Sonar Qube

Code quality checking tool

SonarQube is an open-source tool/platform for inspection of code quality and security

Basically using this tool, we can identify the mistakes or issues in the code. As a DevOps engineer, we identify the bugs and we give them to Developer and they will fix them

--> It is developed by using Java language

--> SonarQube supports most of the programming languages to perform Code review (30+ programming languages definitely will support).

--> With the help of this platform, we can generate Code review report. That report can be given to Developers to fix the code.

Note: Code review is a part of Project Build Process

Sonar issues

--> SonarQube server will identify the following issues in the report

--> Bugs

--> Vulnerabilities (Security hotspots)

--> Duplicate code blocks (repeated code)

--> Code smells (Weak design in application)

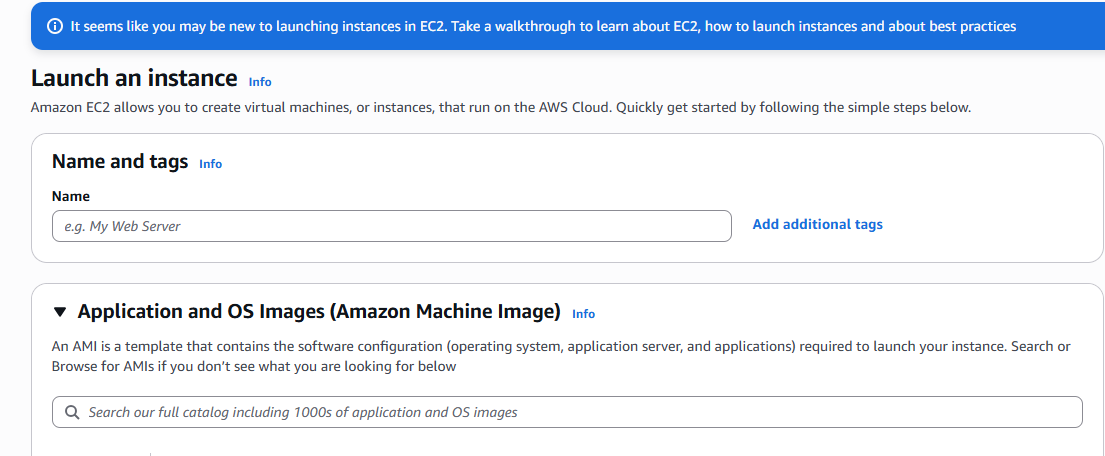
--> Code coverage (How many lines of code is tested in Unit testing)

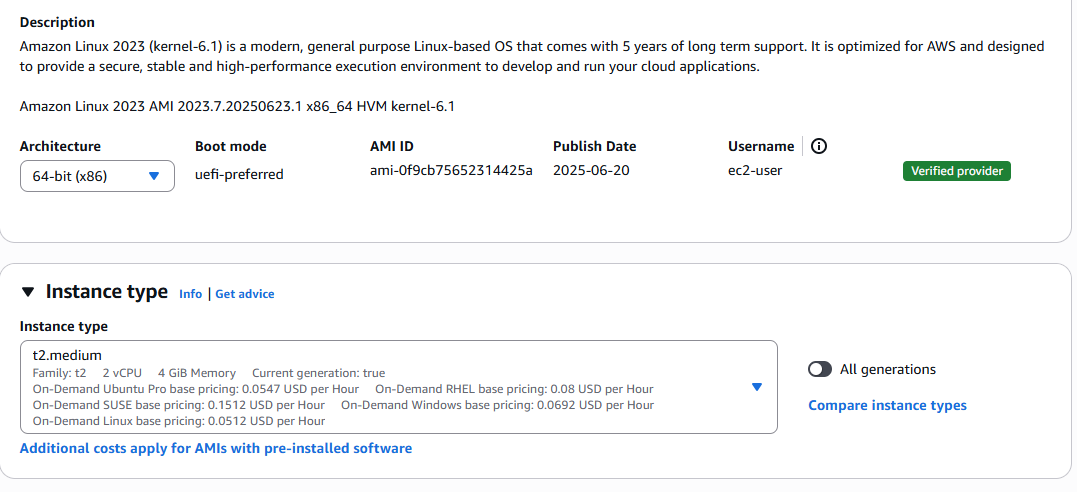
We can add these things in Sonar Quality Profiles

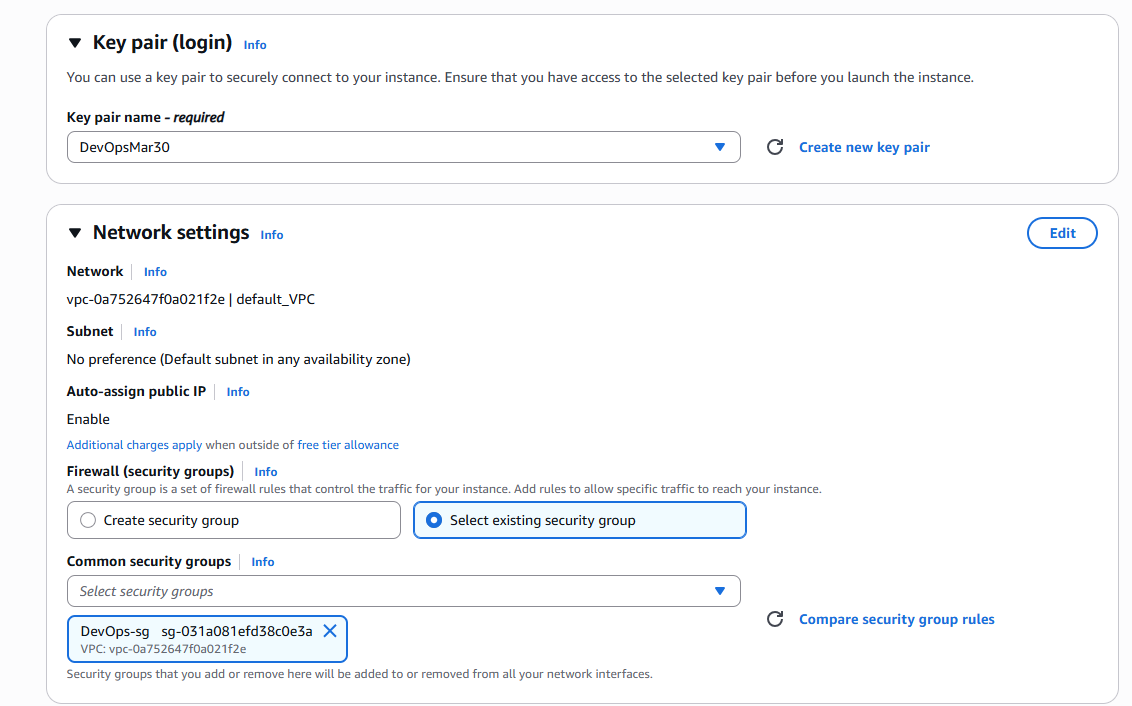
-> Set of rules to be considered during code review

1:07

Launch a new EC2 instance







Launch Instance

[ec2-user@ip-172-31-29-86 ~]$ sudo yum update -y

Amazon Linux 2023 Kernel Livepatch repository 160 kB/s | 17 kB 00:00

Dependencies resolved.

Nothing to do.

Complete!

[ec2-user@ip-172-31-29-86 ~]$ sudo yum install docker -y

[ec2-user@ip-172-31-29-86 ~]$ sudo service docker start

[ec2-user@ip-172-31-29-86 ~]$ sudo service docker start

Redirecting to /bin/systemctl start docker service

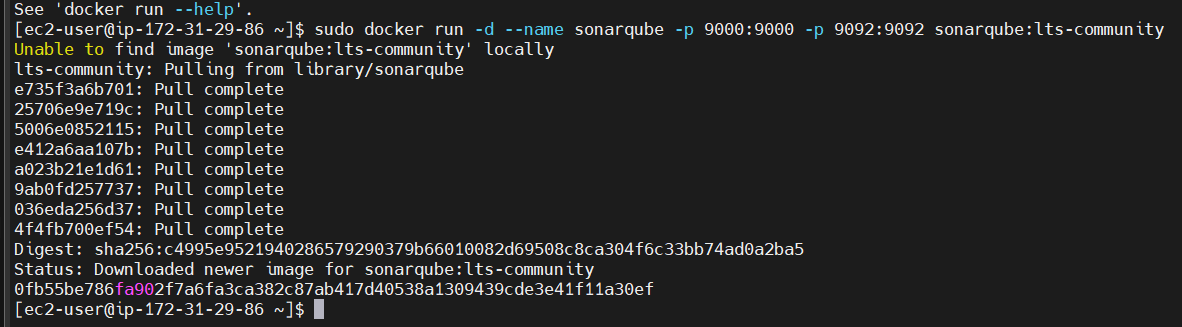
[ec2-user@ip-172-31-29-86 ~]$ sudo usermod -aG docker ec2-user

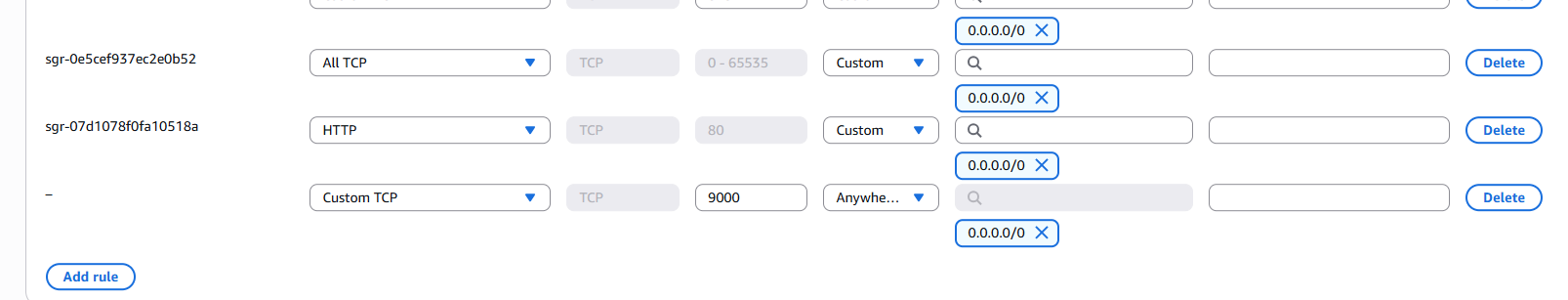
[ec2-user@ip-172-31-29-86 ~]$ docker -v

Docker version 25.0.8, build 0bab007

[ec2-user@ip-172-31-29-86 ~]$ docker run -d --name sonarqube -p 9000:9000 -p 9092:9092 sonarqube:lts-community

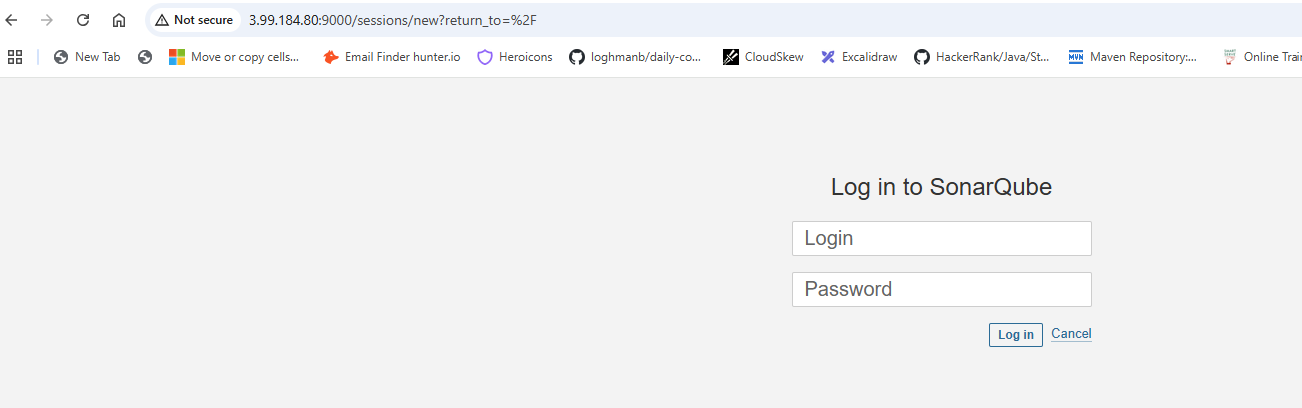
[ec2-user@ip-172-31-29-86 ~]$ sudo docker run -d --name sonarqube -p 9000:9000 -p 9092:9092 sonarqube:lts-community



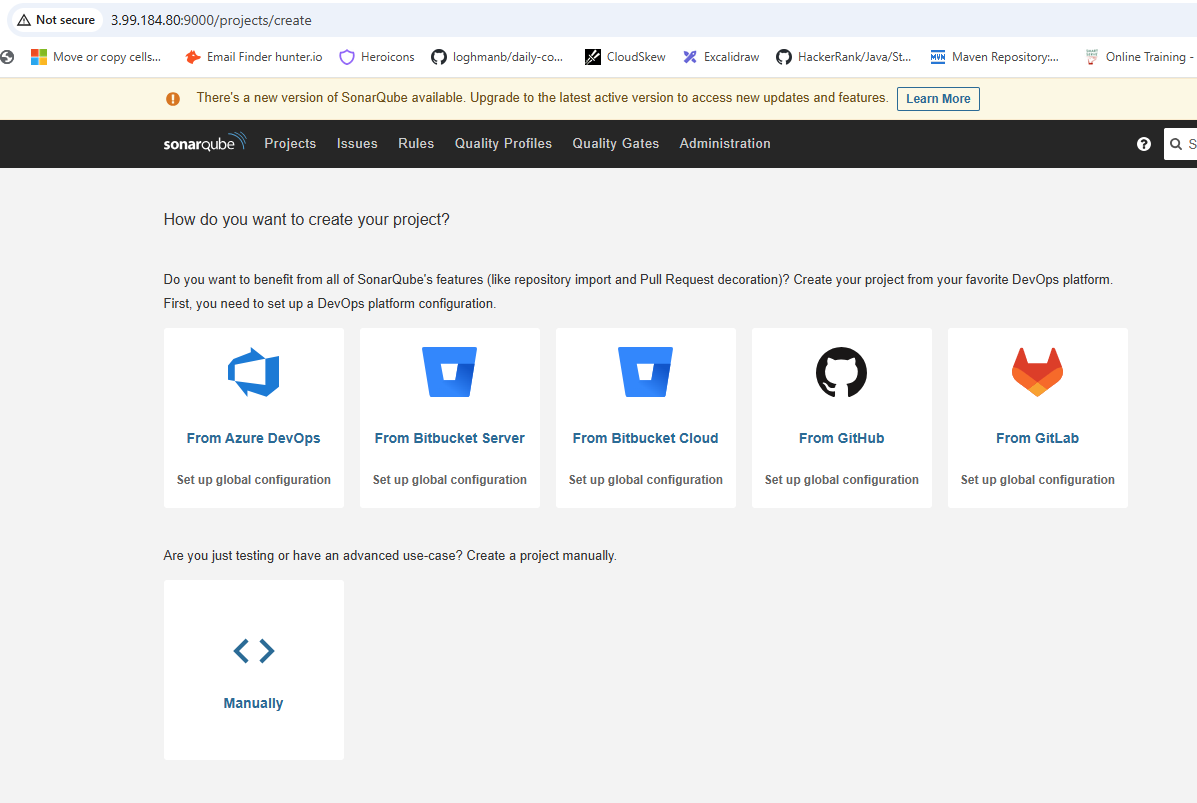


Open: <http://3.99.184.80:9000/sessions/new?return_to=%2F>

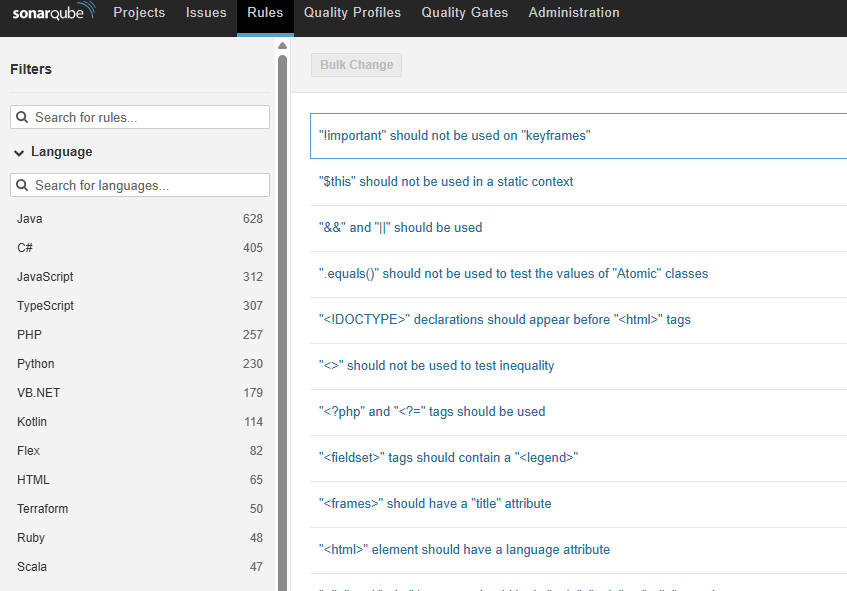
SonarQube is running on port 9000

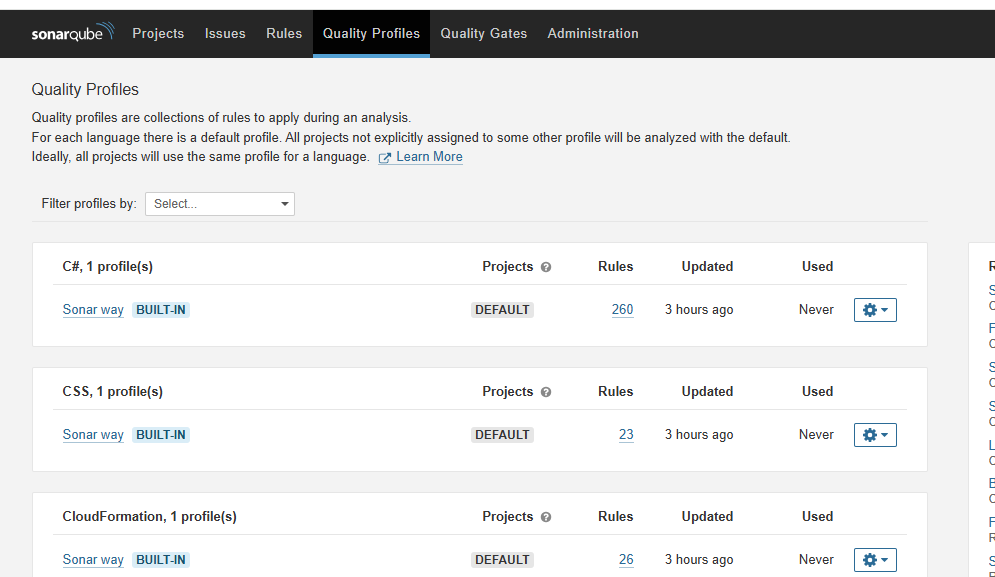


Default username and password is admin, admin

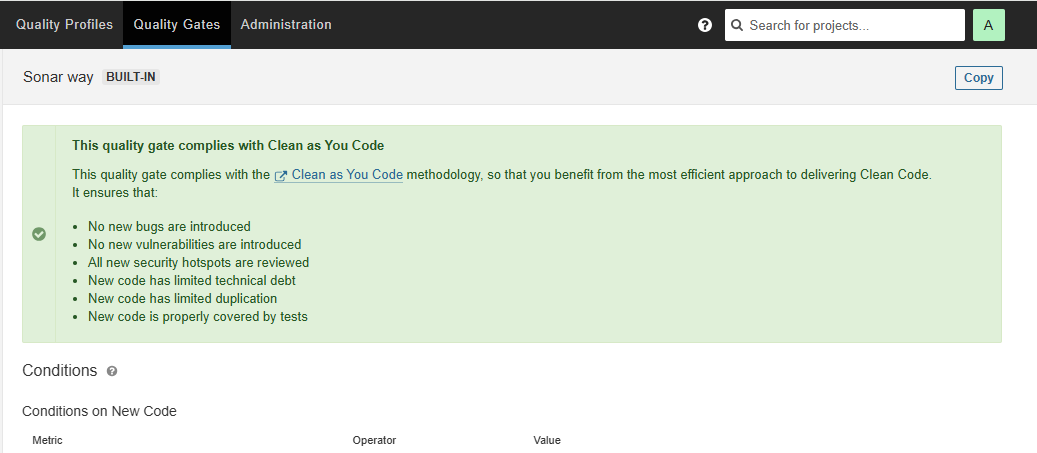


<http://3.99.184.80:9000/projects/create>





If you get Quality Gate passed, it is safe to deploy the application

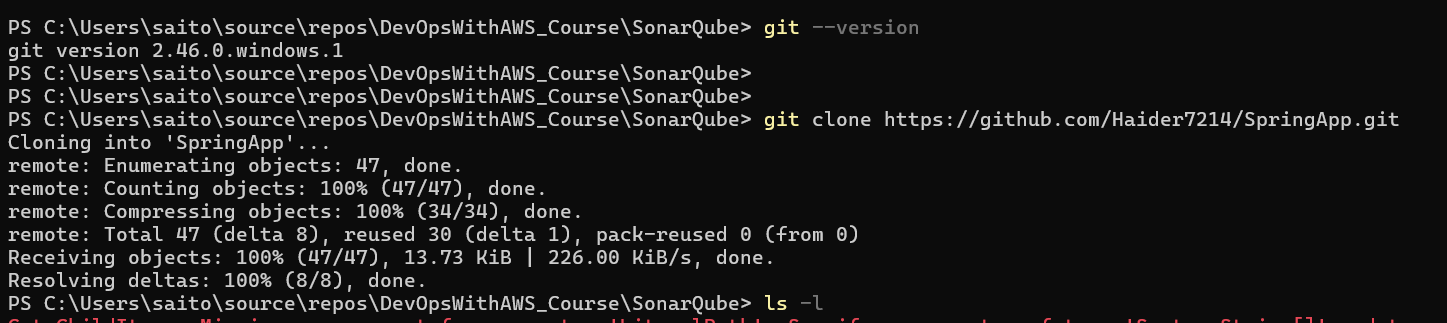


Go to Git

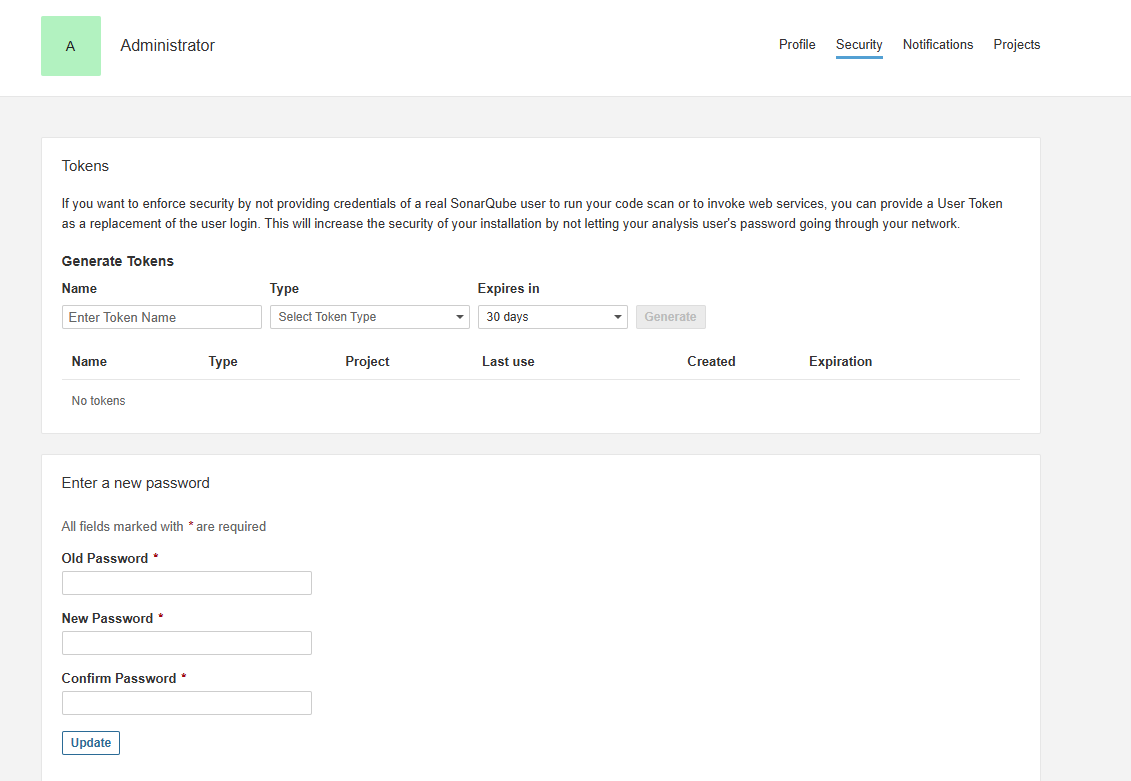
<https://github.com/Haider7214/SpringApp/blob/main/pom.xml>

We can replace the IP with my SonarQube public IP



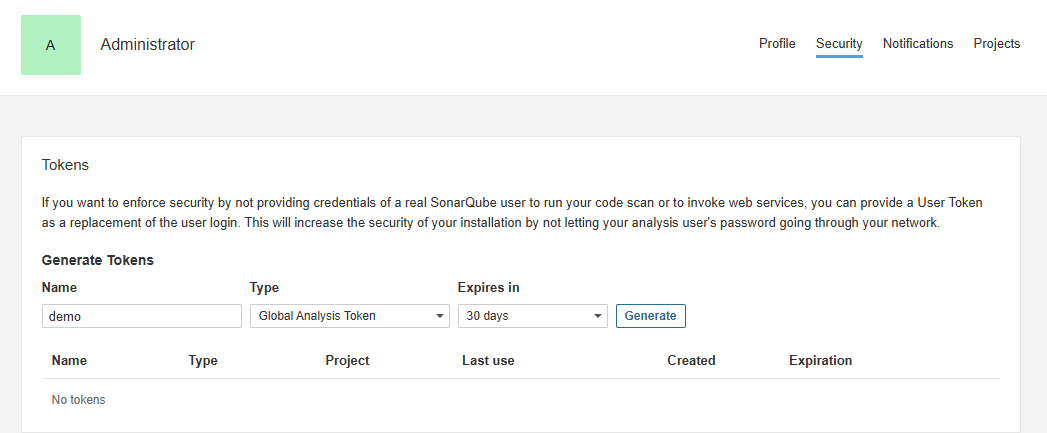


Go to SonarQube “My Account” --> Security



Global Analysis Token

Click Generate

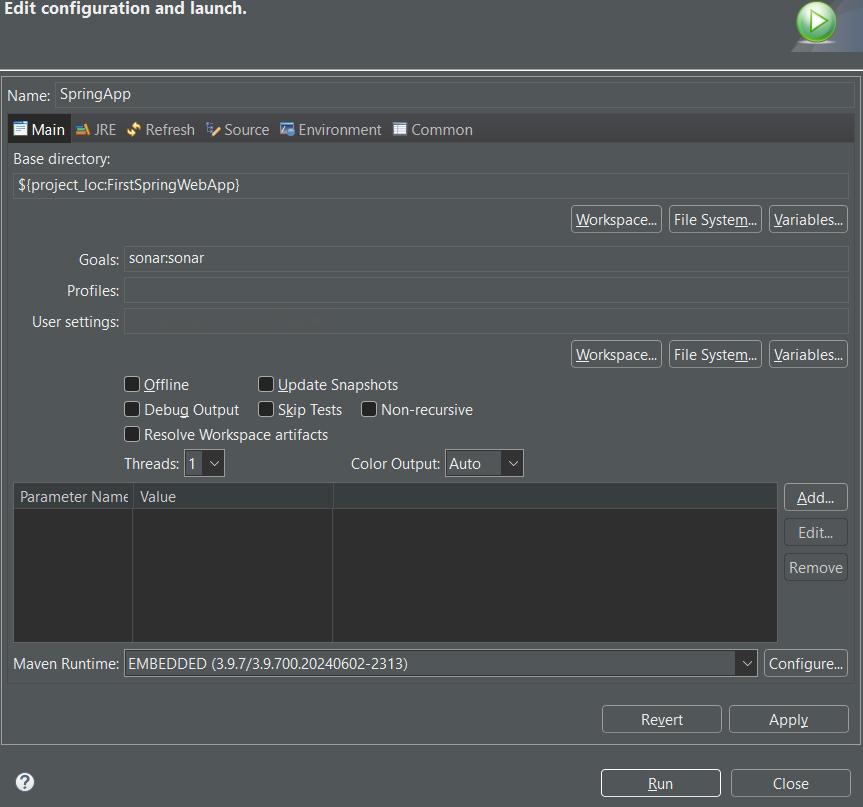


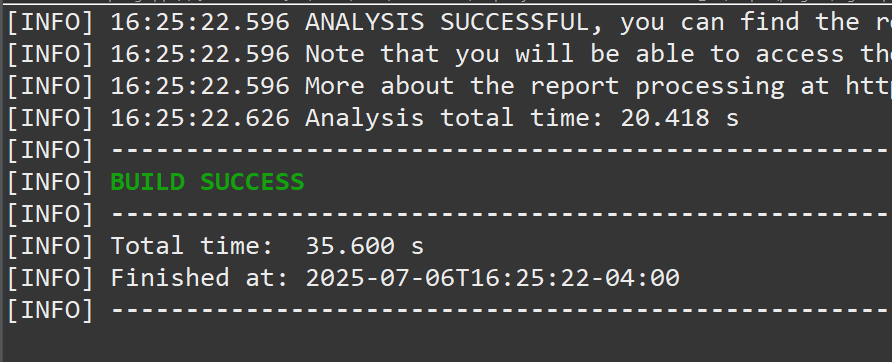
Token: sqa\_89db2d71077985fb78fa1d176ba4abae93a98dd2

Open Pom file and enter Token and Public IP

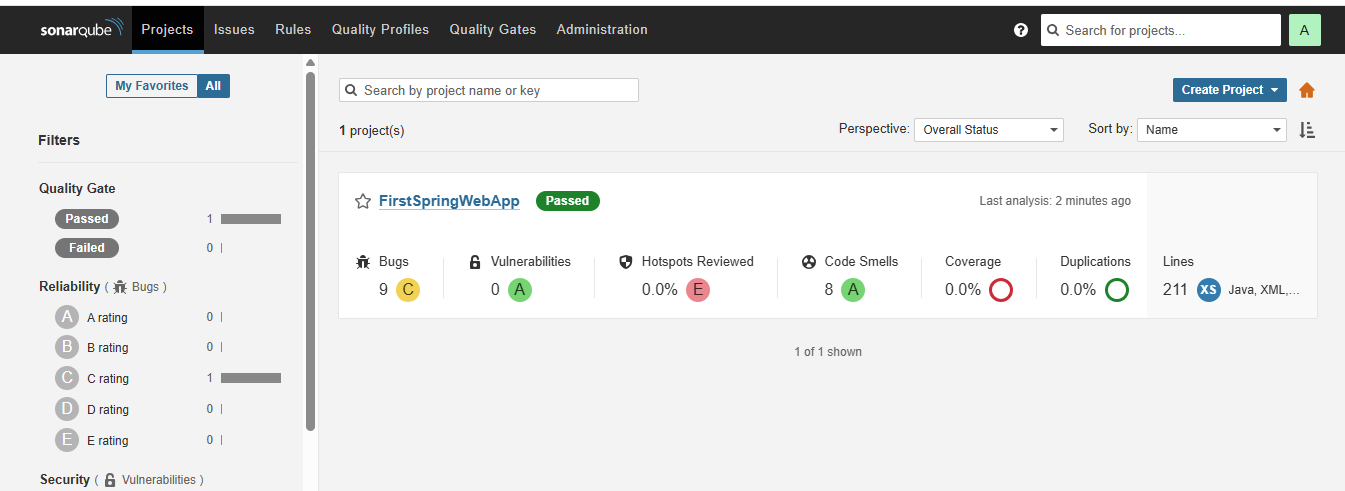


Open project in Eclipse --> right click --> Run As --> Maven Build --> Goals: sonar:sonar

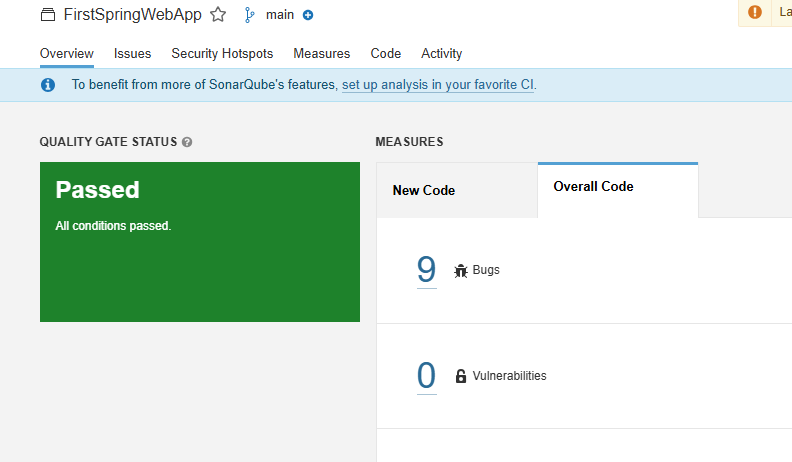




Go back to SonarQube --> Projects

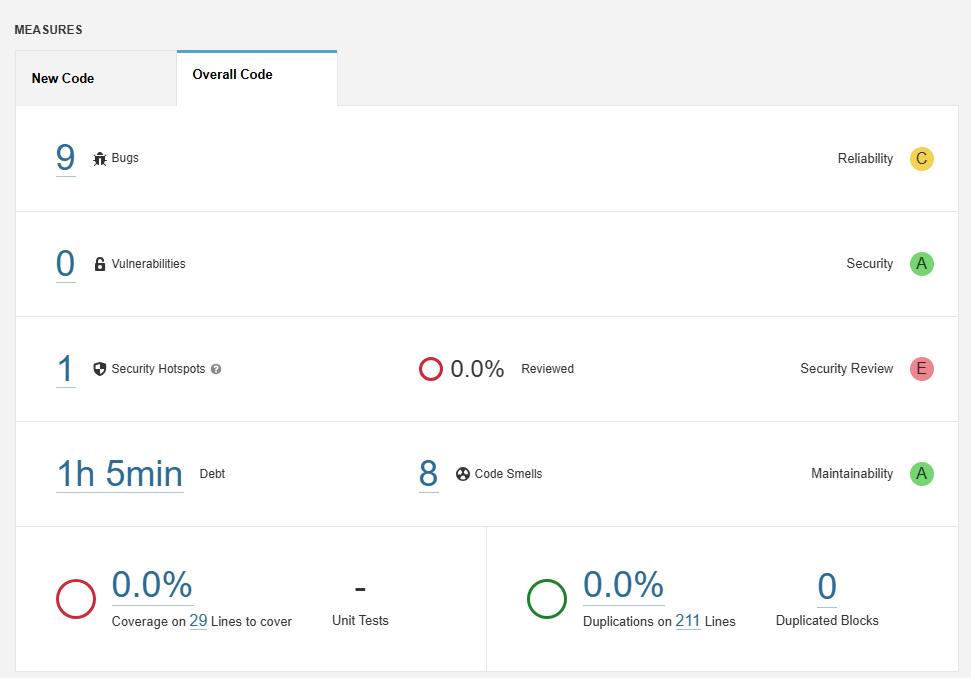


FirstSpringWebApp project is Passed

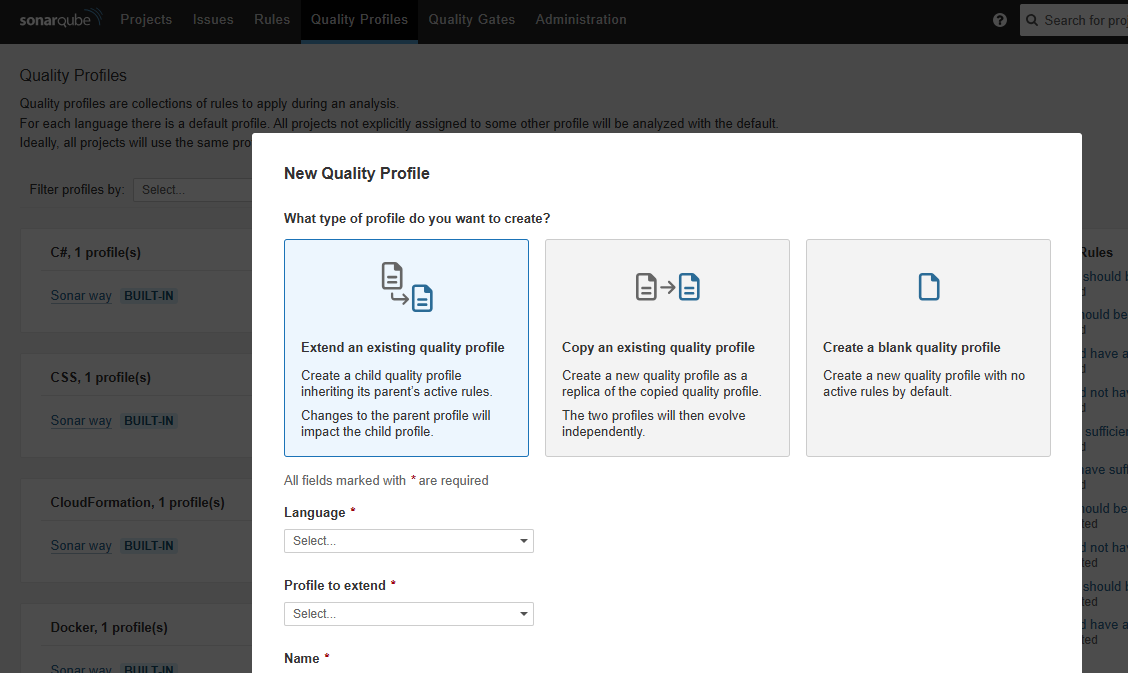


9 bugs are there

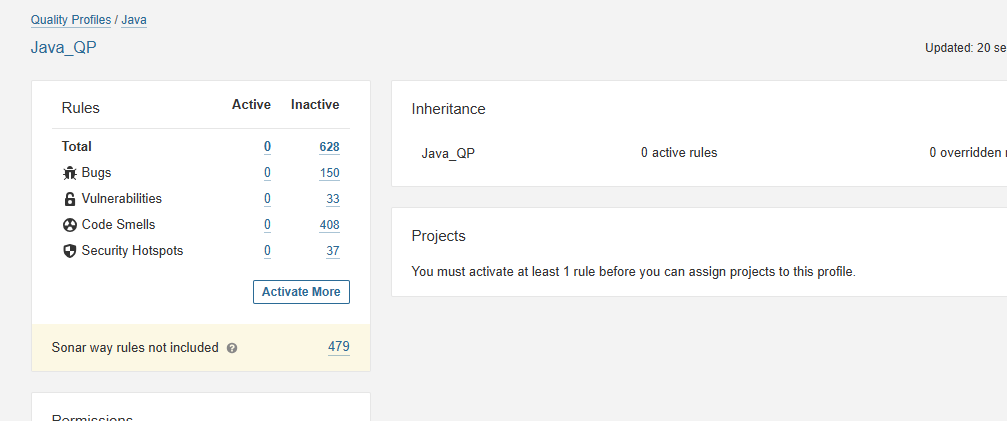
It will take Developers 1h 5min to resolve the issues

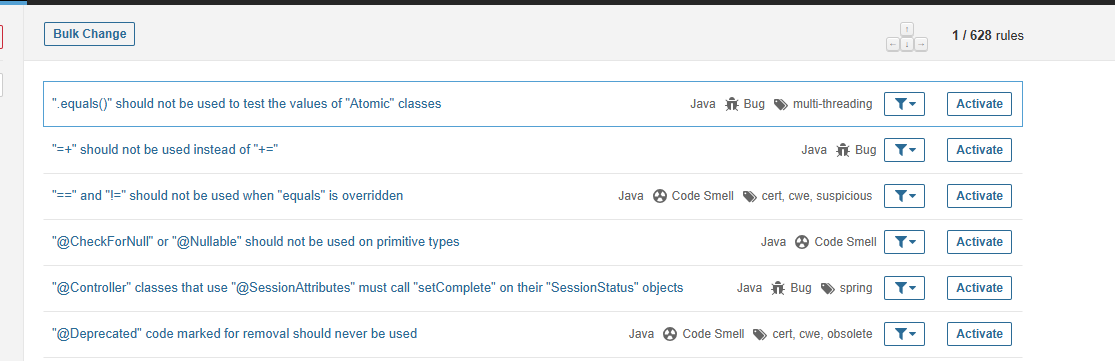


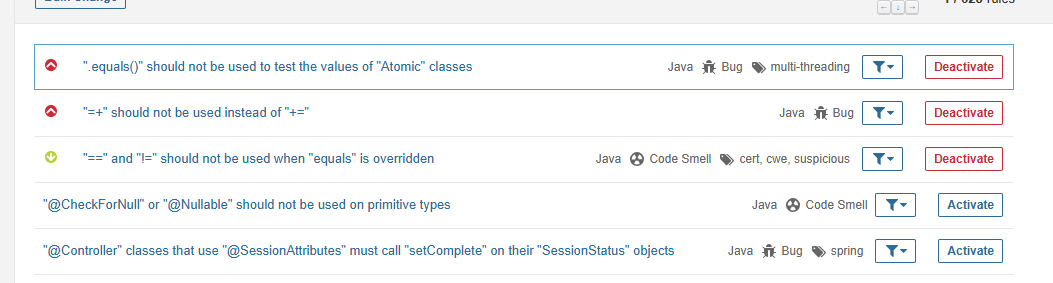
For Quality Profile

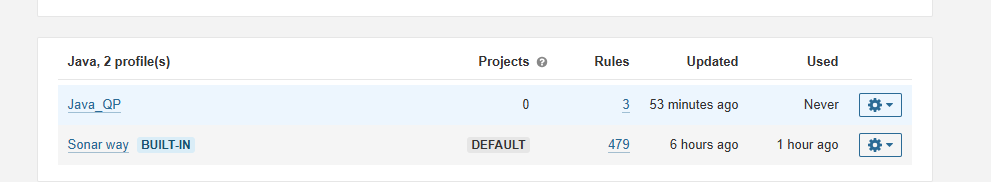


Click on Activate More



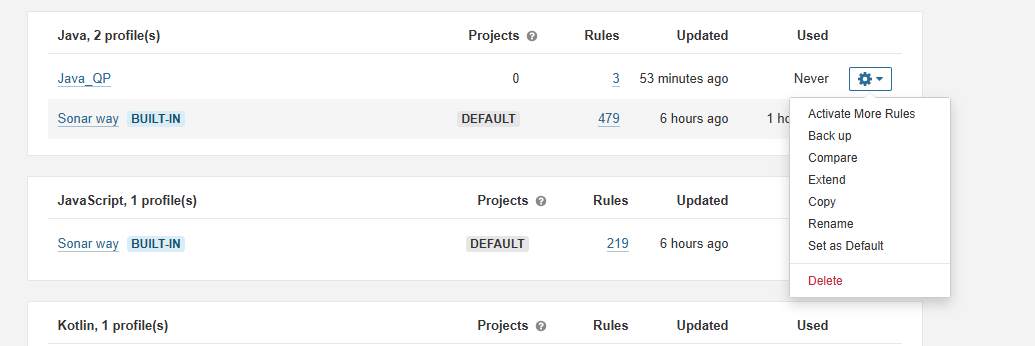




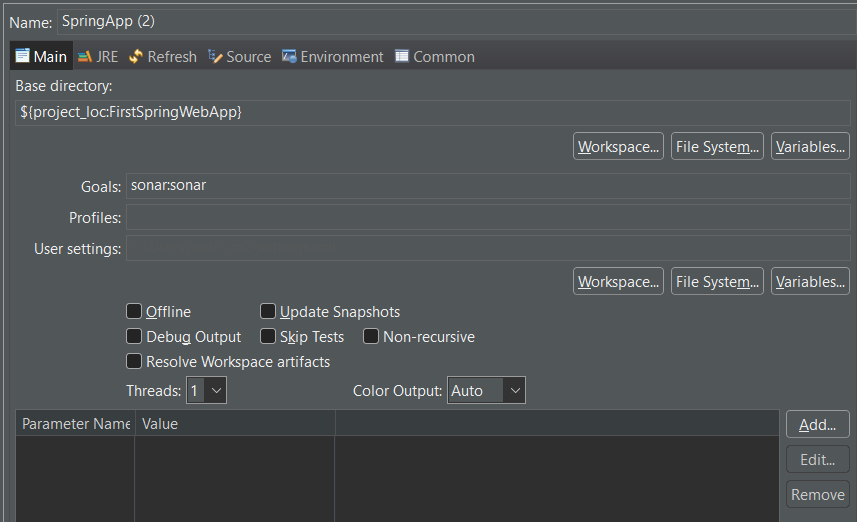


2 Java profiles are there, default has 479 rules

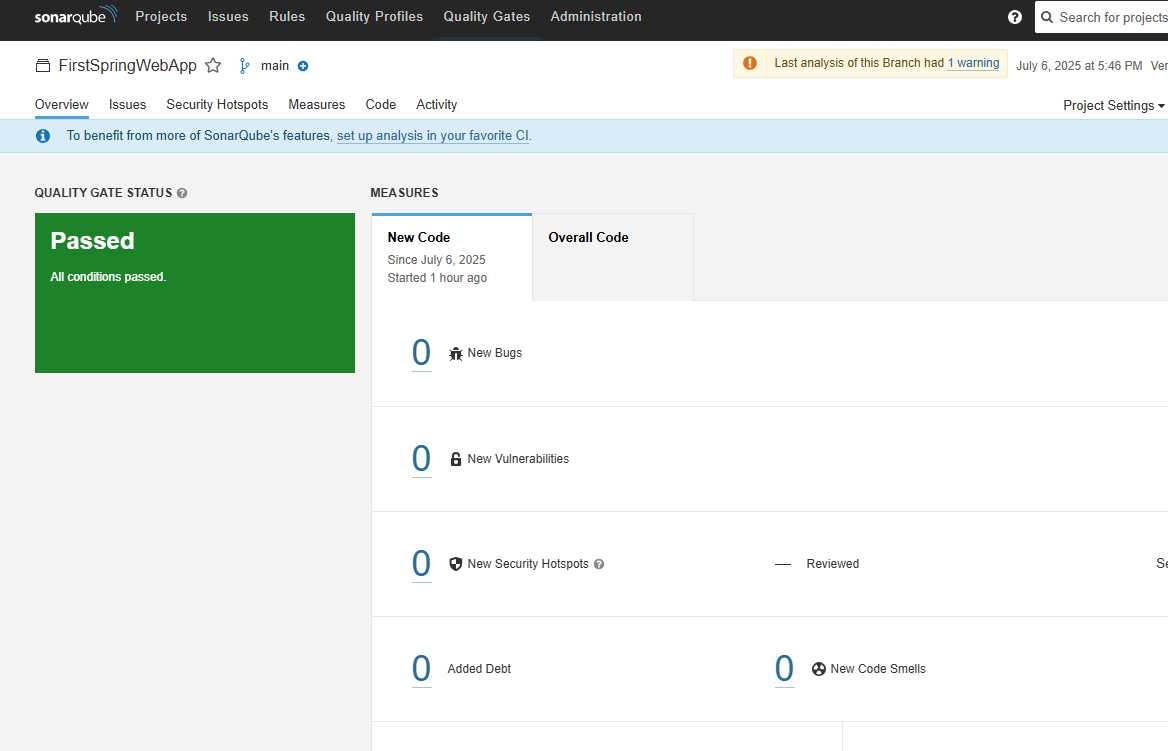
Set my rules as default



Maven Build once again in Eclipse

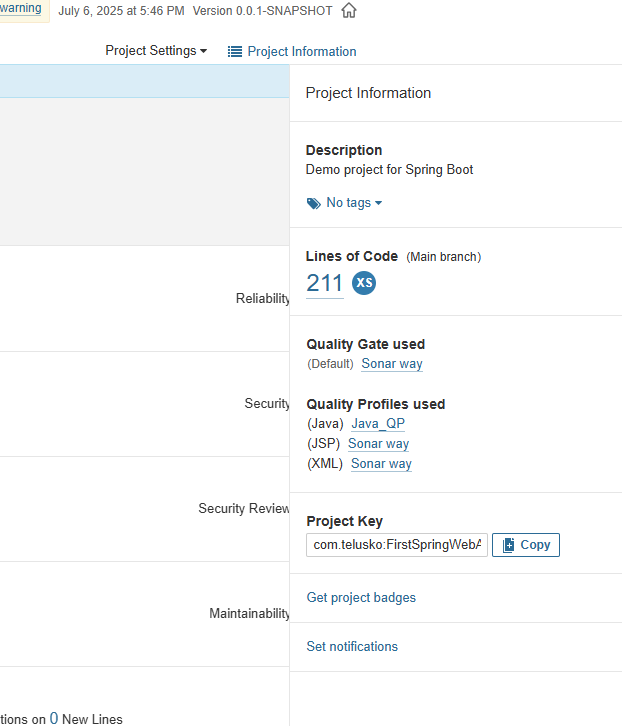


Now we can see 0 bugs based on my 8 rules

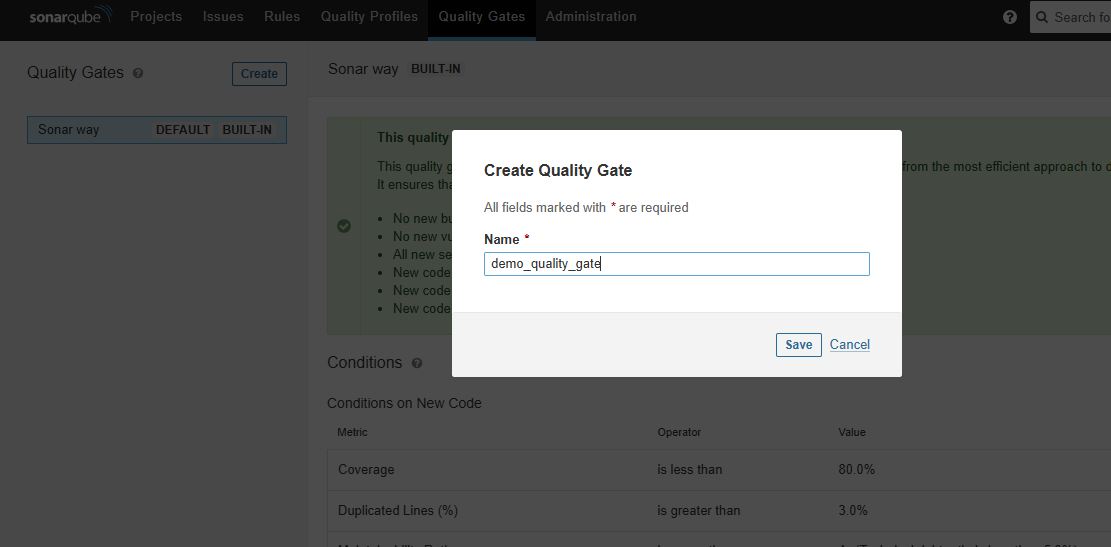


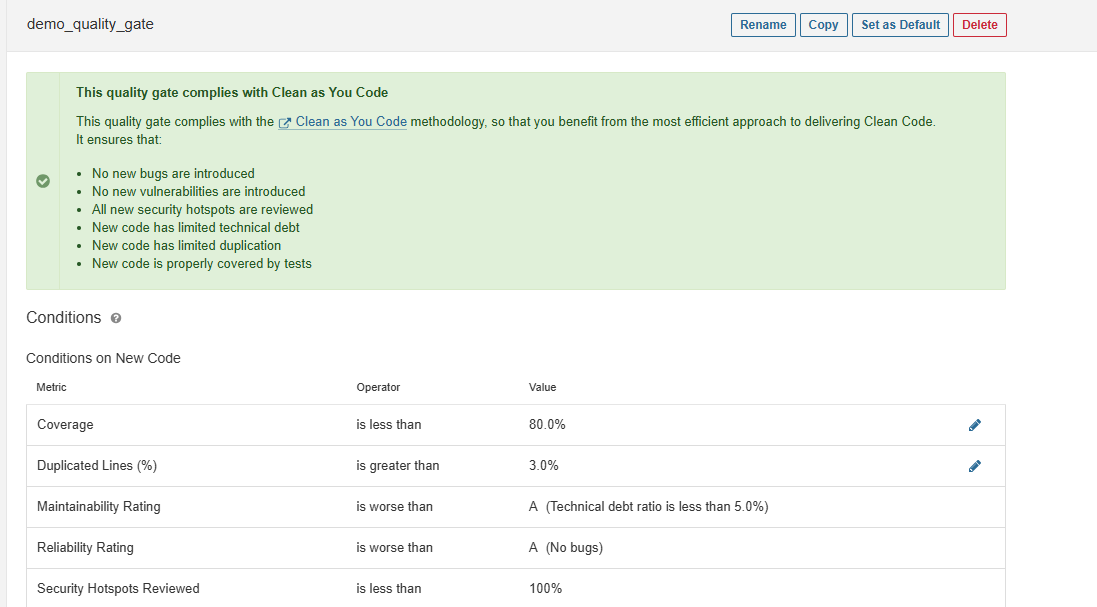
Now 0 bugs

Project Information



211 lines are there and Java\_QP is being used





We can change the Conditions as well

SonarQube setup

1 --> Setup a Linux VM (Amazon Linux AMI --> t2.medium) min 2 GB RAM is required

Connect to that VM once created

2 --> Install Docker in Linux VM

sudo yum update -y

sudo yum install docker -y

sudo service docker start

sudo usermod -aG docker ec2-user

exit

Reconnect back to Linux VM

3 --> Verify the Docker installation

docker -v

4 --> Run SonarQube using Docker image of SonarQube

docker run -d --name sonarqube -p 9000:9000 -p 9092:9092 sonarqube:lts-community

Enable 9000 port number in Linux VM security group

5 --> Access SonarQube public IP of Linux VM:port number (9000)

After accessing default username and password : admin and admin

Change it to your own

Integrate Sonar Server with Java Maven project

1 --> Generated Sonar token for integration

Sonar Dashboard --> My Account --> Security --> Generate Token

2 --> Install Docker in Linux VM <https://github.com/Haider7214/SpringApp/tree/main>

pom.xml under properties

<properties>

<java.version>17</java.version>

<sonar.host.url>http://3.99.184.80:9000/</sonar.host.url>

<sonar.login>sqa\_89db2d71077985fb78fa1d176ba4abae93a98dd2</sonar.login>

</properties>

3 --> mvn clean package --> Open Git bash from a folder where Project is cloned

--> mvn sonar:sonar --> Goal will build project

Sonar Quality Profiles

--> Set of rules to perform code review

--> For every programming language one Quality profile with set of rules to perform Code review would be available (default/built in)

--> Java project --> It has its own Java quality profile --> Java set of rules to perform code review

--> Whenever we perform code review using Sonarqube it will identify our project developed using which language based on that it will execute the language specific quality profile and perform code review

--> We can create our own Quality Profile based on project requirement

- Name: Demo\_App\_QP

- Language: Java

- Parent: None

We can make our quality profile as default profile then it would be applicable for all projects which get reviews under this Sonar server

Sonar Quality Gate

--> Quality Gate represents the overall project code quality whether it is passed or failed, which will decide if project can be deployed or not

Note: If code quality is failed, we should not deploy that particular code or application

In Sonar, we will have default Quality Gate and every project must be passed in order to deploy

If required, we can create our own quality gate as per need or requirement

Summary Note:

--> If project quality gate is failed then we must not accept that code for deployment as a DevOps engineer

--> if we found out any Sonar issues then Development team is responsible to fix all issues and as a DevOps engineer we will perform code review and send code review report to development team

SonarQube --> Sonar setup on Linux VM --> Sonar integration in Maven project --> Sonar token generation --> Server issue types, Quality profiles --> Quality gates