**INTEGRATING SONARQUBE WITH JENKINS:**

Let’s try executing the SonarQube testing for the code from local terminal and see the SonarQube server data processing.

First let’s go to the folder of the project which I have cloned earlier to test the code:

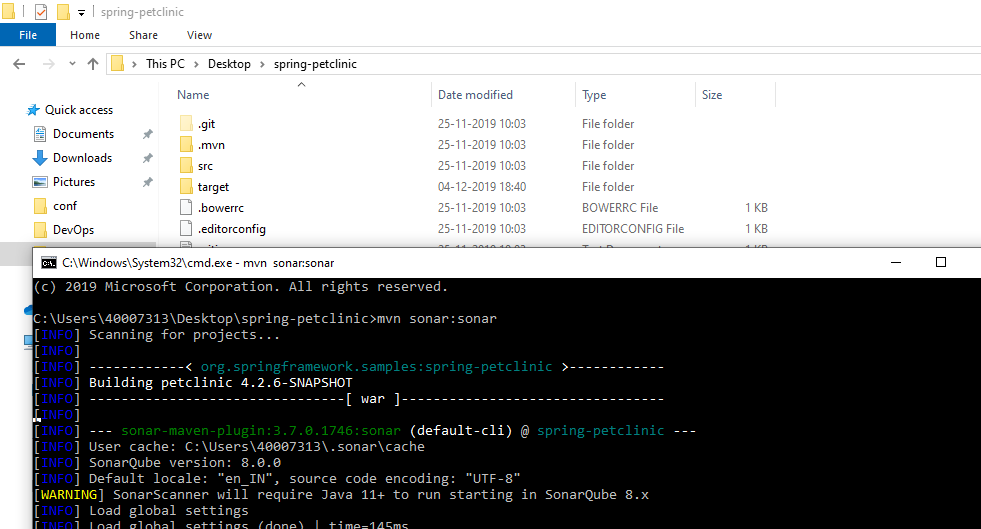
And here I am using maven to execute this because maven has inbuilt plugins for sonar testing of code let’s try that:

Prerequisites are:

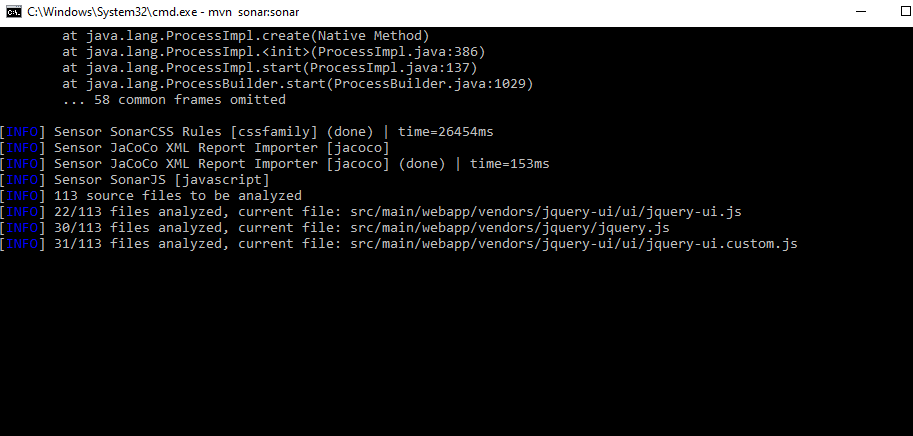
* Should have maven installed
* Jenkins must have been installed
* Should have java 11 and java 1.8 path for Jenkins and
* Should have SonarQube installed

Now let’s see the execution locally instead of Jenkins:

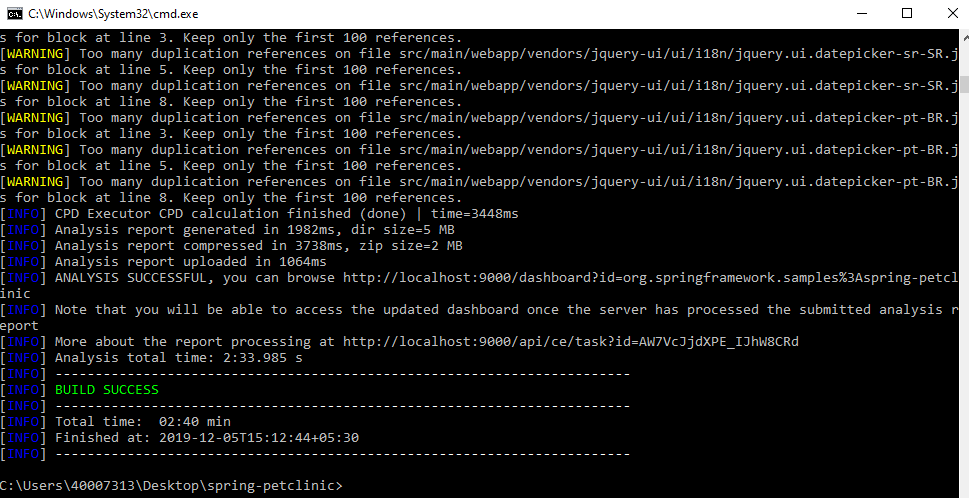
By using command: “mvn sonar:sonar “



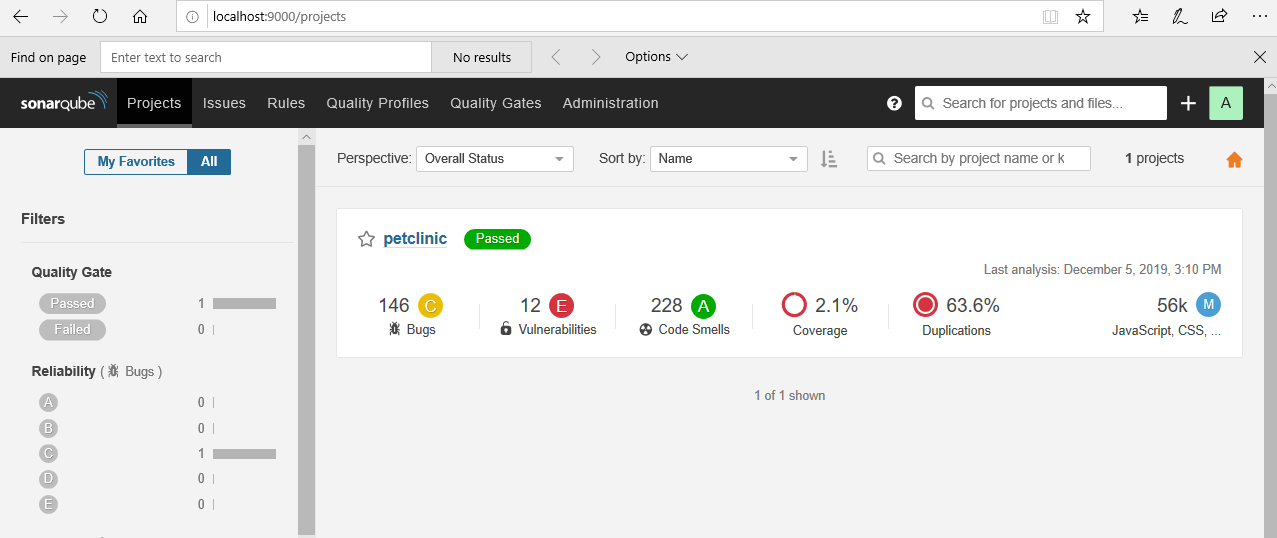
The above snapshot tells that I am in project folder and executed the command “mvn sonar:sonar”. It started processing files.



Now the picture down tells that build is success and report are processing in server “ <http://localhost:9000>”



Now let’s go to the server and observe the data how is it processed and displayed:



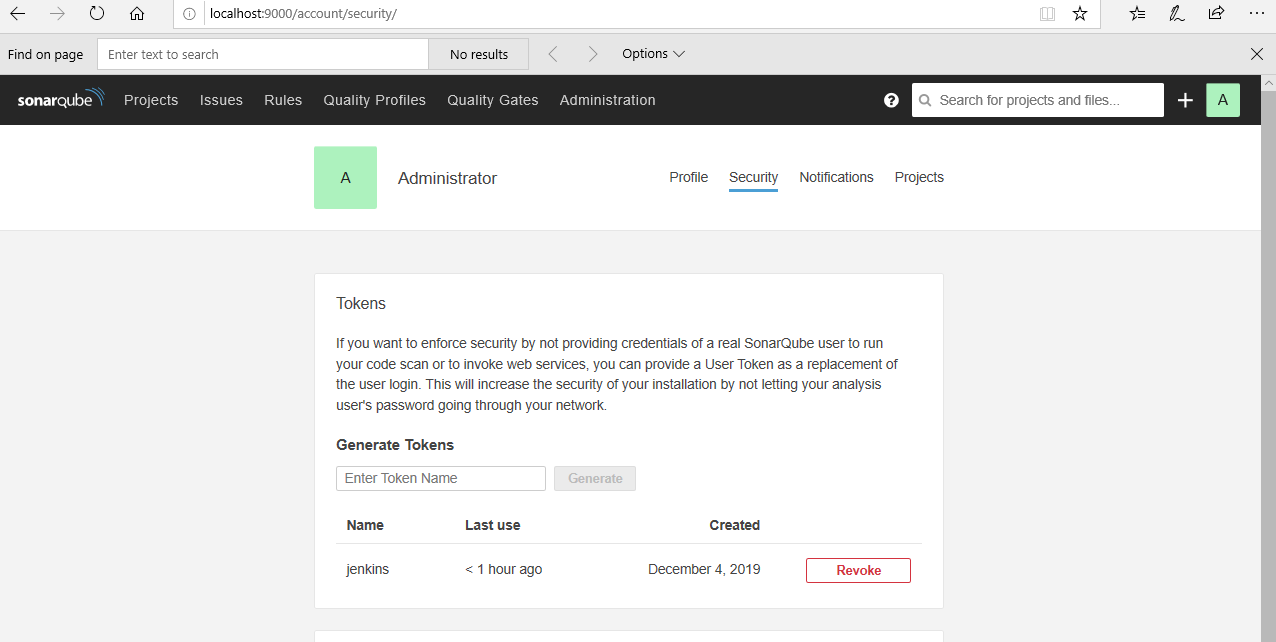
Here we can see that the total representation of code in way of bugs, vulnerabilities, code smells, coverage, duplications.

This is how we use sonar testing of code by manually but now integrating Jenkins with SonarQube comes into the picture.

We here had two ways of testing in Jenkins by using maven to test the code and the other way is to just use sonar scanner.

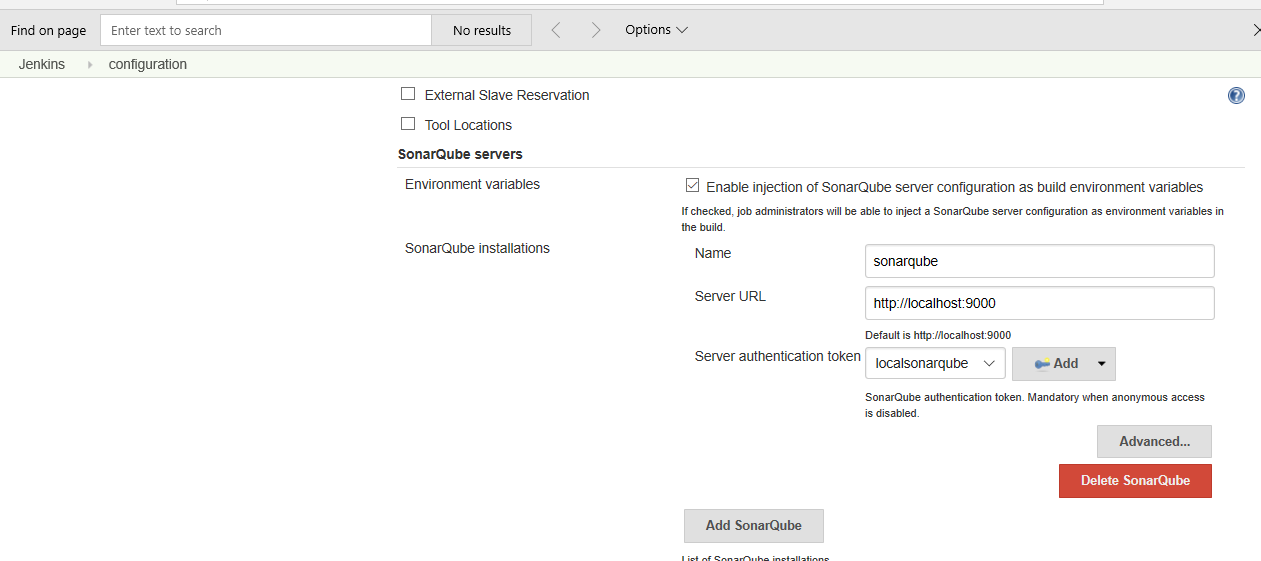
**Let’s see how to use the maven sonar in Jenkins now:**

First let’s configure sonar server in the Jenkins for this we need to generate a token in the server of SonarQube.



I have generated token by going into administrator->security->enter token name (Jenkins) ->generate

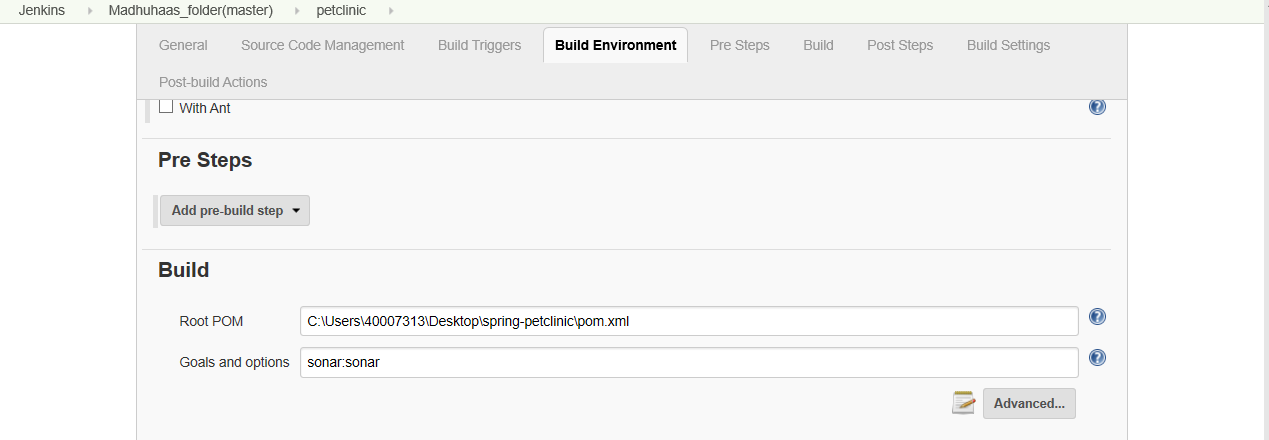
Now I have copied the generated token and used as a credential as localsonarqube in my Jenkins



Here in Jenkins go to manage Jenkins->system configuration->SonarQube servers

And here add server URL which in my case is a local host with 9000 port.

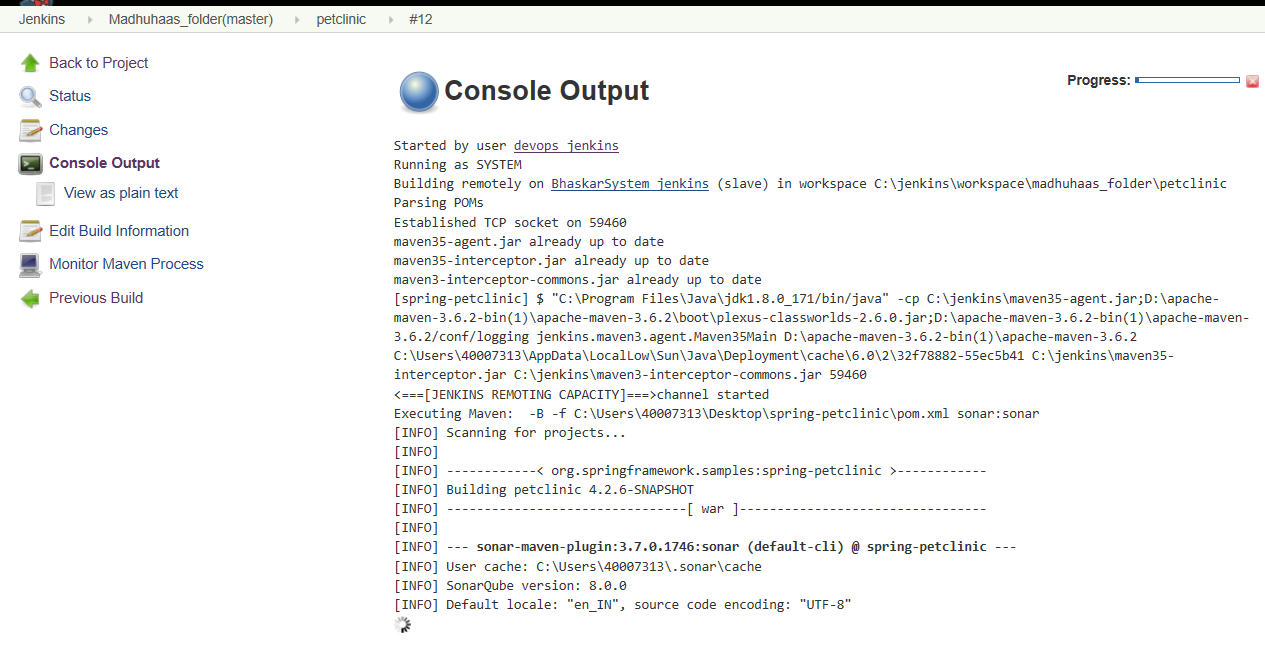
Now go to job creation in the Jenkins

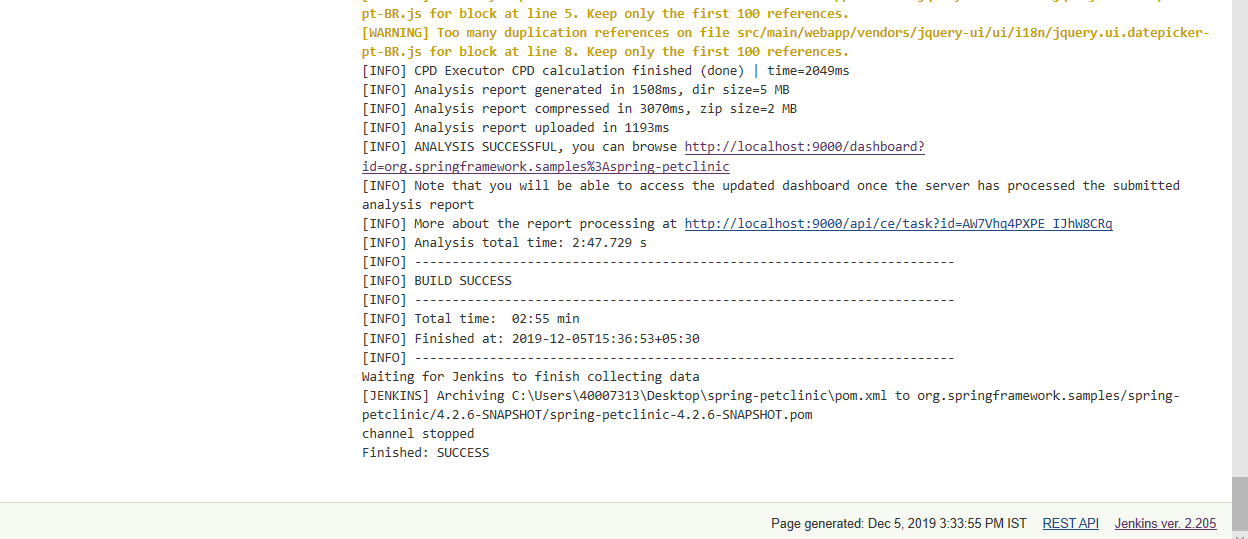


You can see above that I have created a job, and, in the build, menu gave the pom.xml file path and had given goals as “sonar: sonar”

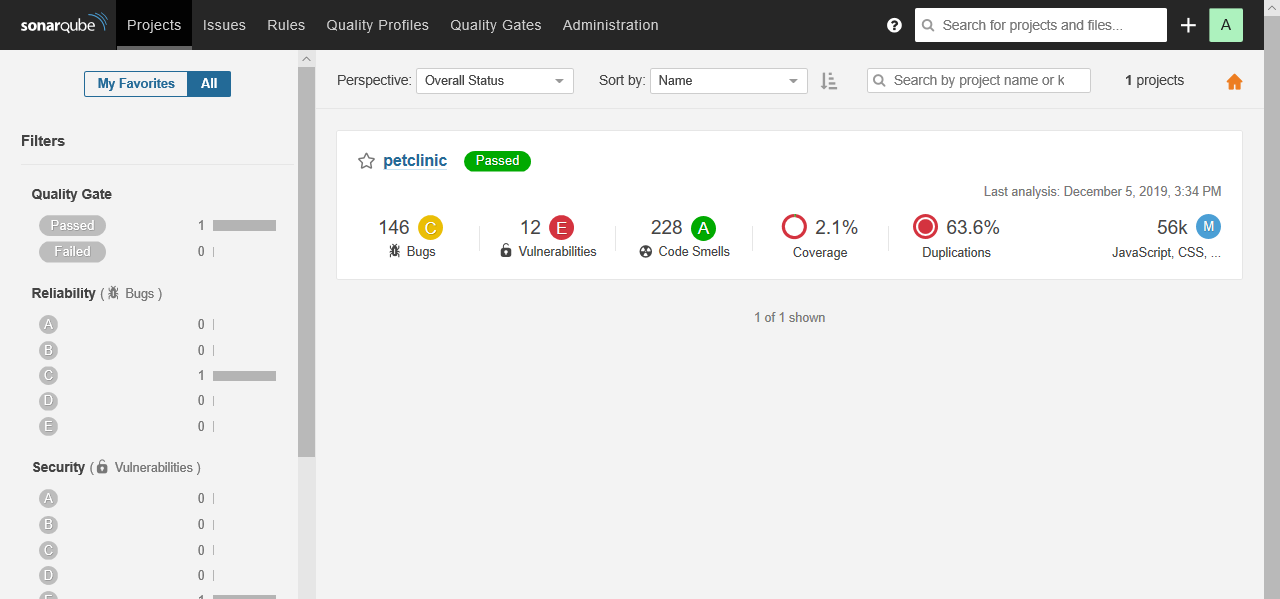
Click on save and start the job. And remember that I haven’t mentioned anything except the two settings in the job now let’s see the console output.

Let’s try starting this job by clinking on build now.





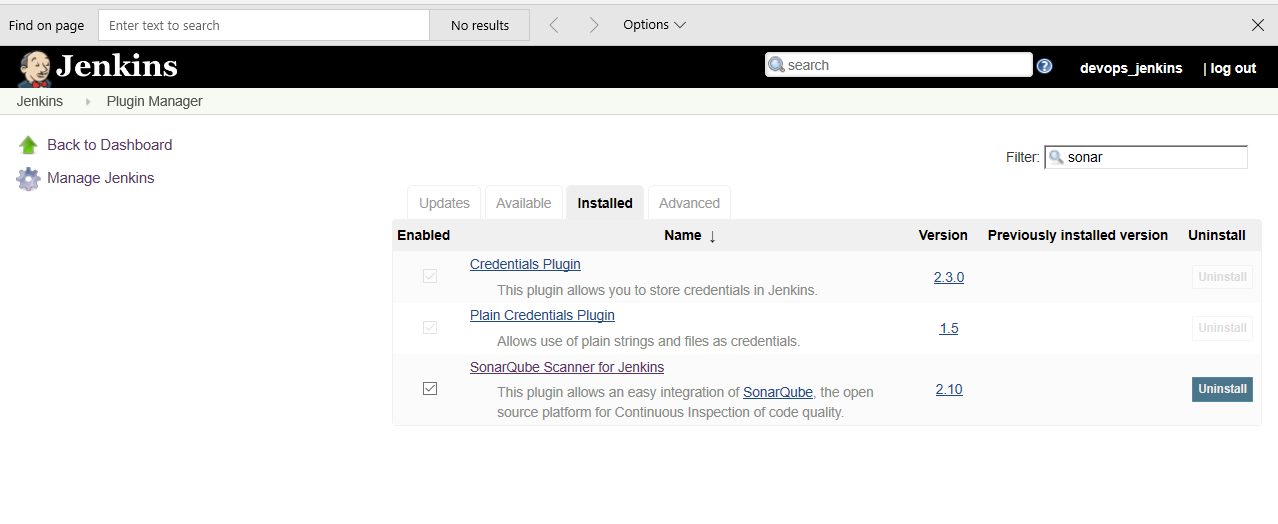
now above image shows that the build is success and pushed the analysis report to the dashboard of the SonarQube server. You can see the time in both the images.



**Now the sonar scanner is used to execute the tests in Jenkins.**

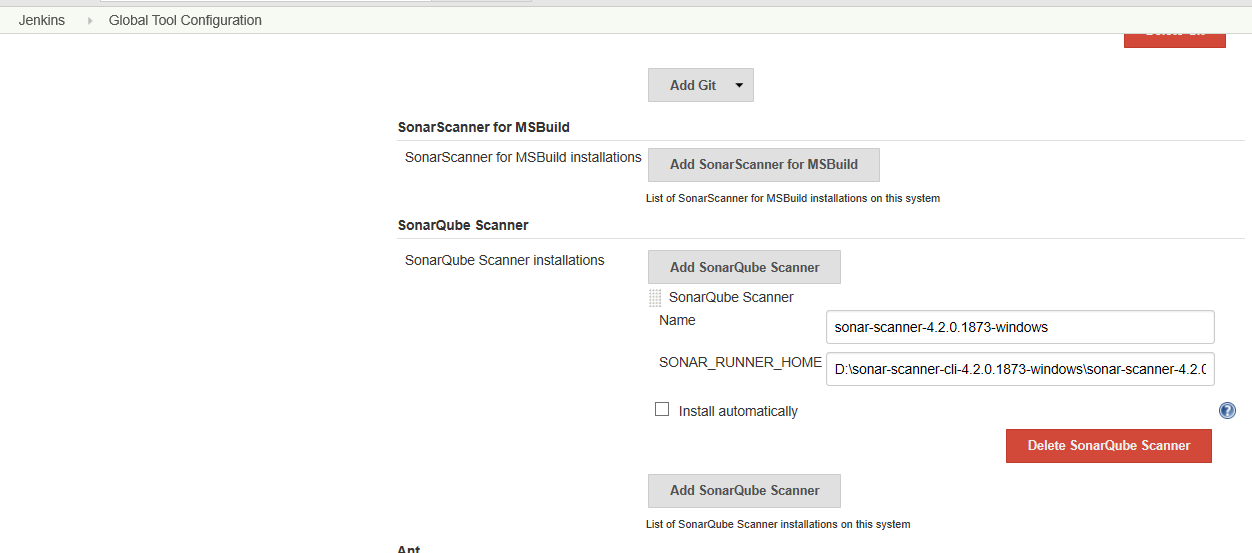
For this we need the sonar scanner downloaded in the local system

And install the plugin named: SonarQube scanner for Jenkins in plugin manager

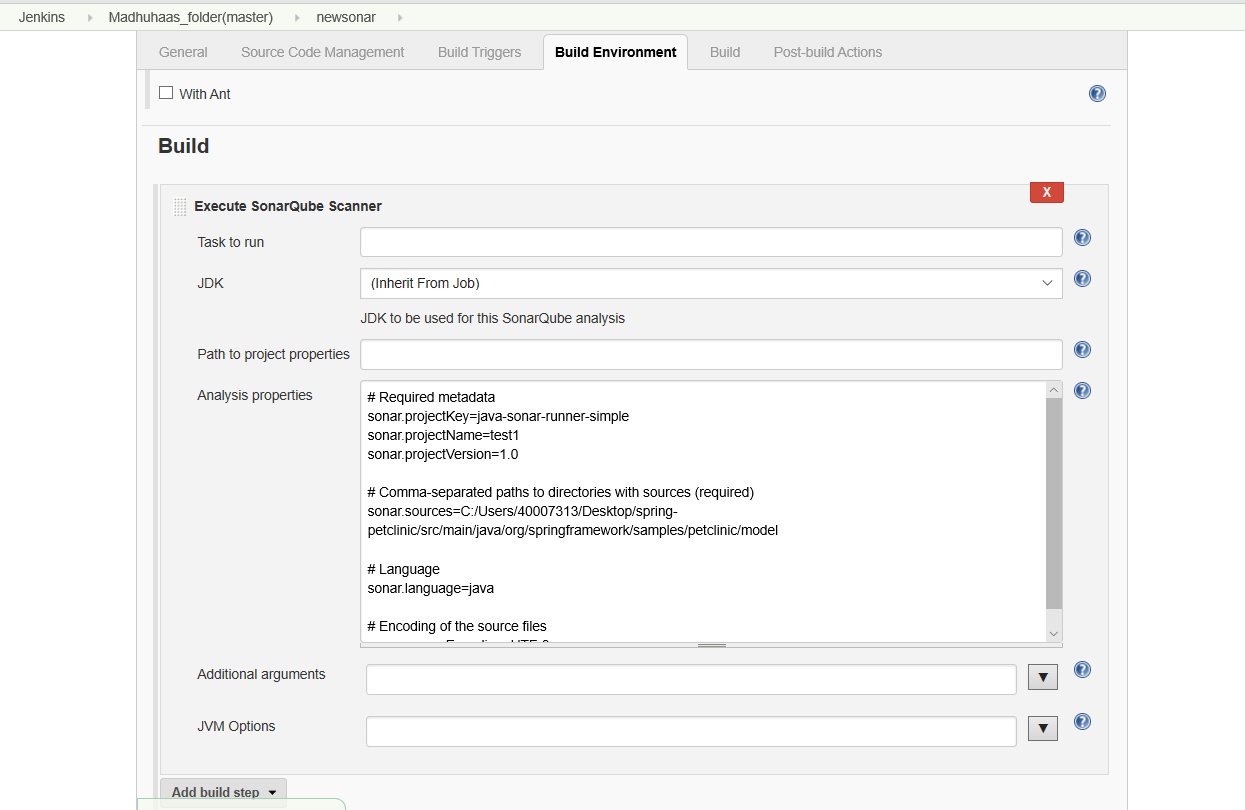


Configuring sonar scanner in the Jenkins:

Go to manage Jenkins-> global tool configuration->SonarQube scanner here give the correct path to the scanner.



now we try creating a job in which we will be using sonar scanner instead of normal maven command sonar: sonar :



I created a job named new sonar in that the settings for build will be:

Go to build and select execute SonarQube scanner and give the analysis properties as:

*# Required metadata*

*sonar.projectKey=java-sonar-runner-simple*

*sonar.projectName=test1*

*sonar.projectVersion=1.0*

*# Comma-separated paths to directories with sources (required)*

*sonar.sources=C:/Users/40007313/Desktop/spring-petclinic/src/main/java/org/springframework/samples/petclinic/model*

*# Language*

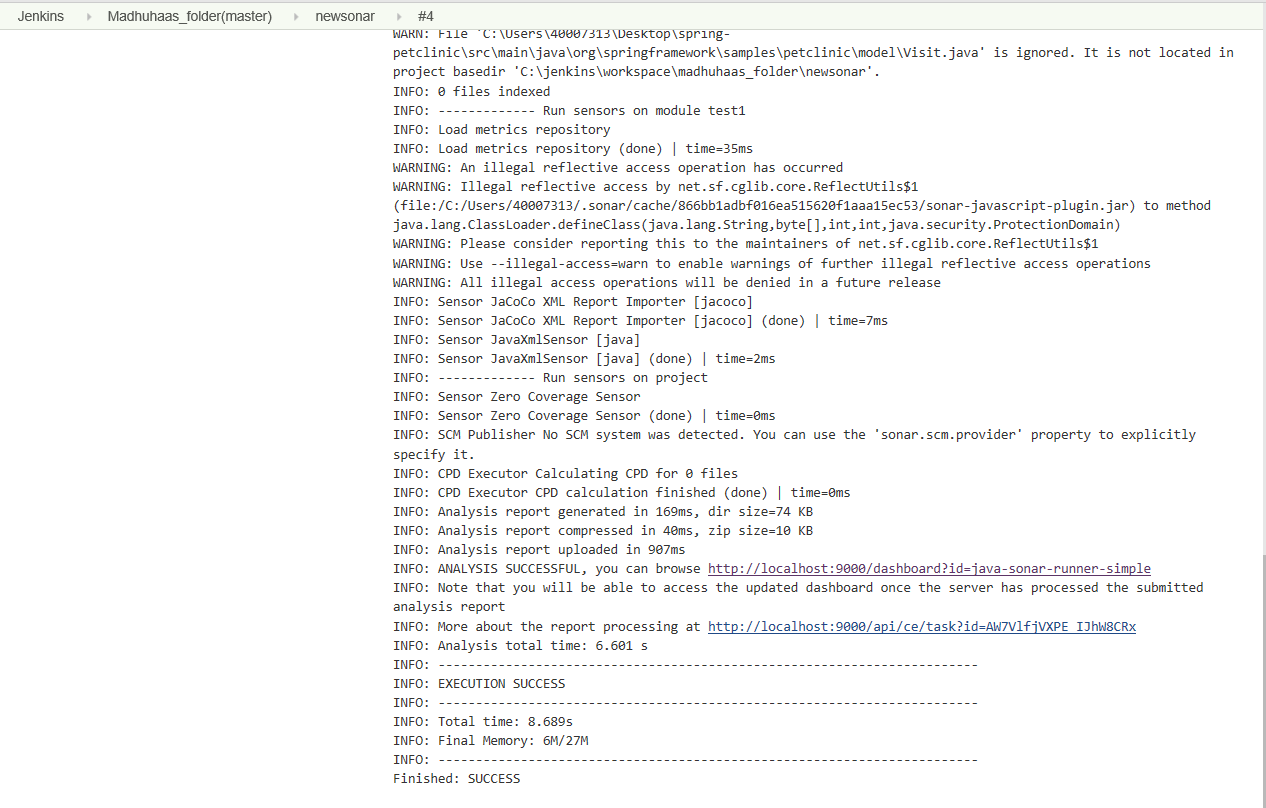
*sonar.language=java*

*# Encoding of the source files*

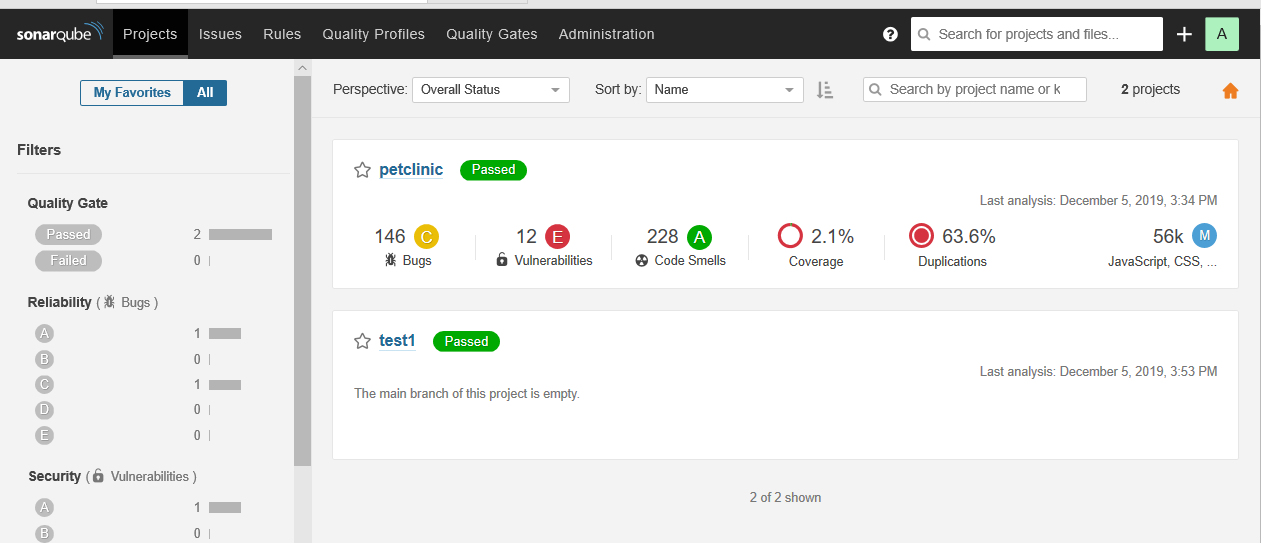
*sonar.sourceEncoding=UTF-8*

here in sonar.sources path I have mentioned the path of the src from where the files are available to be processed in the sonar scanner and no other settings were selected instead of this.

Now we will try executing the job in Jenkins:



It says success and pushed to the server. Let’s check the server for the project.



And here the test 1 is the project name in properties analysis that we tried executing with the SonarQube in Jenkins and we were successful here.

THANK YOU