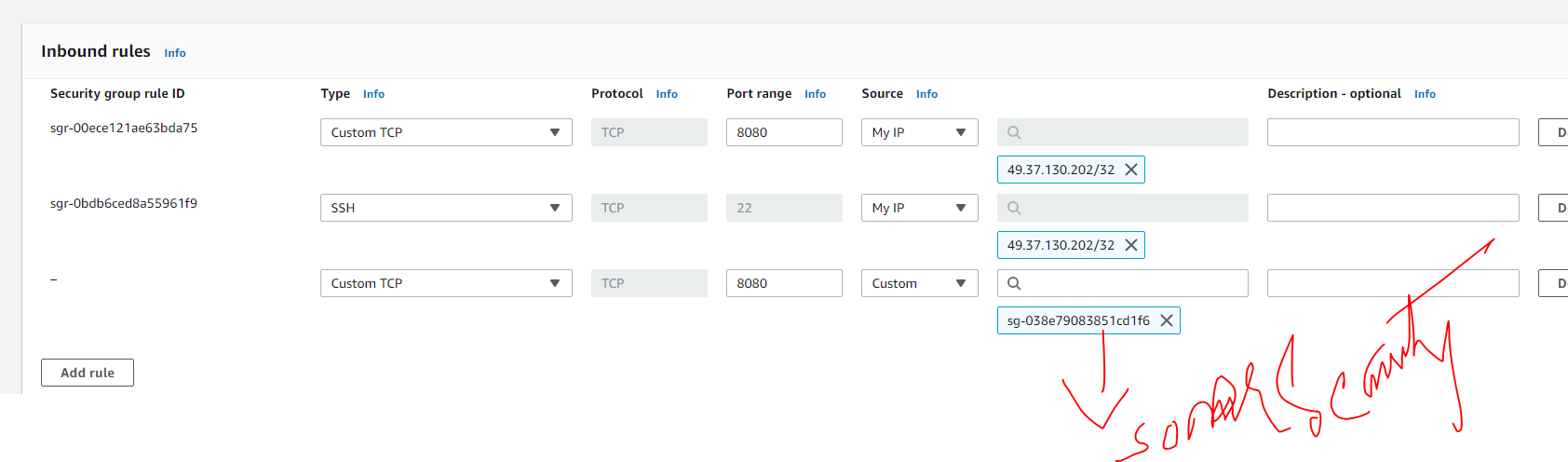
1. Sonarqube run on port 9000

Security group Name: sonar-sg

Description: sonar-sg



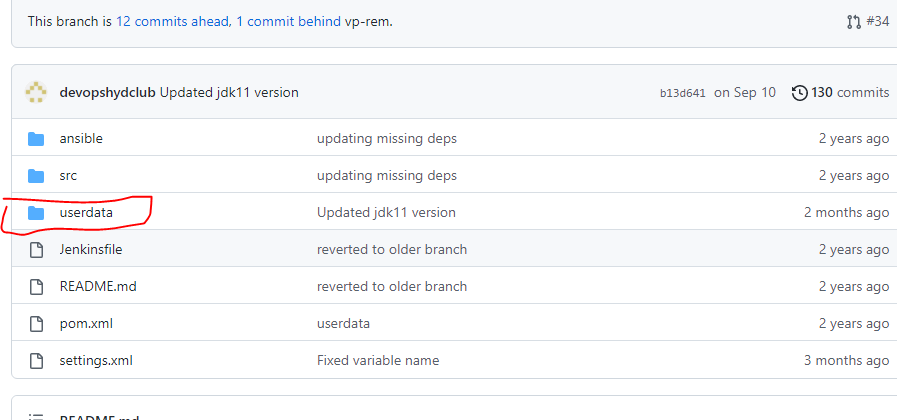
1. Now select back jenkins server and update sonar security group in inbound rule.



**EC2 Instance:**

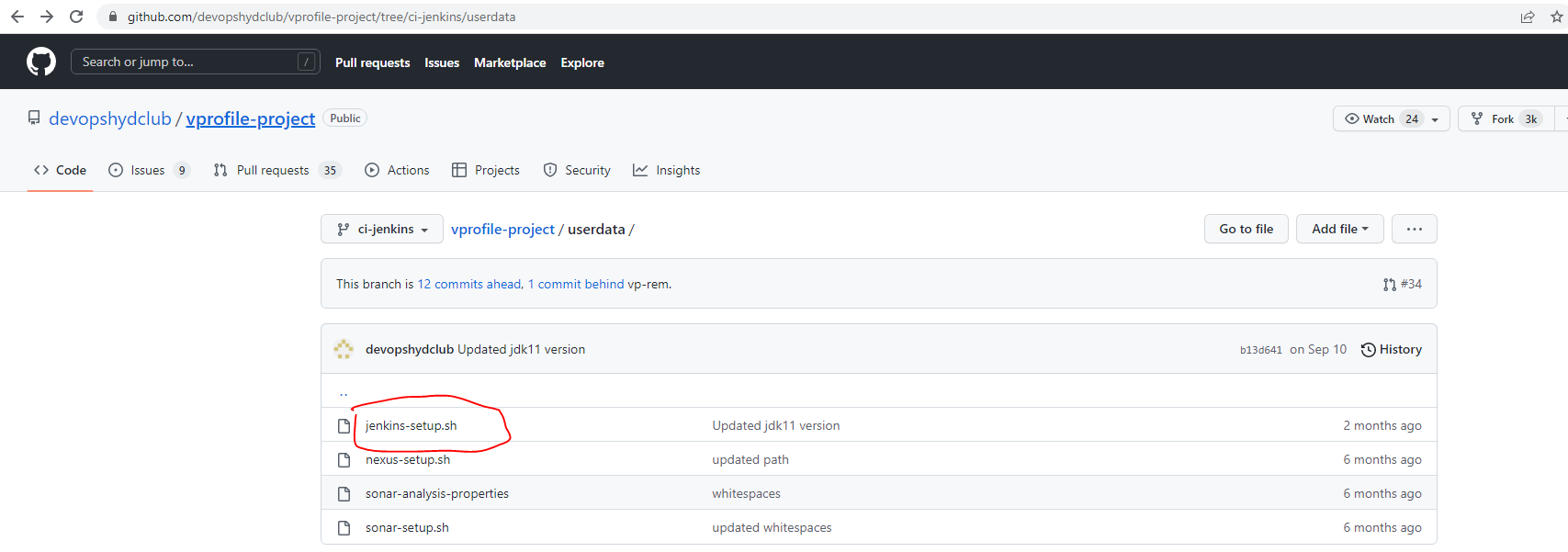
1. Create 3 new amazon machine for Jenkins, SonarQube, Nexus.
2. For Installation of all the servers, navigate to GitHub

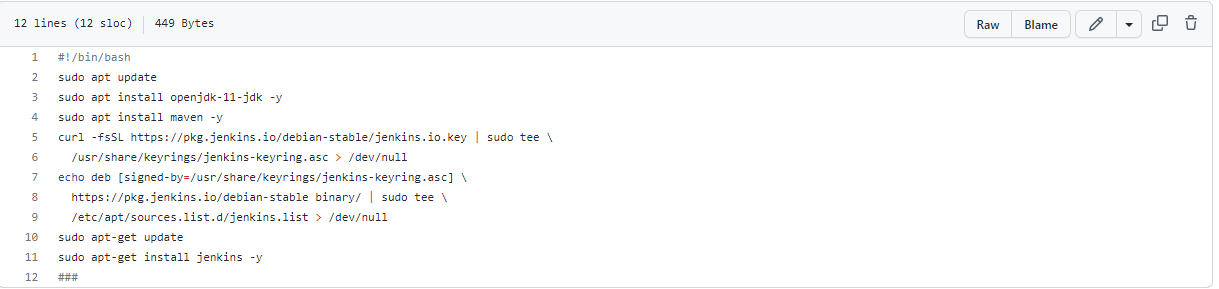
<https://github.com/devopshydclub/vprofile-project/tree/ci-jenkins>



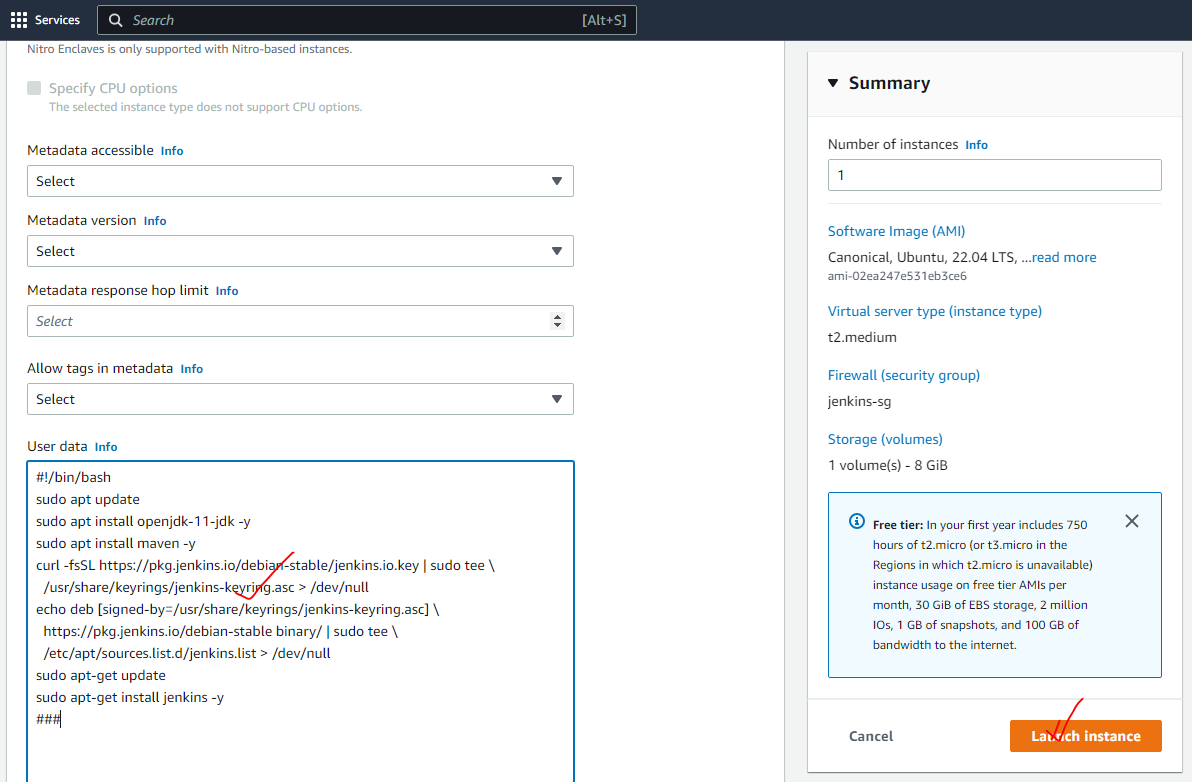
1. From the location, Go to userdata folder
2. In the userdata folder, you will find all the setup files for Jenkins, SonarQube, Maven.
3. Jenkins Configuration:
4. Create new amazon ubuntu machine and select t2. medium
5. Select group: jenkins-sg (which we created in security group connection)
6. In the advance details section, scroll down to last till you find userdata.

Paste the jenkins.sh code in the section.





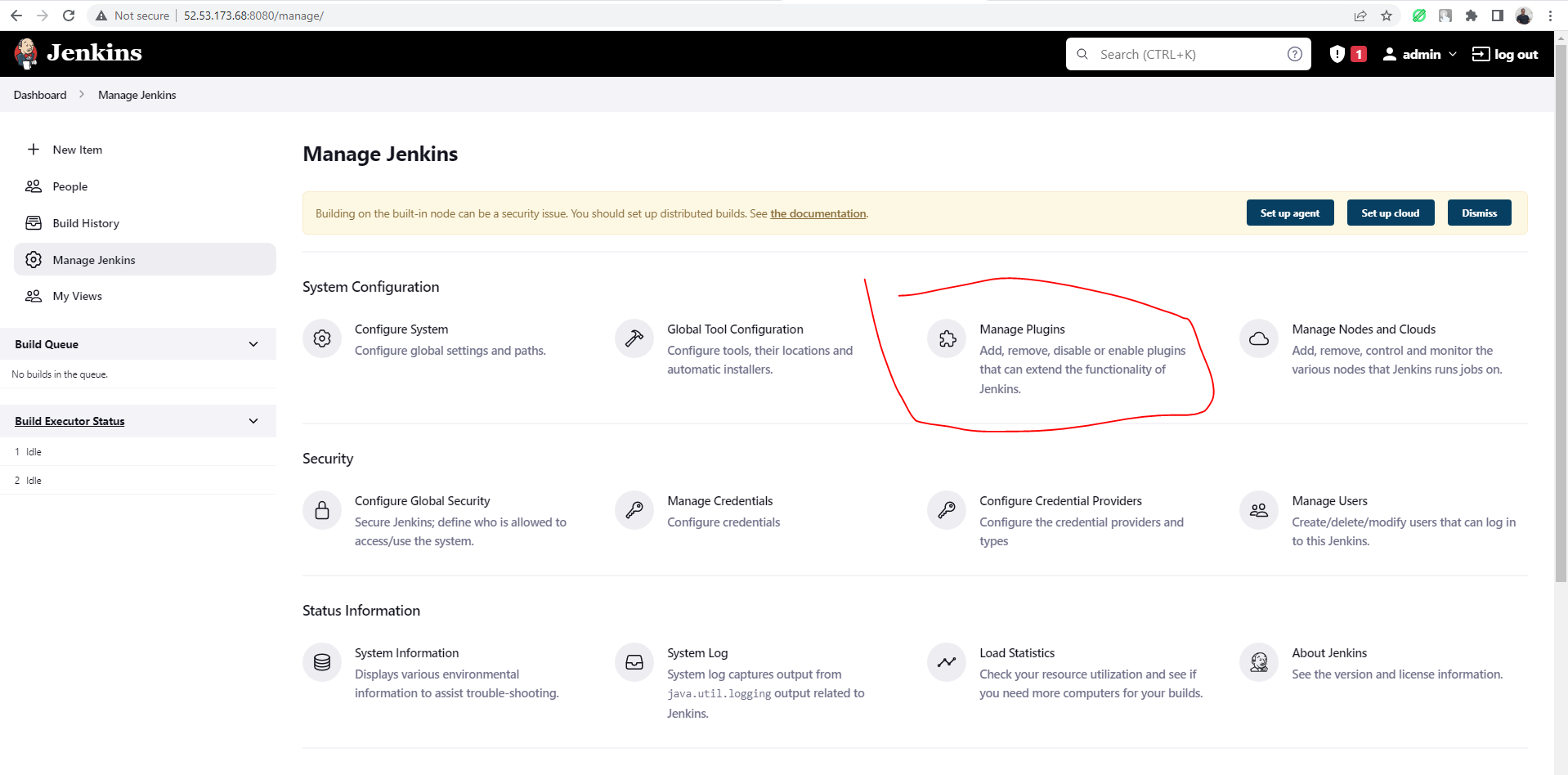
1. Paste the above code in the amazon ubuntu machine and launch instances



1. Nexus Configuration:
2. Create new amazon Linux 2 machine and select t2. Medium
3. Select group: nexus-sg (which we created in security group connection)
4. In the advance details section, scroll down to last till you find userdata.
5. Paste the nexus-setup.sh code in the section
6. SonarQube Configuration:
7. Create new amazon ubuntu machine and select t2. Medium
8. Select group: sonar-sg (which we created in security group connection)
9. In the advance details section, scroll down to last till you find userdata.
10. Paste the sonar-setup.sh code in the section

**Post Installation:**

1. Jenkins Login: After successfully launching the instance, Launch Jenkins virtual machine
2. Start Jenkins by running the following command: **systemctl start jenkins**
3. Launch JenkinsURL: <http://52.53.173.68:8080/>
4. Install plugins:



**Manage Jenkins**

|

Search Plugin: Maven integration

|

GitHub integration

|

Nexus artifact uploader

|

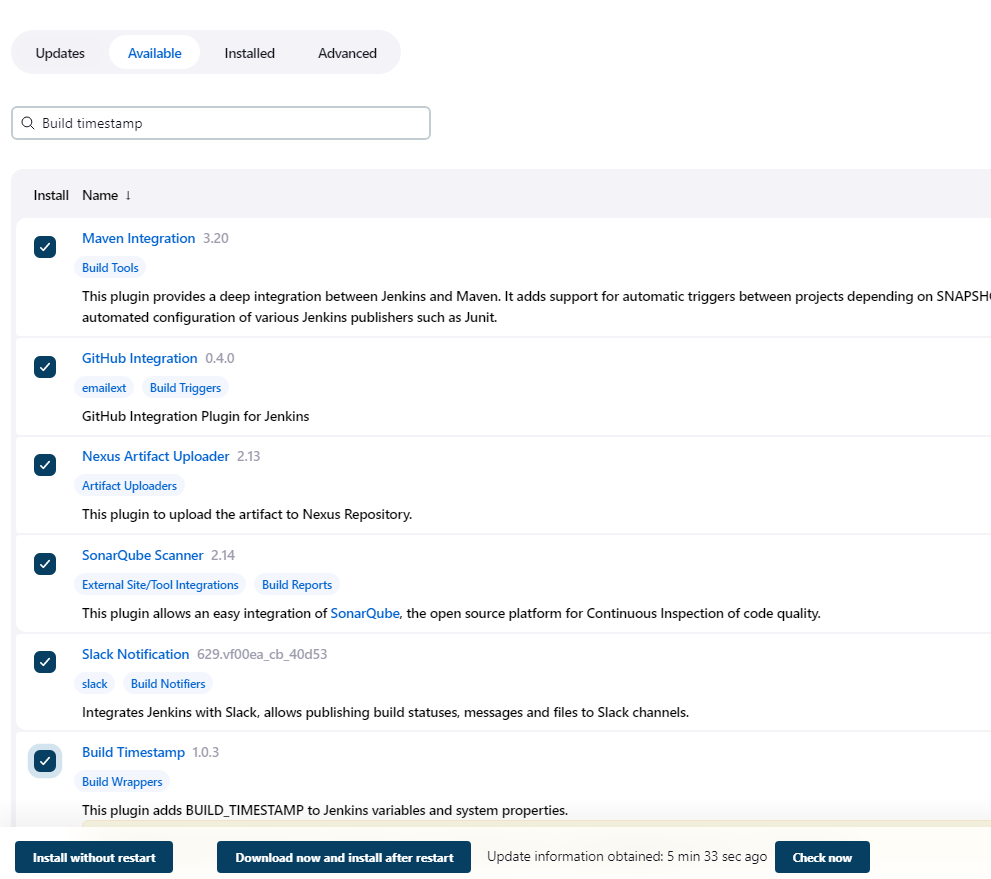
Sonarqube scanner

|

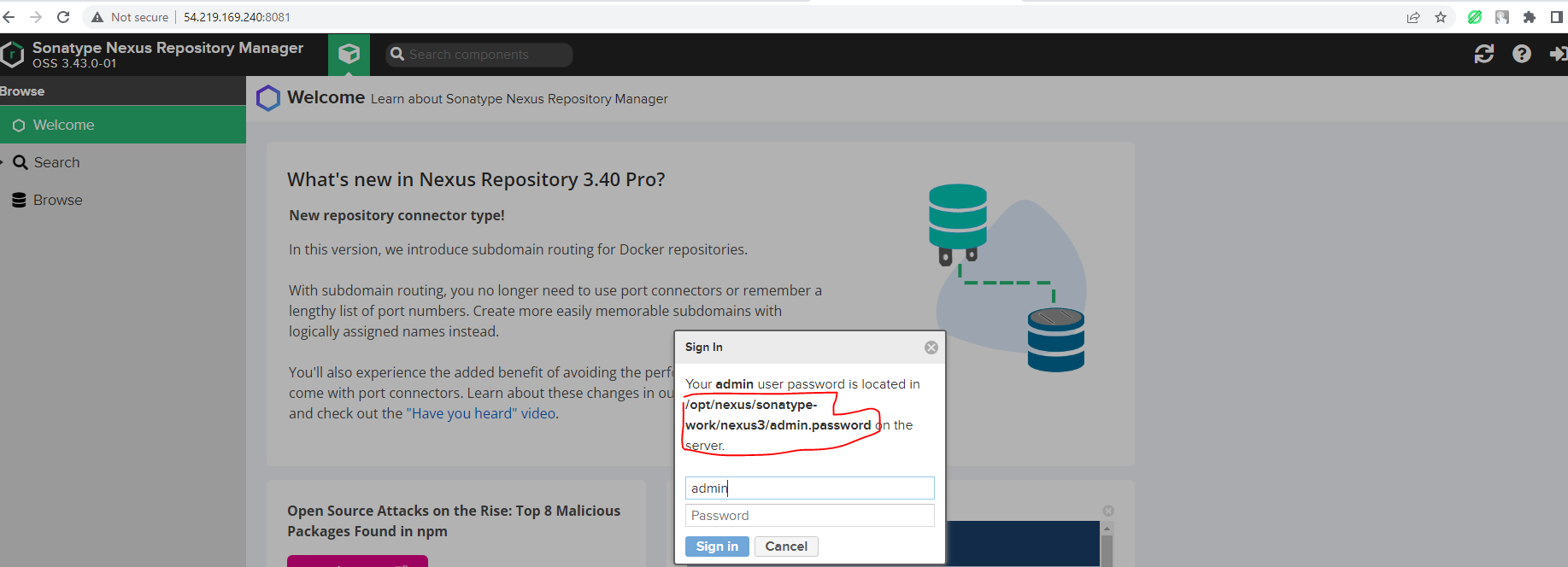
Slack notification

|

Build timestamp



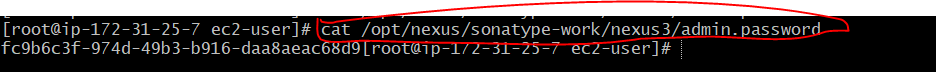
1. Start nexus by running the following command: **systemctl start nexus**
2. Launch nexus URL: <http://54.219.169.240:8081/>



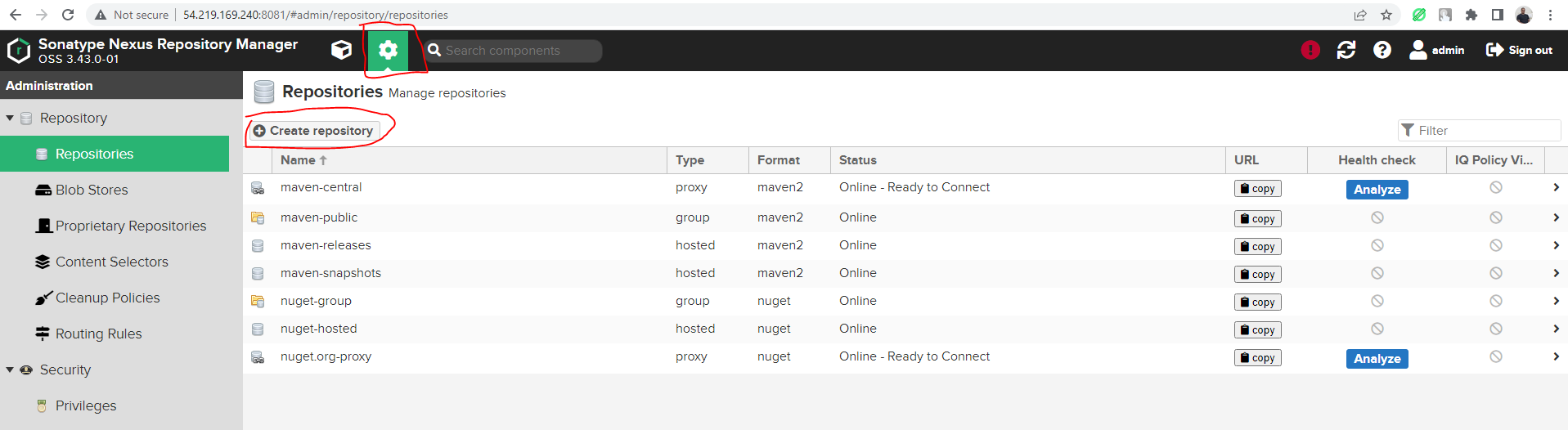
1. Default UserName: admin

Password: we need to copy the above path and paste in command prompt to get the password.

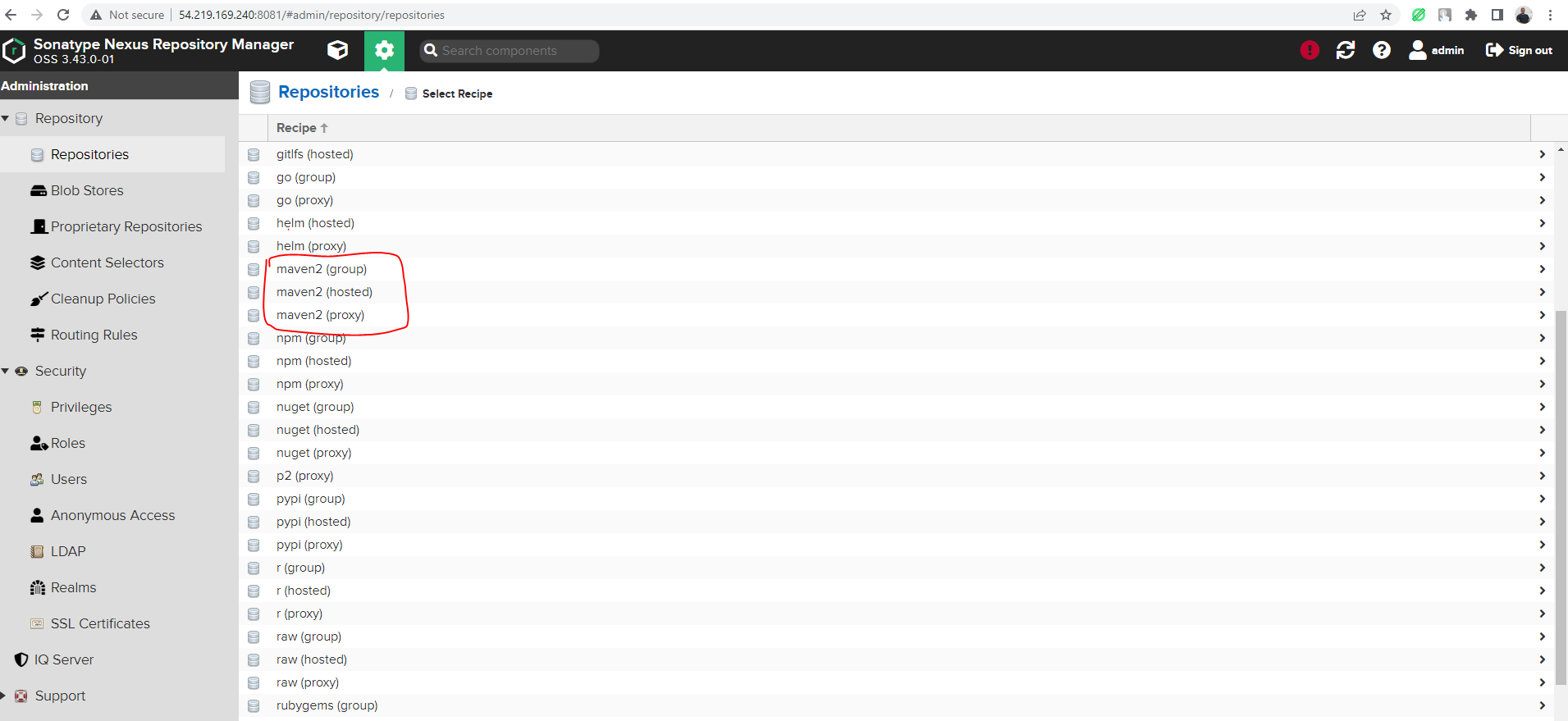
Location: /opt/nexus/sonatype-work/nexus3/admin.password



Updated password: admin

1. Click on repository(gear) icon and click on **“Create repository”**
2. Search for **maven2hosted** : Which stores (store artifact) and then create repository

Repo Name: vprofile-release



1. Search for **maven2proxy**: Which store maven dependencies

Repo Name: vpro-maven-central

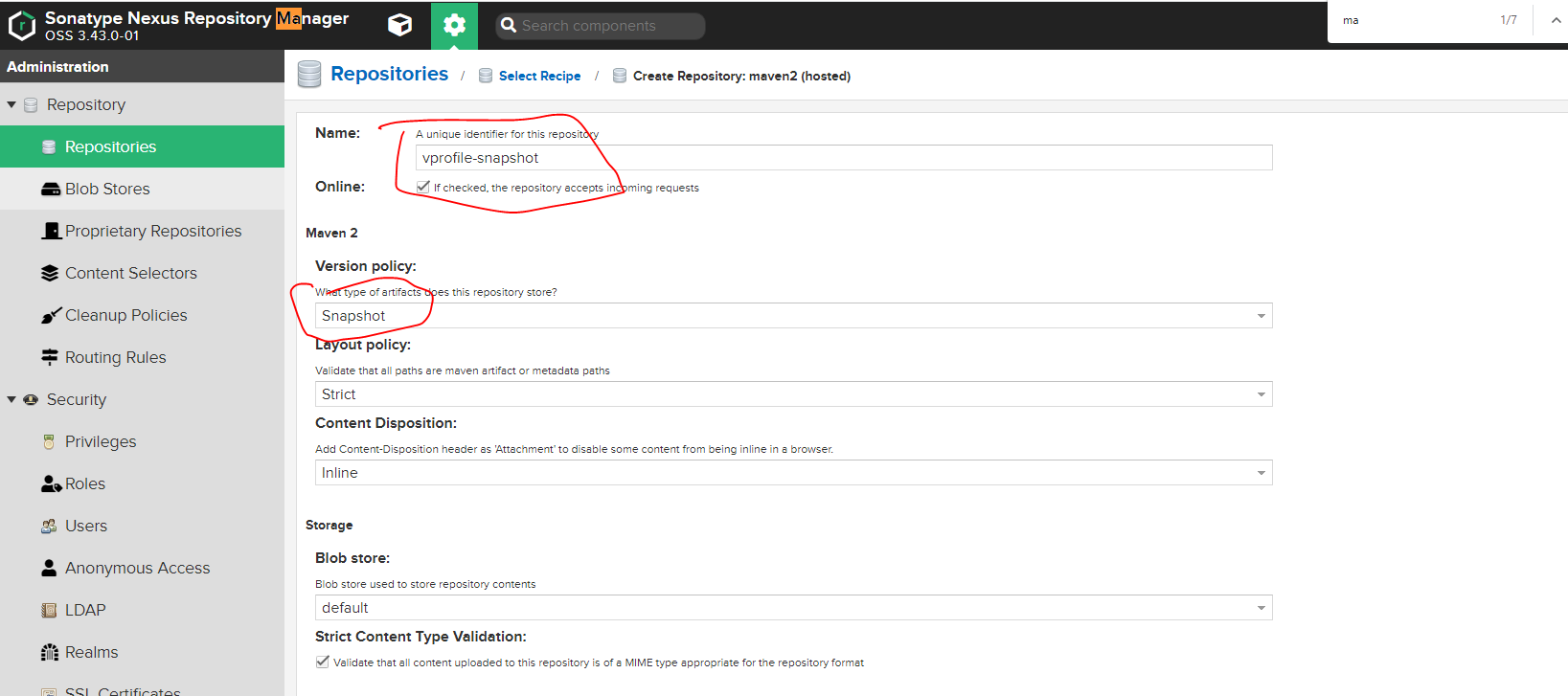
Remote : <https://repo1.maven.org/maven2/>

Then create repository

1. Search for maven2hosted:

Repo Name: vprofile-snapshot

version policy : Snapshot



1. Search for maven2(group):

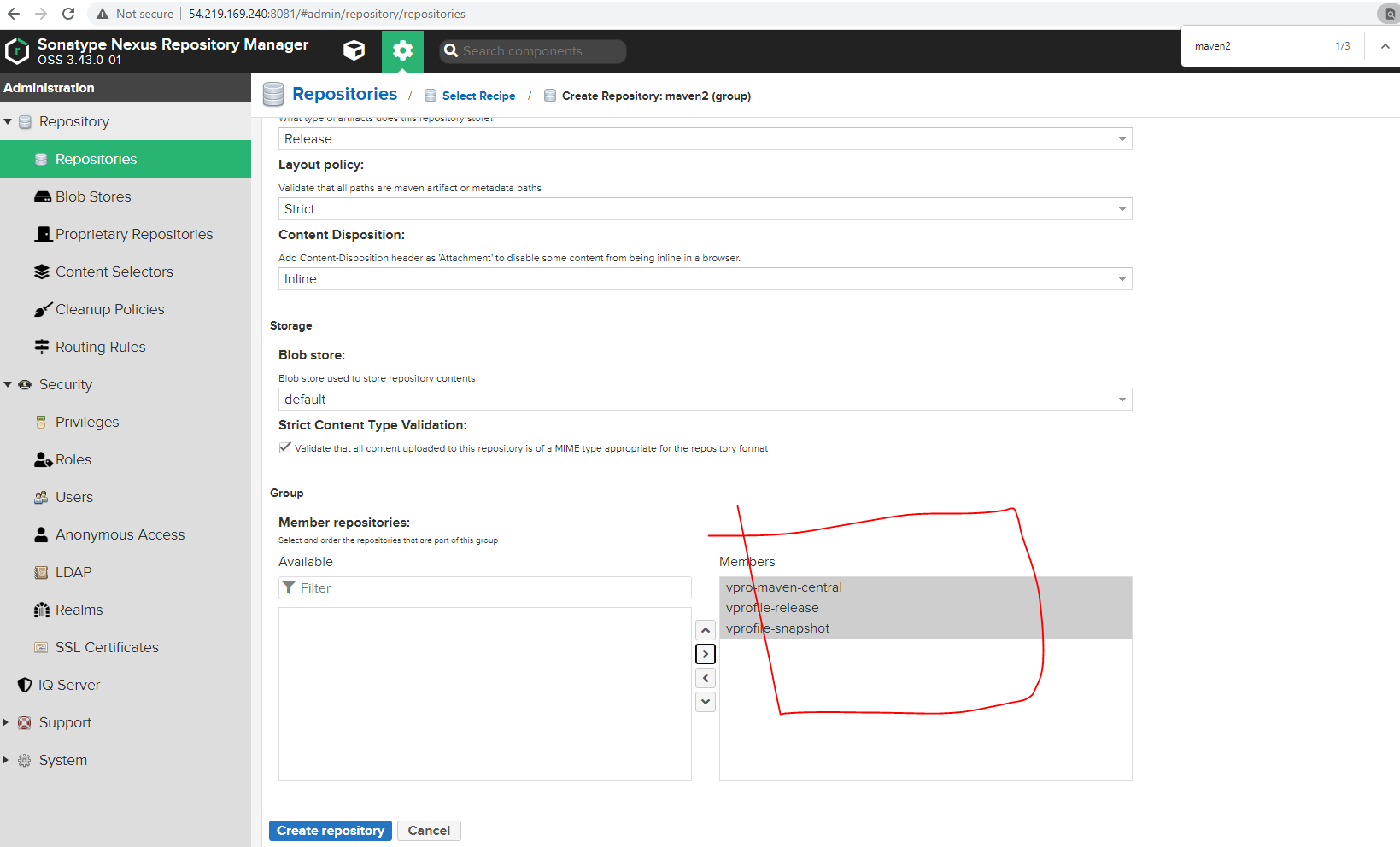
Repo Name: vpro-maven-group

Member repositories > Available :

vpro-maven-central

vrpofile-release

vprofile-snapshot



1. Start sonarqube by running the following command: **systemctl start nexus**
2. Launch sonarqube URL: <http://18.144.18.131/>
3. Login -Default username admin

Default password: admin

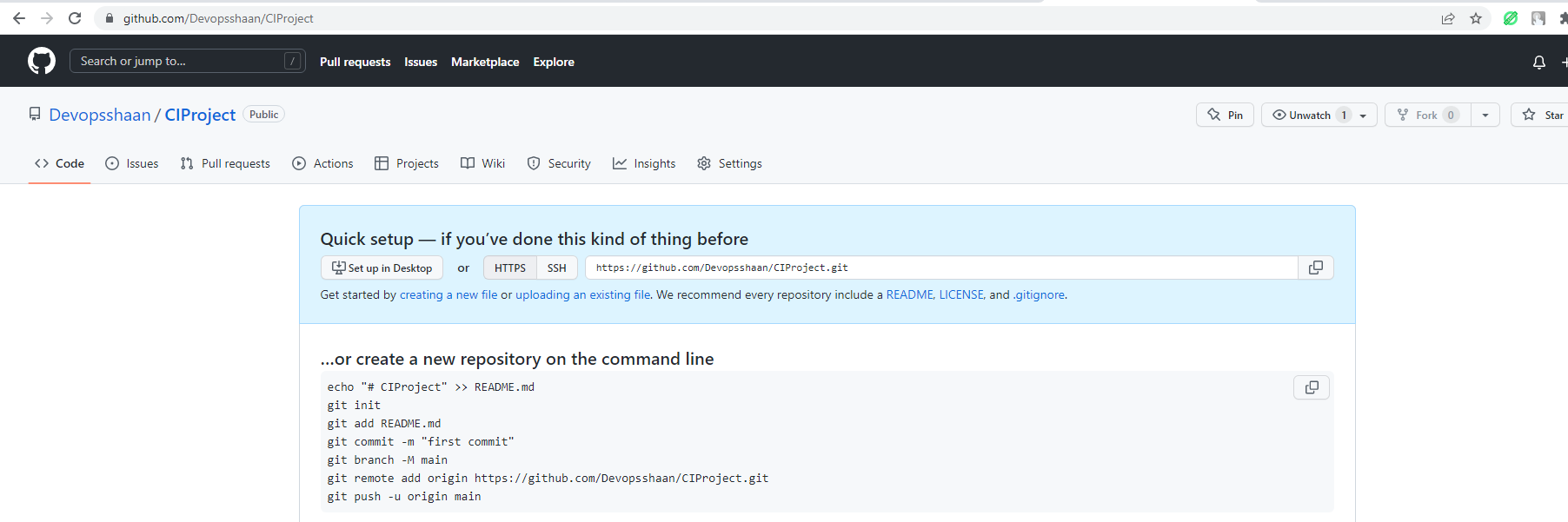
Change password: admin

Token Name : sonartoken

Token : 982aa6447e41132c7ab1409af05604f1e48feb31

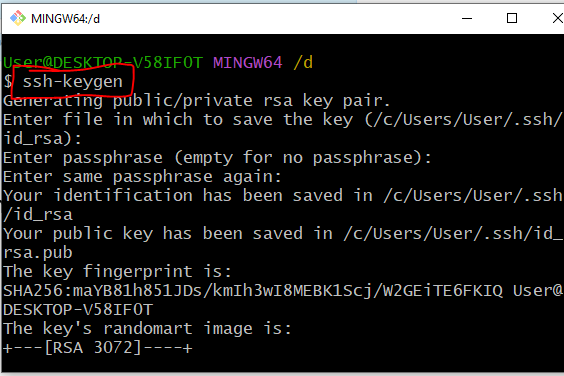
**Step 1: CREATE GITHUB REPOSITORY AND ADD PROJECT**

1. Go to GitHub and create new repository



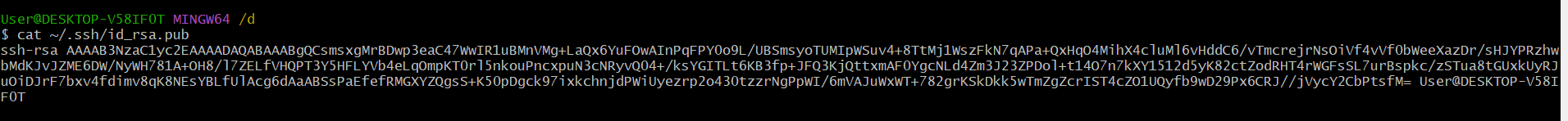
1. To add keys in GitHub, we need to get keys from gitbash for the project we created.
2. Open GitBash and enter –

**ssh-keygen**

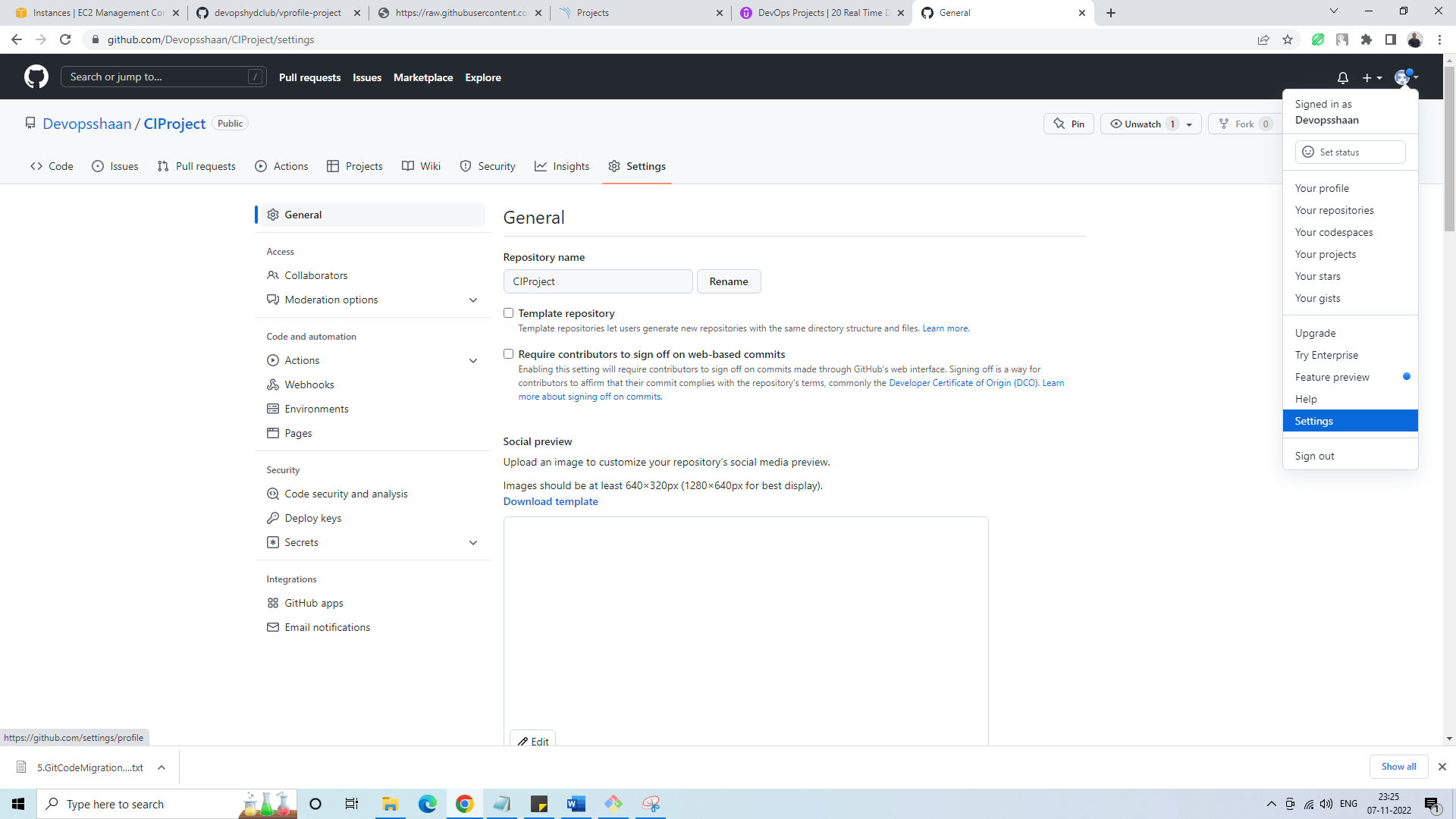


1. Type the below command to get public key

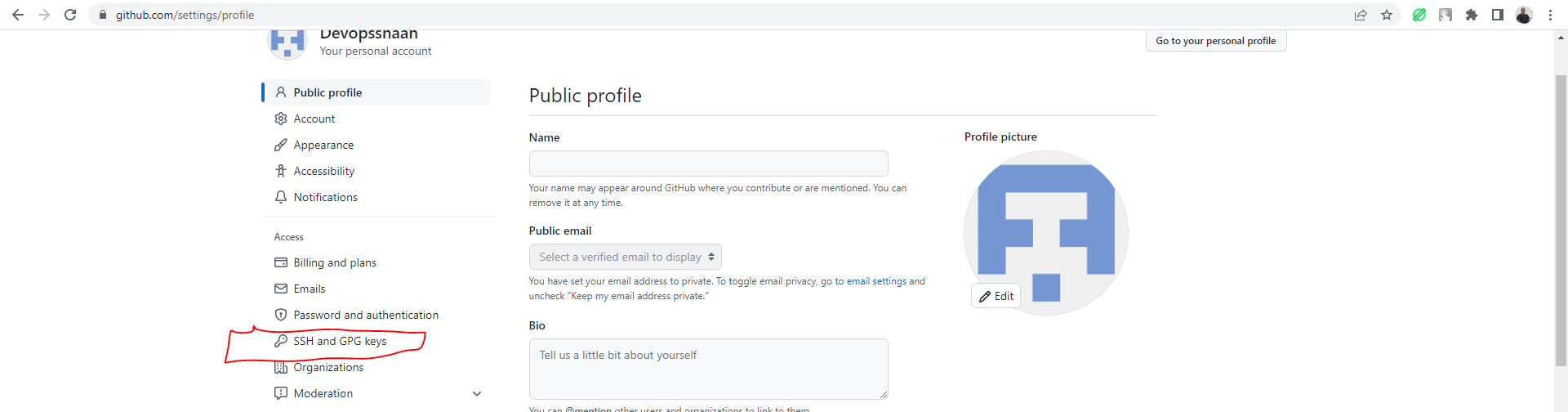
**cat ~/.ssh/id\_rsa.pub**



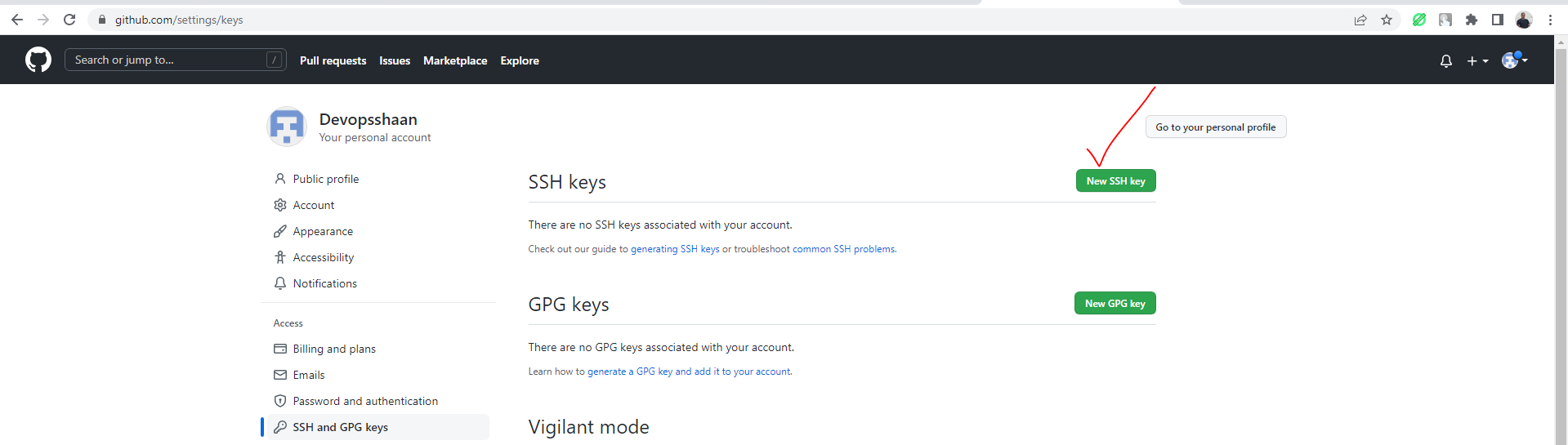
Copy the public key and add in github. Go to Github and click on setting



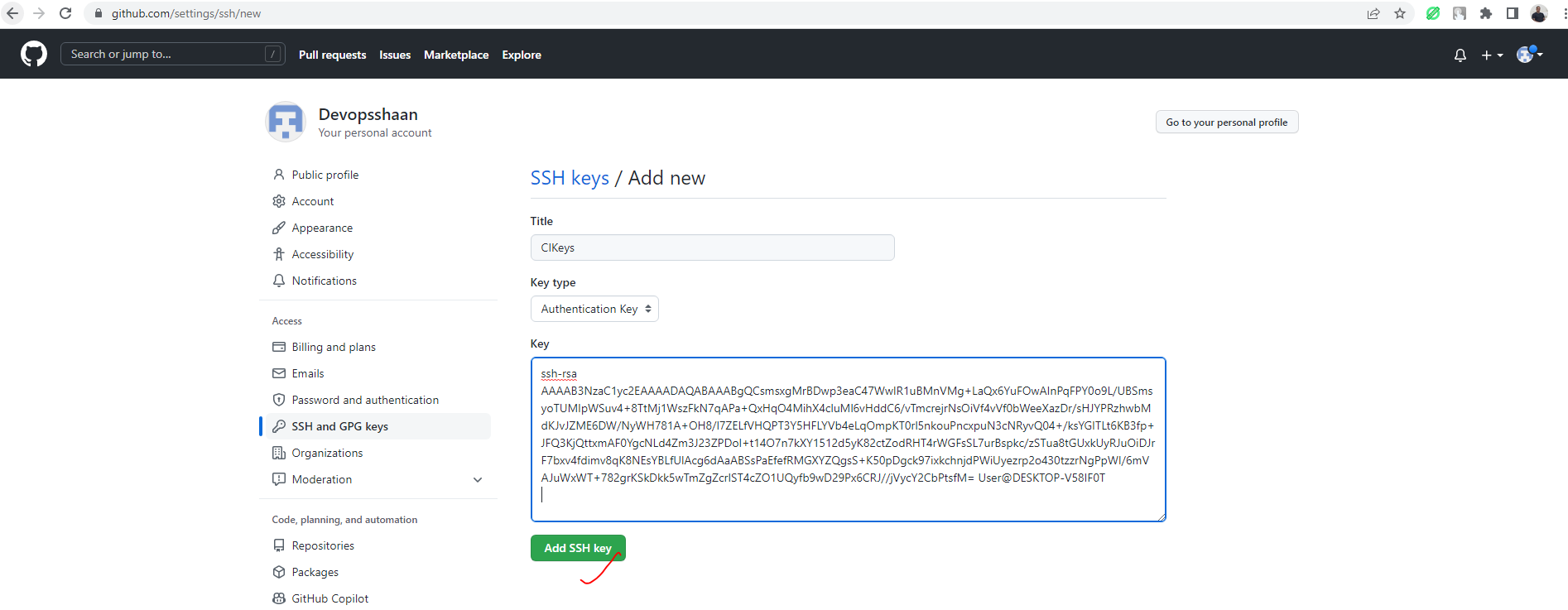
click on SSH and GPG keys as shown below.



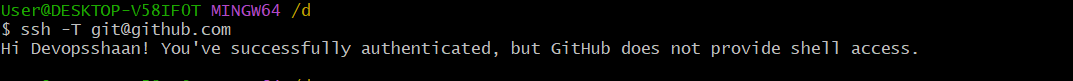
Click on SSH keys



Paste the keys in the key section and click add ssh key



1. After updating the key, we need to verify with below command in git



1. Create a directory in our local

**mkdir projectgitrepo**

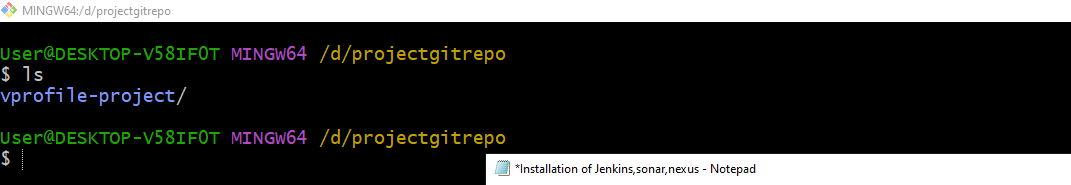
**cd projectgitrepo**

1. Clone the project

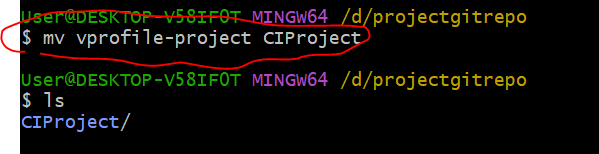
git clone -b ci-jenkins <https://github.com/devopshydclub/vprofile-project.git>

1. Change the project name

vprofile-project to our newly created github name



**mv vprofile-project CIProject**

Changed the project name to CIProject

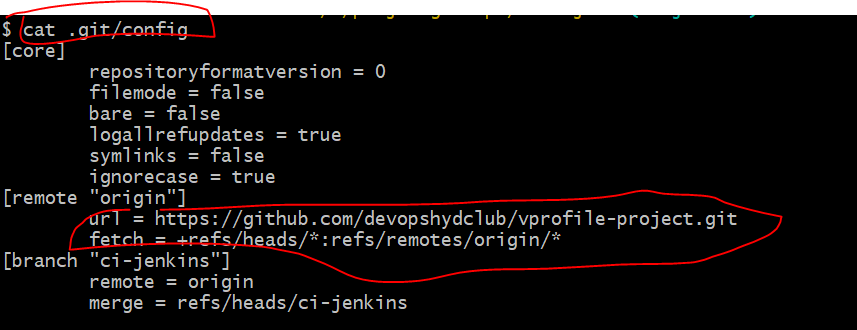
1. Go inside the project :

**cd CIProject**

**ls**

1. Go to config folder from **.git**

**cat .git/config**



In the above screenshot, we are seeing URL which is not our github repository URL, we need to change it

**git remote set-url origin** [**git@github.com:Devopsshaan/CIProject.git**](mailto:git@github.com:Devopsshaan/CIProject.git)



Now again go to config folder to verify, now it is showing our right GitHub repository



1. Now create a branch

**git branch -c main**

1. To push our code to GitHub.

**git push --all origin**

1. Open visual studio editor

**code .**

**STEP 2:**

**Build Job with Nexus Repo**

1. Launch Jenkins Instance from amazon aws.
2. Login to Jenkins URL: <http://13.56.209.171:8080/>

Configure Java and Maven in Jenkins URL

1. Manage Jenkins

|

Global Too Configuration

|

JDK: Add JDK

Name: JAVA\_HOME

Path: /usr/lib/jvm/java-11-openjdk-amd64

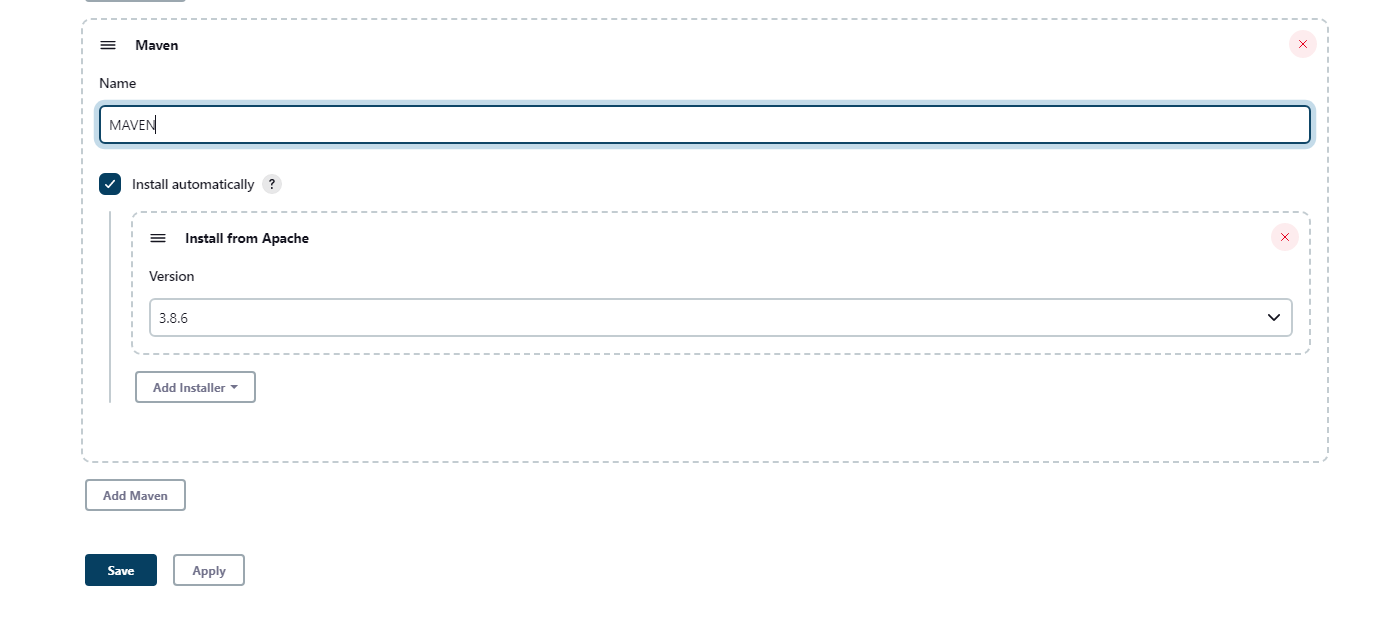
**Note : To find a path execute : ls -l /etc/alternatives/java**

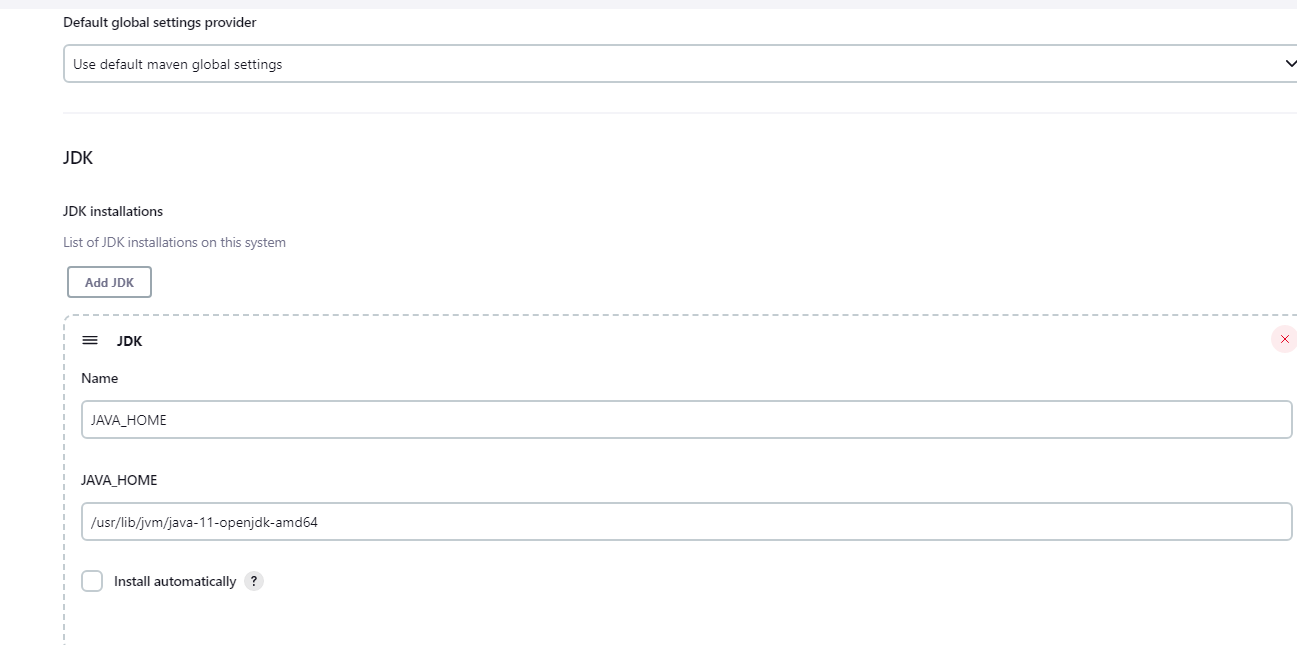
1. Maven: Add Maven

NAME: MAVEN

Version: 3.8.6

Save





1. Configure Nexus Login Credentials
2. Manage Jenkins

|

Manage credentials

|

Global credentials

|

Add Credentials

|

Kind: Username and password

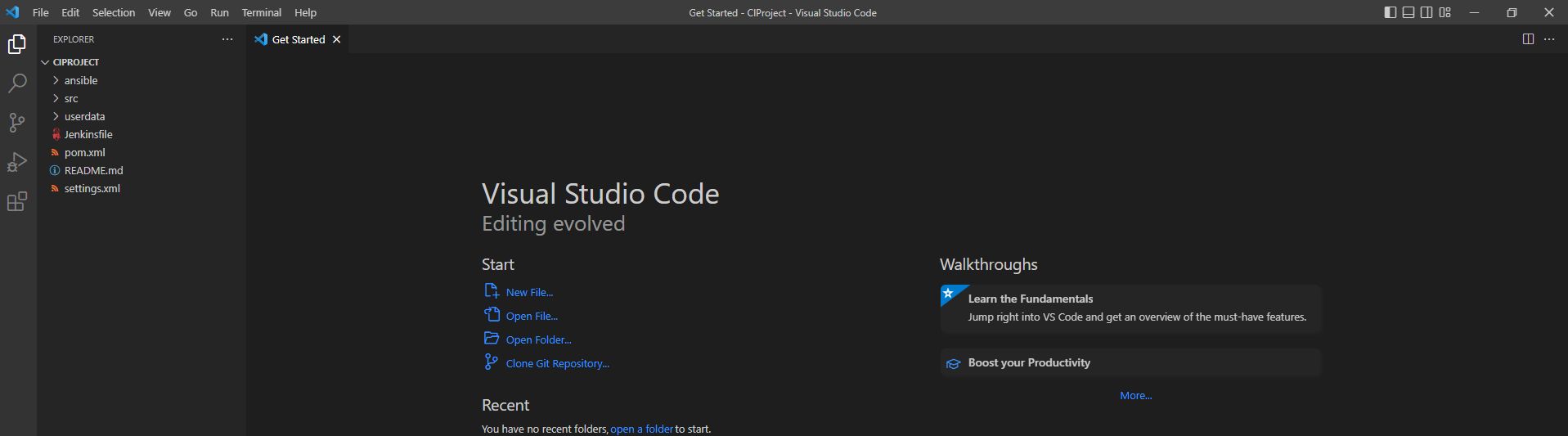
Username: admin

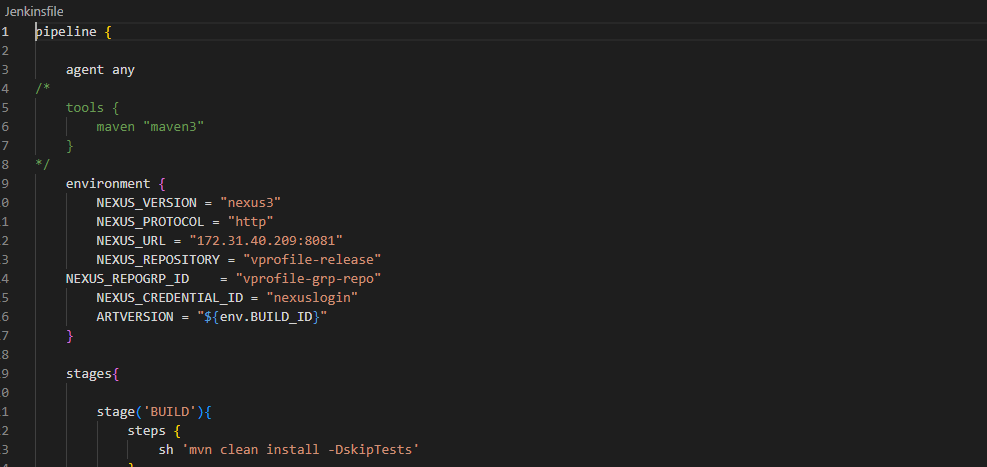
Password: admin

ID: nexuslogin

Description: nexuslogin

1. Launch visual code editor and open jenkins file.



priv

1. Create a pipeline in Jenkins now.

Click on New Item

|

Name : viprofile-ci-pipeline (Select Pipeline)

|

Pipeline script from SCM

|

SCM :Git

|

Repository URL : [git@github.com:Devopsshaan/CIProject.git](mailto:git@github.com:Devopsshaan/CIProject.git)

|

Get the private key:

Launch GitBash and navigate to the project- Fire the command : cat ~/.ssh/id\_rsa

paste private key in the Jenkins(Add global credentials)

Branch : \*/ci-jenkins

JenkinsFile

Save

Note: If you’re getting error at Repository URL

login to jenkins virtual machine

sudo -i

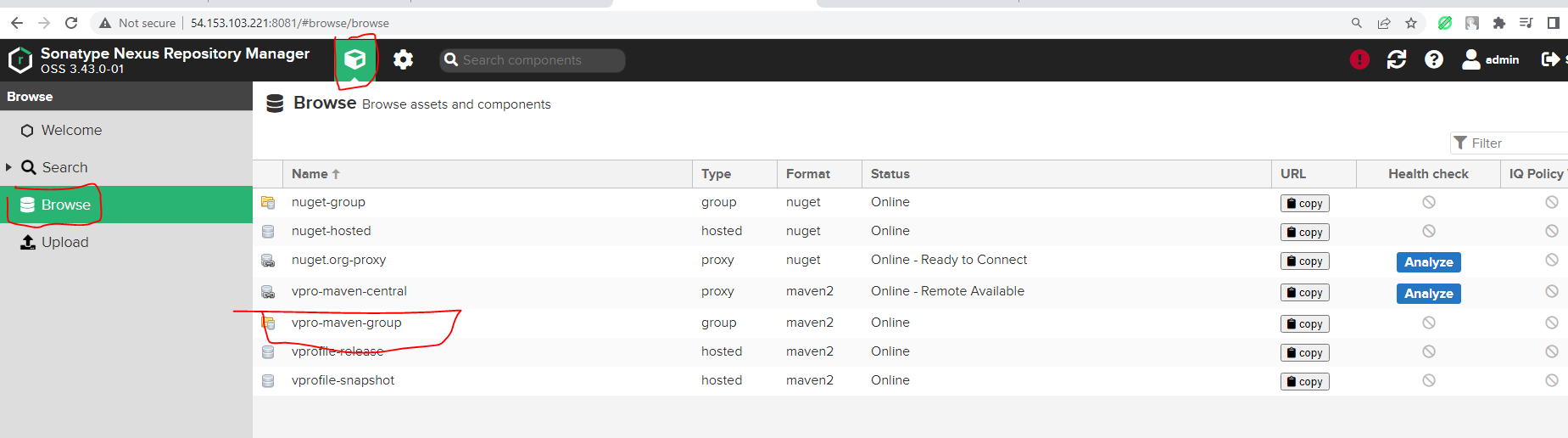
su - jenkins

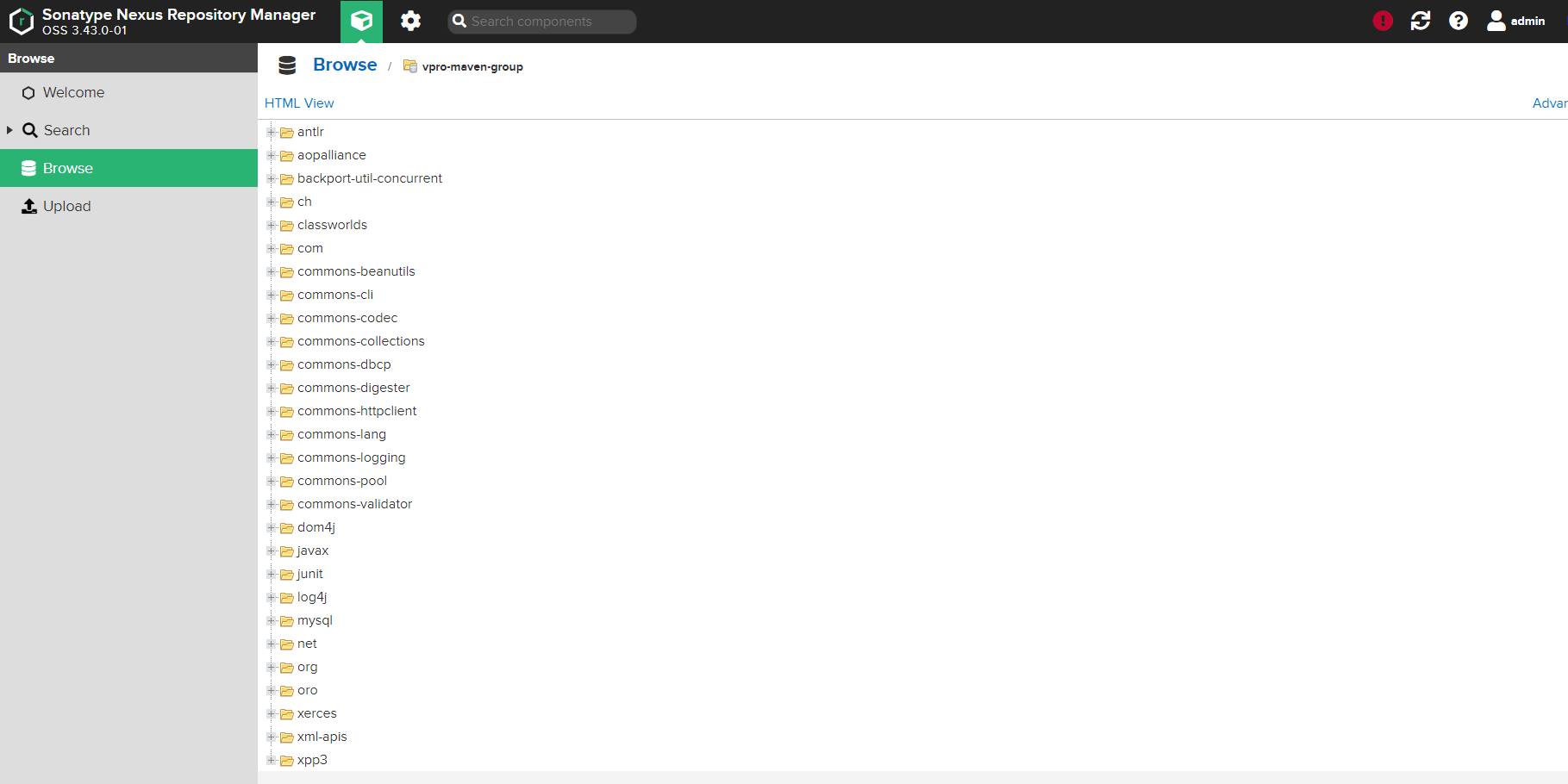
git ls-remote -h git@github.com:Devopsshaan/CIProject.git

cat .ssh/known\_hosts

**Build Now**

--After success in Jenkins, come to Nexus repository and click on browser ->vpro-maven-pro

To find the repositories got stores



**Jenkins File**

pipeline {

    agent any

    tools {

        maven "MAVEN"

        jdk "OracleJDK8"

    }

    environment {

        SNAP\_REPO = 'vprofile-snapshot'

        NEXUS\_USER = 'admin'

        NEXUS\_PASS = 'admin'

        RELEASE\_REPO = 'vprofile-release'

        CENTRAL\_REPO = 'vpro-maven-central'

        NEXUSIP = '172.31.25.7'

        NEXUSPORT = '8081'

        NEXUS\_GRP\_REPO = 'vpro-maven-group'

        NEXUS\_LOGIN = 'nexuslogin'

    }

    stages {

        stage('Build'){

            steps {

                sh 'mvn -s settings.xml -DskipTests install'

            }

        }

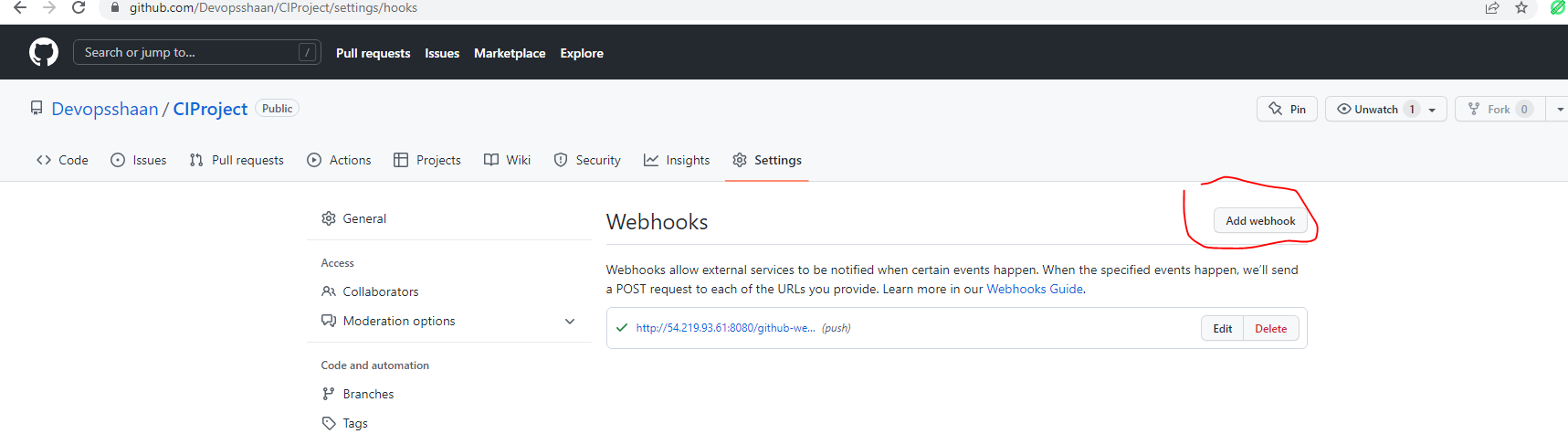
    }

}

**STEP 3:**

**Configure WEBHOOK URL in Jenkins.**

1. To add Jenkins webhook URL, go to GitHub and click on settings.
2. Click on webhooks
3. Click on Add webhooks
4. Payload URL : <http://54.219.93.61:8080/github-webhook/>



1. In Jenkins File Update and configure.

Login to Jenkins file, click on configure.

Checkmark: GitHub hook trigger for GITScm polling

Update the file

Save

pipeline {

    agent any

    tools {

        maven "MAVEN"

        jdk "OracleJDK8"

    }

    environment {

        SNAP\_REPO = 'vprofile-snapshot'

        NEXUS\_USER = 'admin'

        NEXUS\_PASS = 'admin'

        RELEASE\_REPO = 'vprofile-release'

        CENTRAL\_REPO = 'vpro-maven-central'

        NEXUSIP = '172.31.25.7'

        NEXUSPORT = '8081'

        NEXUS\_GRP\_REPO = 'vpro-maven-group'

        NEXUS\_LOGIN = 'nexuslogin'

    }

    stages {

        stage('Build'){

            steps {

                sh 'mvn -s settings.xml -DskipTests install'

            }

            post {

                success {

                    echo "Now Archiving."

                    archiveArtifacts artifacts: '\*\*/\*.war'

                }

            }

        }

        stage('Test'){

            steps {

                sh 'mvn -s settings.xml test'

            }

        }

        stage('Checkstyle Analysis'){

            steps {

                sh 'mvn -s settings.xml checkstyle:checkstyle'

            }

        }

    }

}

**STEP 4:**

**Code analysis with SonarQube**

1. Login to Jenkins and configure credentials for SonarQube
2. In Jenkins

Click on Manage Jenkins

|

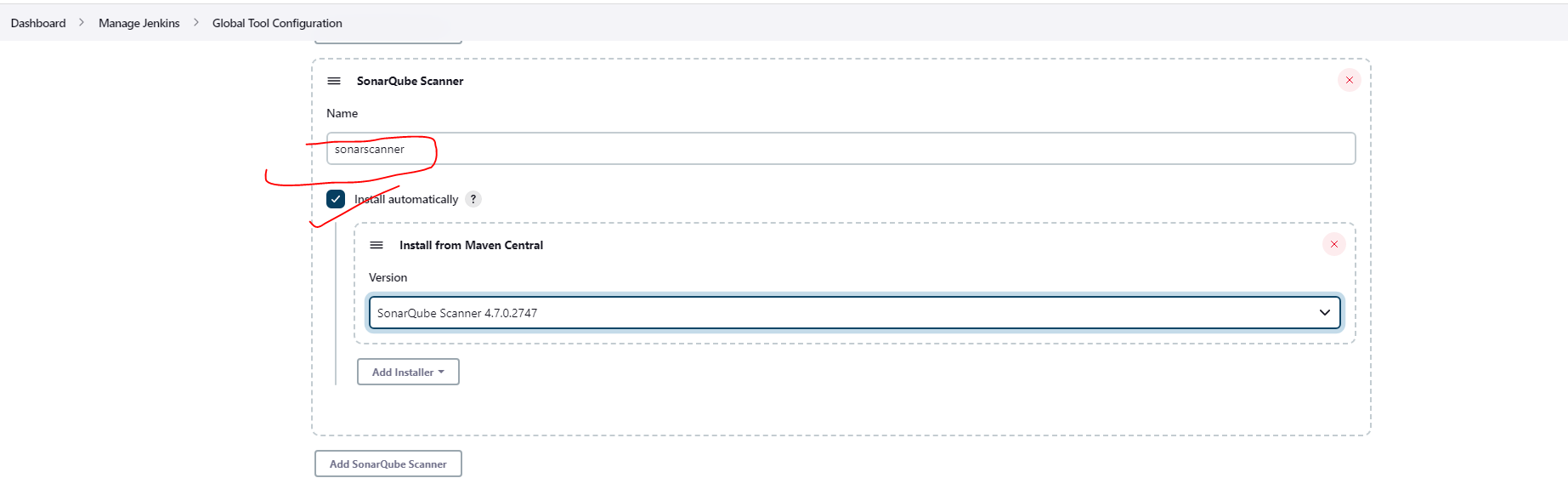
Global tool configuration

|

Search for SonarQube Scanner -> Add SonarQube Scanner

Name: sonarscanner

Check: Install automatically



Save.

1. Configure system to store sonarqube server in jenkins

Manage Jenkins

|

Configure system

|

Search for : SonarQube servers

|

Add SonarQube

|

Name: sonarserver

sever: sonarprivateip

authentication token: login to sonarqube -settings-token

secret text

secret: paste the token

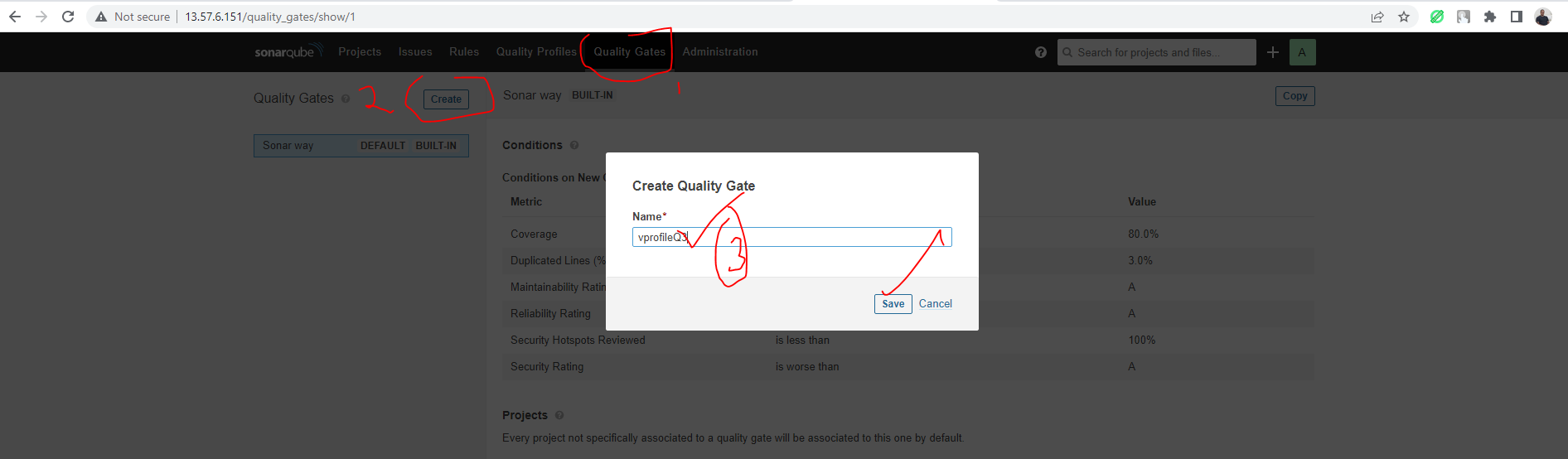
ID: sonartoken

save

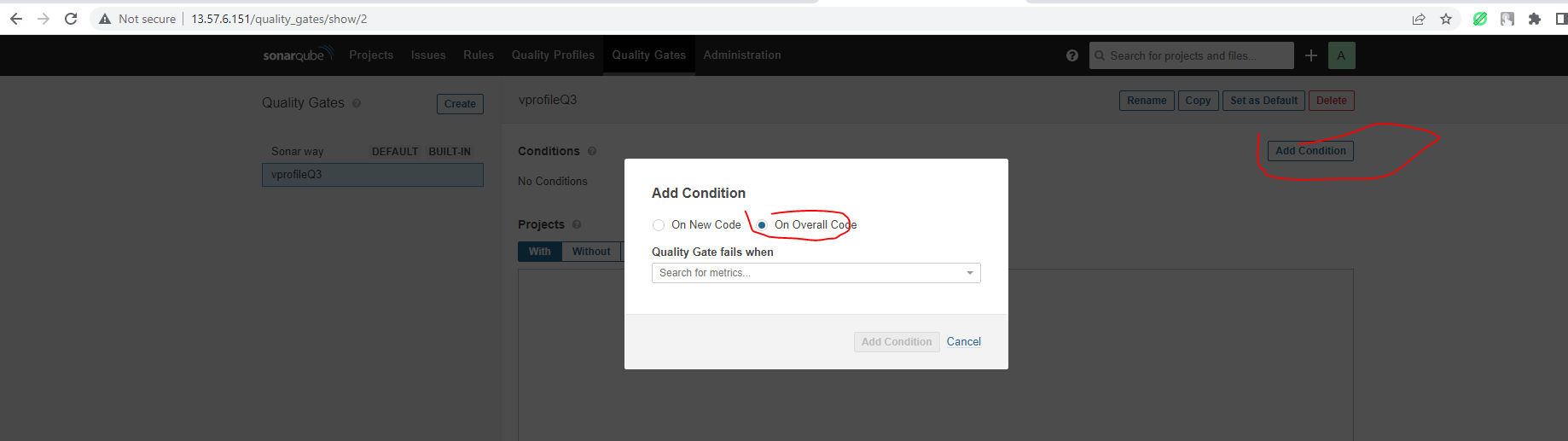
**STEP 4:**

**Quality Gates in SonarQube.**

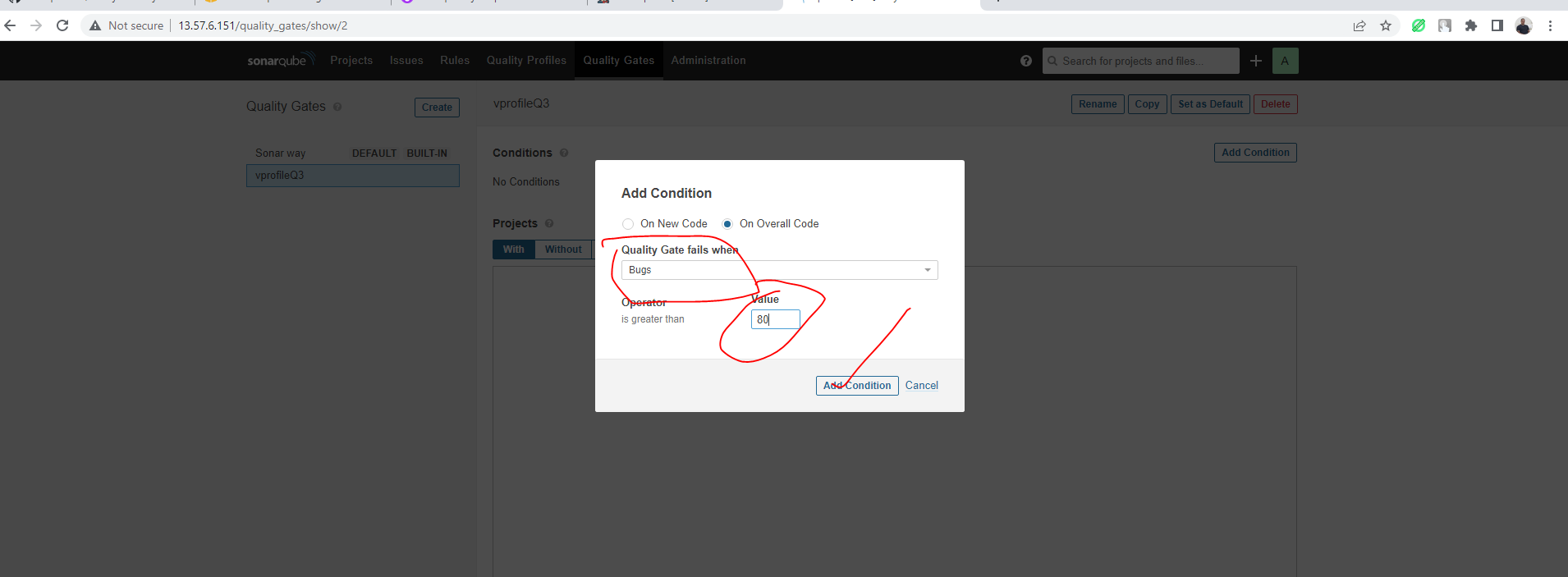
1. Set up quality gates for sonarqube
2. In sonarqube > click on quality gates > and click on create



1. In the next page, click on Add condition
2. Click on overall condition

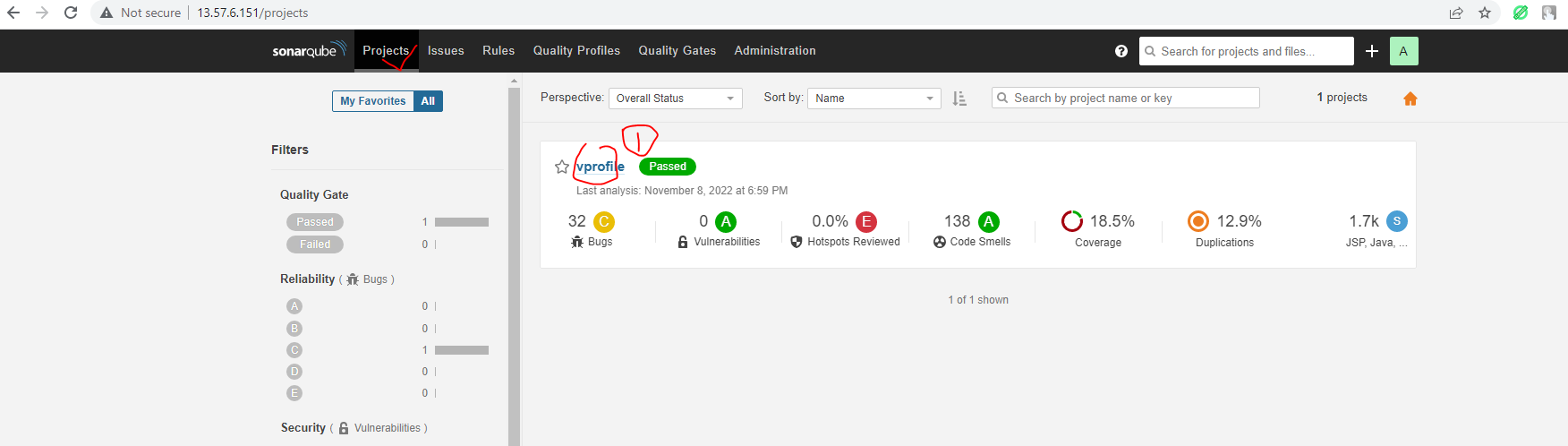


1. Select Bugs from dropdown and provide the condition if bug is below 80 it is passed or if it crosses more than 80 then it is failed

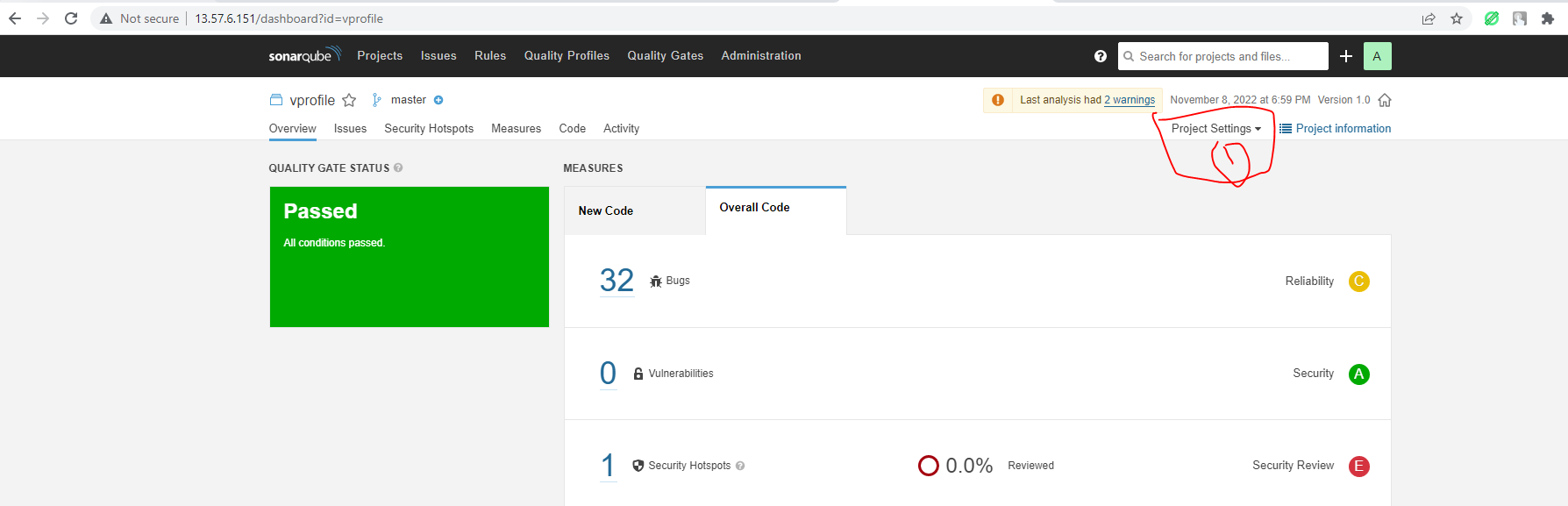


1. Select quality gate option for the project

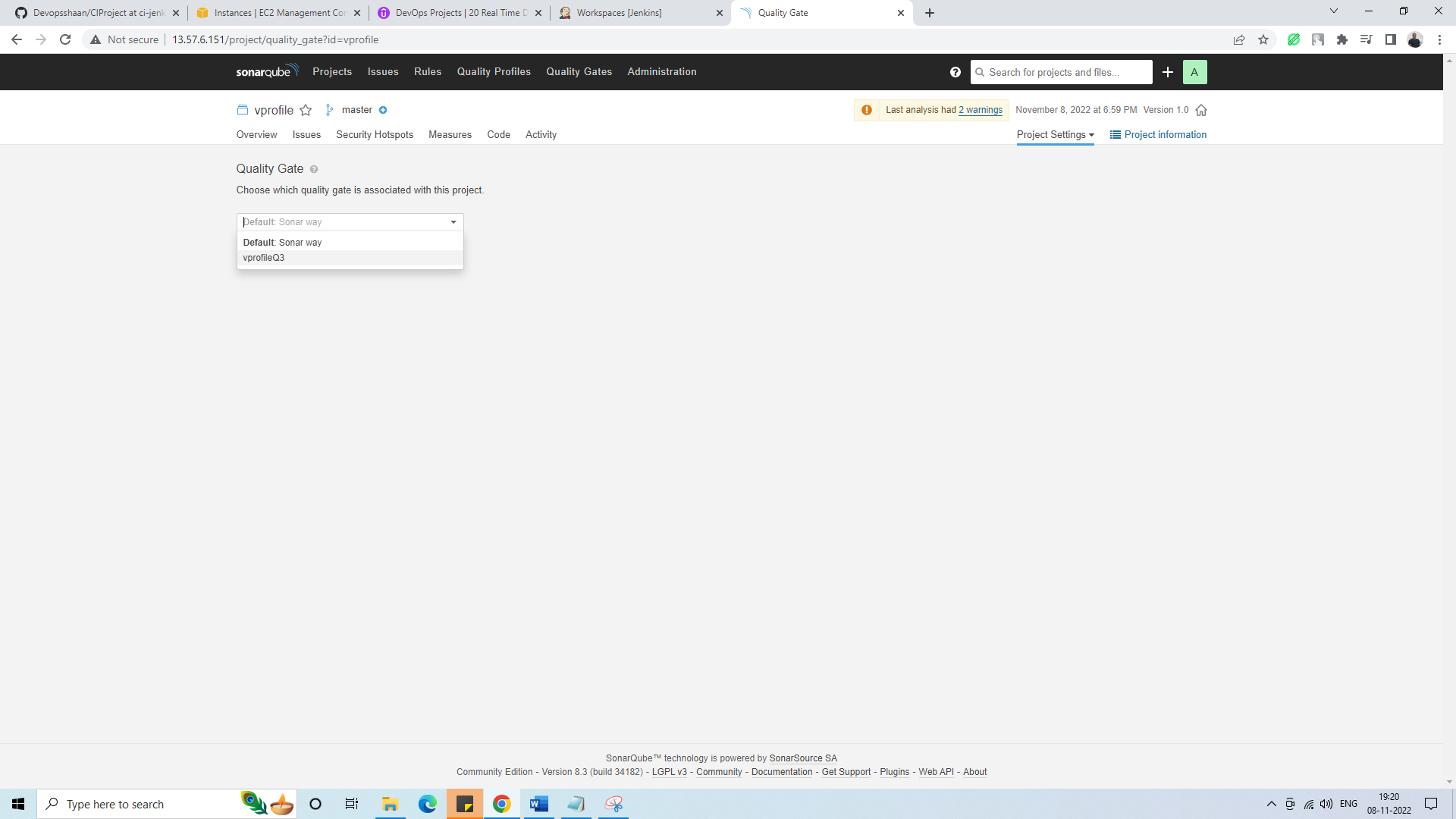
Click on Project > Click on project name



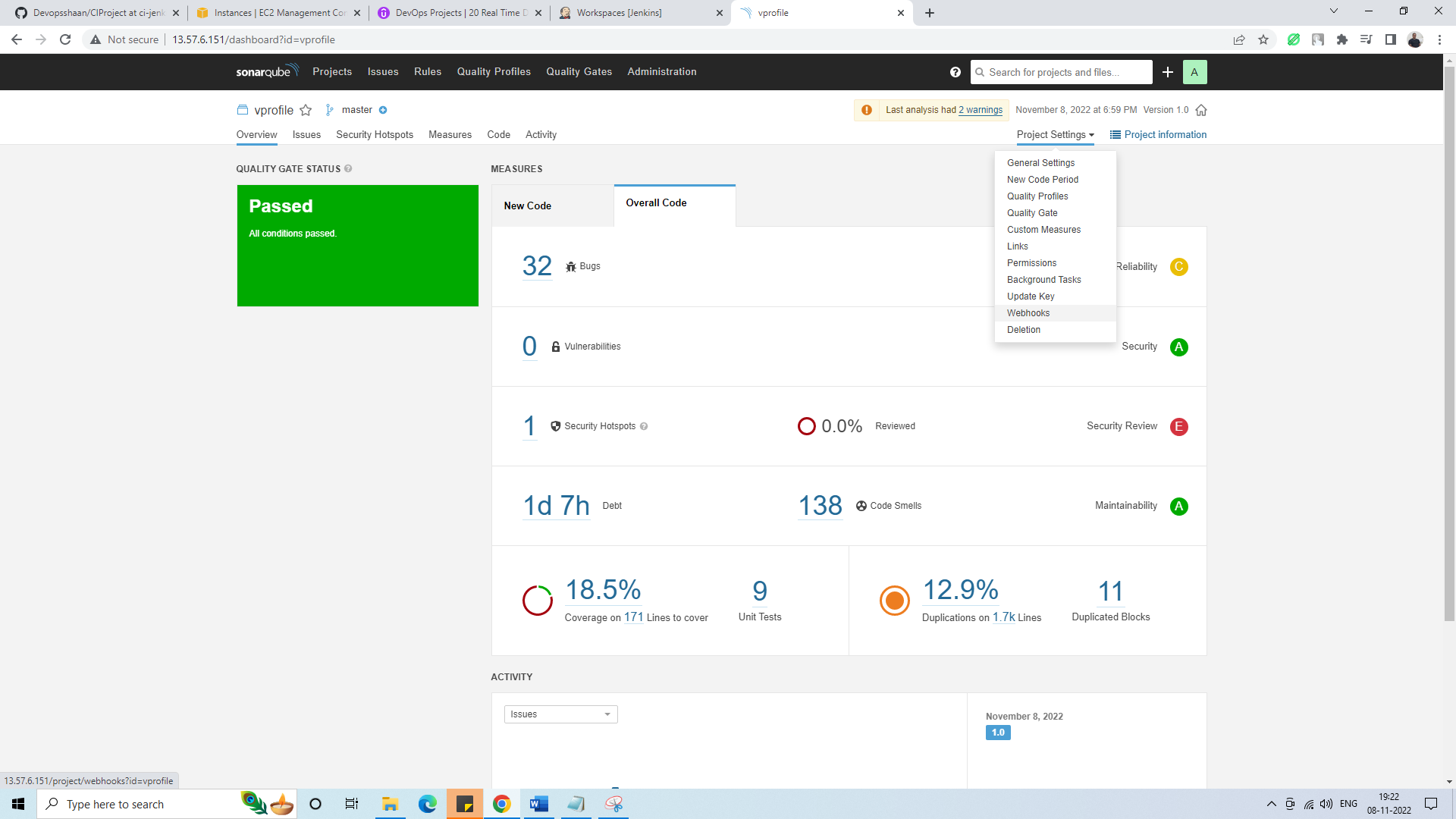
1. Click on project setting dropdown and select quality gates



1. Select the qualitygate name created.

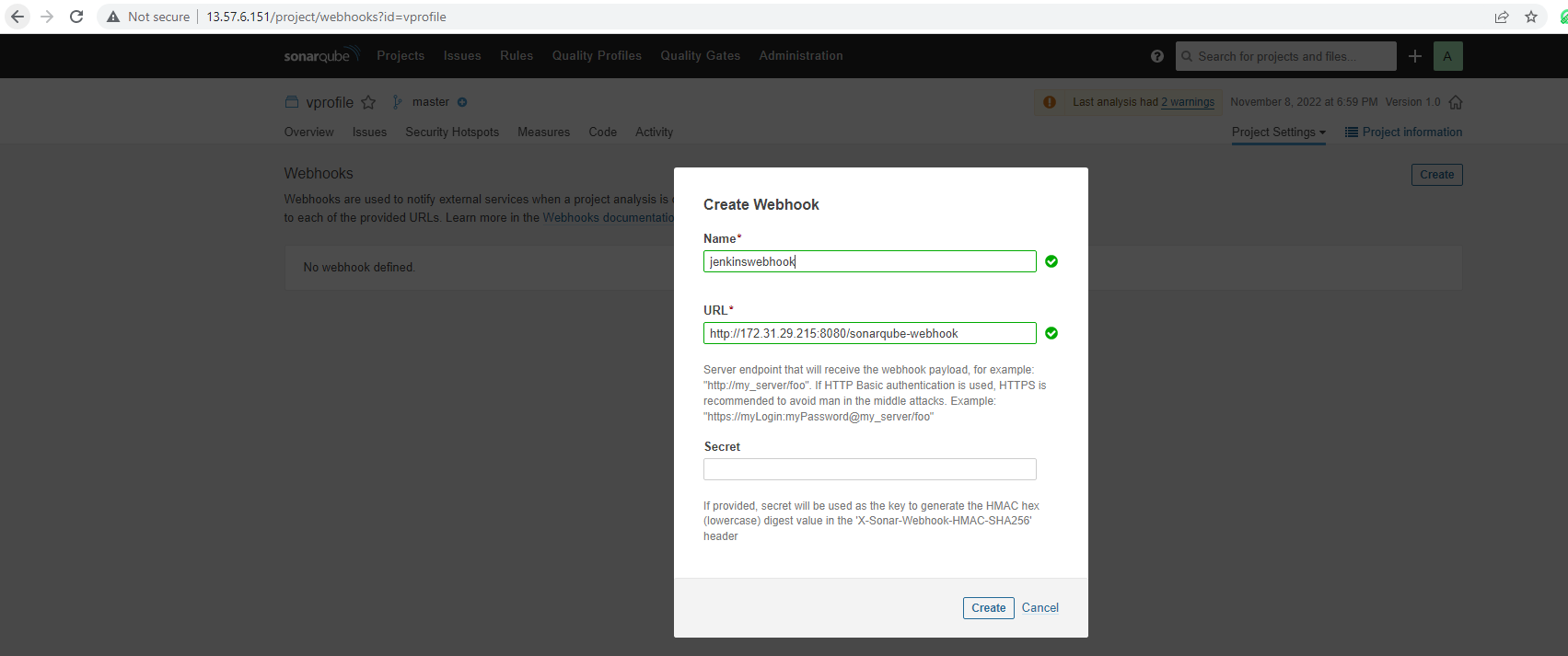


1. Now create webhook in the same page



1. Provide webhook URL details

url: jenkinsprivateip:8080/sonarqube-webhook



JenkinsFile

pipeline {

    agent any

    tools {

        maven "MAVEN"

        jdk "OracleJDK8"

    }

    environment {

        SNAP\_REPO = 'vprofile-snapshot'

        NEXUS\_USER = 'admin'

        NEXUS\_PASS = 'admin'

        RELEASE\_REPO = 'vprofile-release'

        CENTRAL\_REPO = 'vpro-maven-central'

        NEXUSIP = '172.31.25.7'

        NEXUSPORT = '8081'

        NEXUS\_GRP\_REPO = 'vpro-maven-group'

        NEXUS\_LOGIN = 'nexuslogin'

        SONARSERVER = 'sonarserver'

        SONARSCANNER = 'sonarscanner'

    }

    stages {

        stage('Build'){

            steps {

                sh 'mvn -s settings.xml -DskipTests install'

            }

            post {

                success {

                    echo "Now Archiving."

                    archiveArtifacts artifacts: '\*\*/\*.war'

                }

            }

        }

        stage('Test'){

            steps {

                sh 'mvn -s settings.xml test'

            }

        }

        stage('Checkstyle Analysis'){

            steps {

                sh 'mvn -s settings.xml checkstyle:checkstyle'

            }

        }

        stage('Sonar Analysis') {

            environment {

                scannerHome = tool "${SONARSCANNER}"

            }

            steps {

               withSonarQubeEnv("${SONARSERVER}") {

                   sh '''${scannerHome}/bin/sonar-scanner -Dsonar.projectKey=vprofile \

                   -Dsonar.projectName=vprofile \

                   -Dsonar.projectVersion=1.0 \

                   -Dsonar.sources=src/ \

                   -Dsonar.java.binaries=target/test-classes/com/visualpathit/account/controllerTest/ \

                   -Dsonar.junit.reportsPath=target/surefire-reports/ \

                   -Dsonar.jacoco.reportsPath=target/jacoco.exec \

                   -Dsonar.java.checkstyle.reportPaths=target/checkstyle-result.xml'''

              }

            }

        }

         stage("Quality Gate") {

            steps {

                timeout(time: 1, unit: 'HOURS') {

                    // Parameter indicates whether to set pipeline to UNSTABLE if Quality Gate fails

                    // true = set pipeline to UNSTABLE, false = don't

                    waitForQualityGate abortPipeline: true

                }

            }

        }

    }

}

**STEP 5:**

**PUBLISH TO QUALITY GATES.**

1. Go to jenkins
2. Configure system
3. Build Timestamp
4. checkmark Enable Build\_Timestamp pattern: yy-MM-dd-HHmm
5. save

**STEP 6:**

**SLACK NOTIFICATION**

Slack Notification :

-search for slack login

-create a workspace : vprofilecicd

-what are you working

devopscicd

-add channel :jenkinscicd --create

-search in google slack app

search for jenkins ci

-add to slack

-choose channel(which we created)

-copy the token and store in some where

-----Now go to jenkins and integrate with token

Manage jenkins

configure systems

find :slack

workspace : vprofilecicd

credentails :

add

kind :secret text

paste the slack token

ID: slacktoken

Description :slacktoken

save

channel : #jenkinscicd

save