SonarCloud integration in Jenkins

In this page, we will describe How to create SonarCloud Account and its configuration with Jenkins

Objectives

Realize Project using **SonarCloud, Jenkins, Maven and GitHub**

Project Context :

- **Alain Pierre** is the **manager** of a team of **7 developers** of an **IT solution** development company called **e-Max Soft (a Cameroonian firm of high end software development)**.
- He has just set up a project of a **Software for geolocation of patients for their assistance by doctors.**
- His team, being only experienced in **DEV**, has created a **GitHub repository** for the codes of this project.
- These developers use the **Java language, HTML** an **Java Script** to write their code, as build tools, they use **maven** to **compile** their source code in an archaic way.
- **Alain Pierre** the manager of this very ambitious project **recruits** you in this team as a **DevOps engineer**.

With the notions learned at **Utrains**, your

1- first mission within this team is to set up the source code analysis for this project.

2- the second mission would be to migrate the database to the AWS cloud,

3- the third and last mission will be to integrate artifactory and then to implement the continuous deployment

I - First Mission : **set up the source code analysis for this project.**

The major steps to accomplish this first mission:

Step 1 : install and configure a **Jenkins server** for the project

Step 2 : integrate **Maven** in Jenkins : the build tool

Step 3 : create a repository in **GitHub** and then get the project code in this repository

Step 4 : create a maven project in Jenkins and link it to our **GitHub** project

Step 5 : create an **account in SonarCloud**

Step 6 : Integrate **SonarCloud** in **Jenkins** and perform the code analysis

Step 7 : Build and check the Result

Step 1 : install and configure Jenkins

as a reminder, we have our jenkins server installed with vagrant which contains the following characteristics :

- **IP Address : 192.168.56.17**
- **user : vagrant**
- **password : vagrant**

the process is as follows:

1- Open the terminal,

2- connect to the server with **ssh**

3- use the **yum** package manager to **install git**

```
ssh vagrant@192.168.56.177
```

You can go to this link if you want to review the implementation of this jenkins server :

[Set Up Jenkins Server]

Step 2 : integrate Maven in Jenkins

- this step is already done, see the link below for a reminder :

[Maven integration in Jenkins : confluence Link]

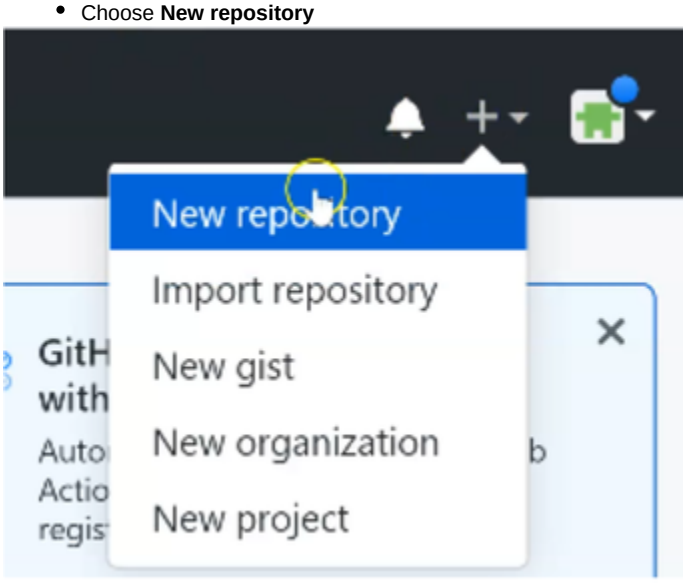
Step 3 : create a repository in GitHub and then get the project code in this repository

- Log in to your **GitHub** account by clicking in this bellow link

http://github.com

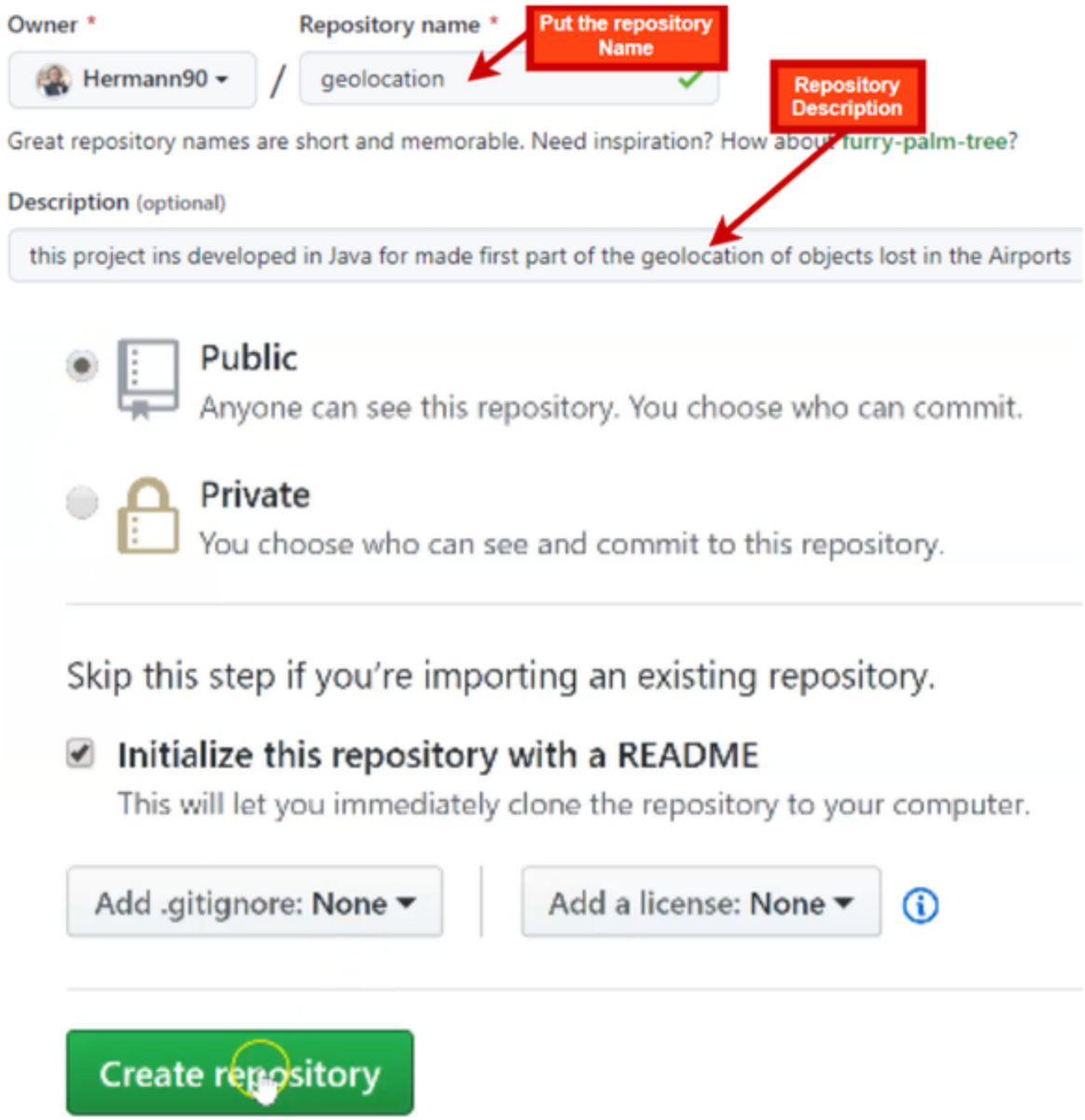
Let's create a repositories in **GitHub** named **geolocation**

- Click on the sign **+** to **create a new repository**
- Choose **New repository**



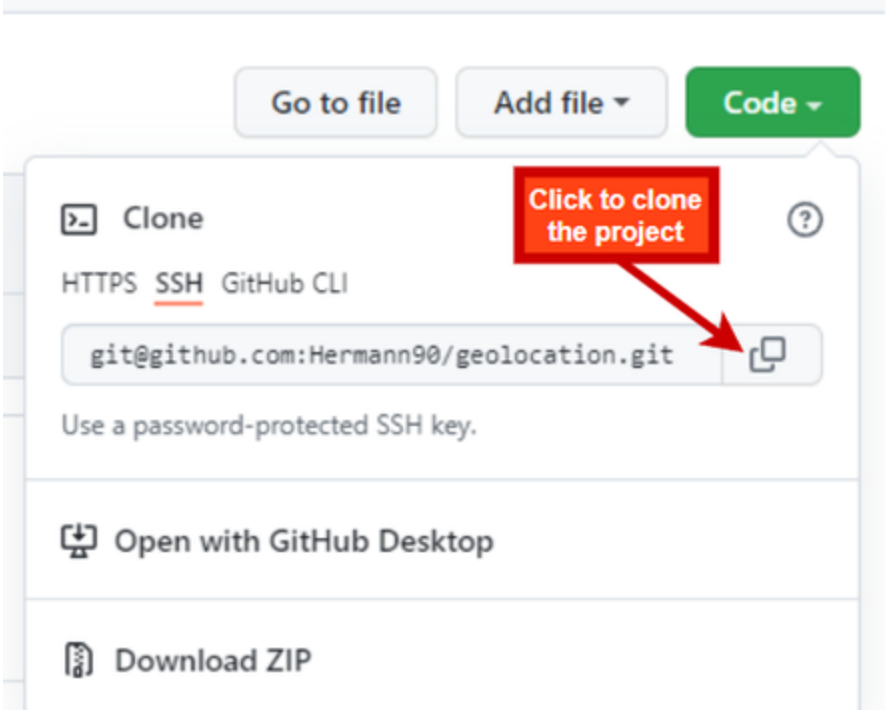
a- Repository creation

- Give it a valid **name: geolocation**
- Make sure the repository is **Public**
- Check the button to **Initialize this repository with a README** file
- Click on **Create repository**



b- Repository clone

- Now, let's copy the clone link of the repository
- click in **Code**
- Click on the sign to **copy the link**

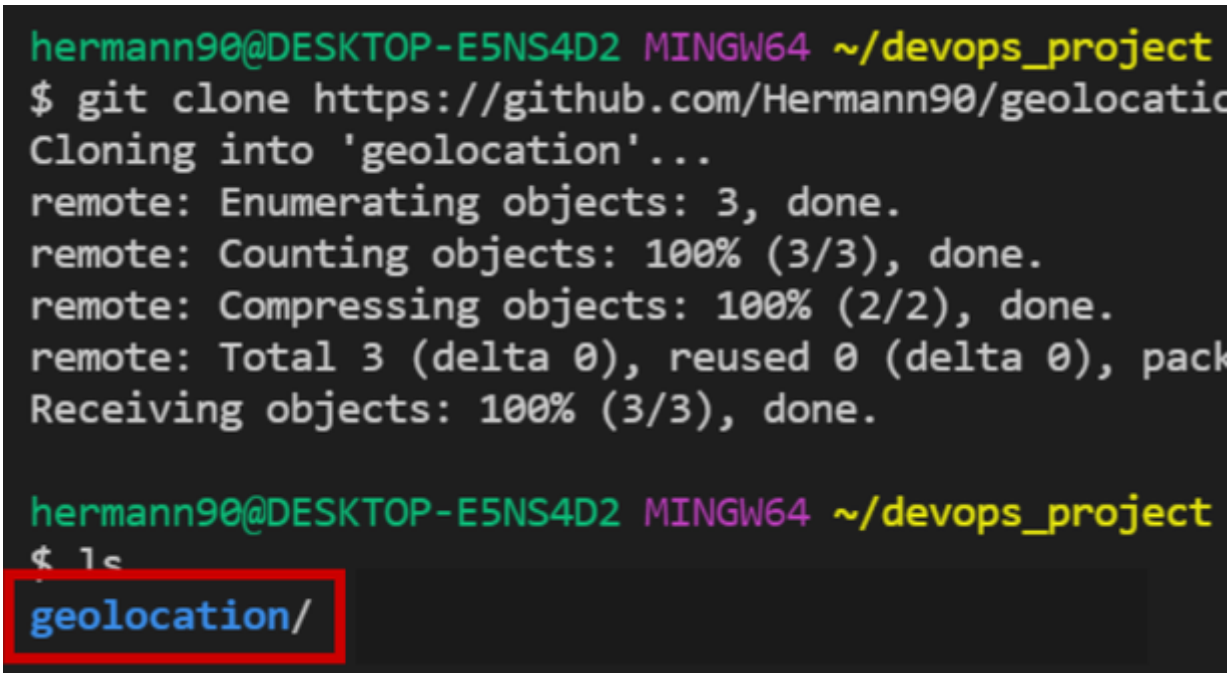


- Now, open the **Git bash** app in your system
- Create a folder for our **DevOps project** and open it

```
mkdir devops_project
cd devops_project
```

- Use the **$ git clone** command to clone the repository here

```
git clone paste-the-link-here
ls
```

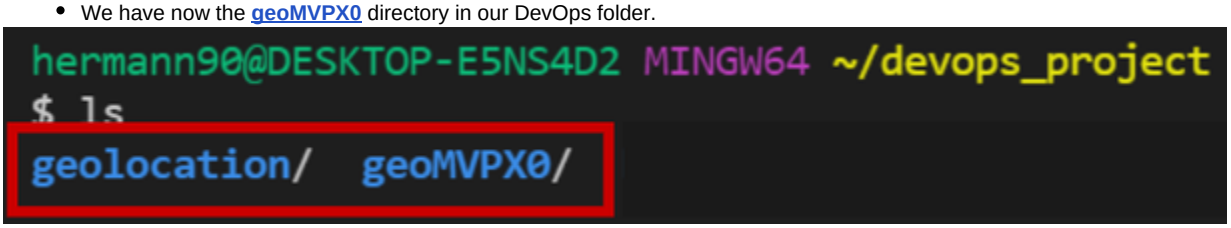


c- Clone the Alain Pierre team's Project

- Now, we will **clone the Alain Pierre repository** which contains the source code of the Geolocation project.

```
$ git clone https://github.com/utrains/geoMVPX0.git
```

- We have now the **geoMVPX0** directory in our DevOps folder.



- **copy the contents of the project team's repository to our test repository**
- we have to copy the content of the **geoMVPX0** repository in the **geolocation** repository

```
cp -r geoMVPX0/* geolocation/
cd geolocation
ls
```

```
hermann90@DESKTOP-E5NS4D2 MINGW64 ~/devops_project
$ cp -r geoMVPX0/* geolocation/

hermann90@DESKTOP-E5NS4D2 MINGW64 ~/devops_project
$ cd geolocation/

hermann90@DESKTOP-E5NS4D2 MINGW64 ~/devops_project/geolocation (main)
$ ls
mvnw*  mvnw.cmd  pom.xml  README.md  src/  target/
```

d- Commit the changes in our repo

```
git add --all
git commit -m "First commit for Geolocation"
git push origin main
```

- if you open your repository, you will be able to see all the files and folders that are added to it.
- now we have the source codes of the geolocation project in a repository that will allow us to perform the **Sonar analysis**.

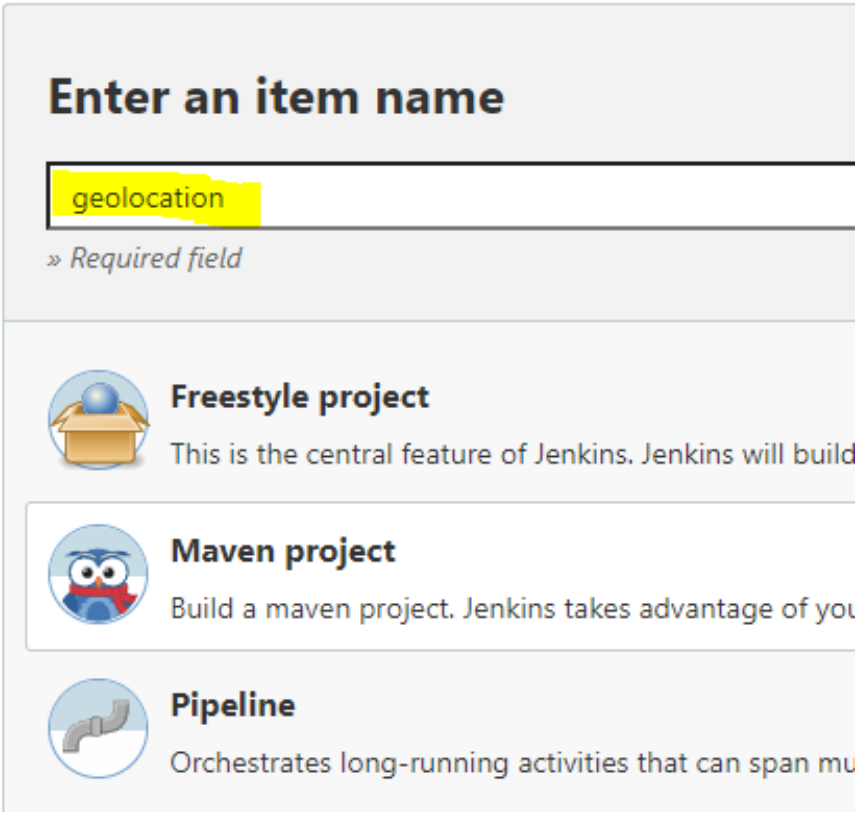
Step 4 : create a maven project in Jenkins and link it to our GitHub project named geolocation

1- Login in your **Jenkins Web Interface**

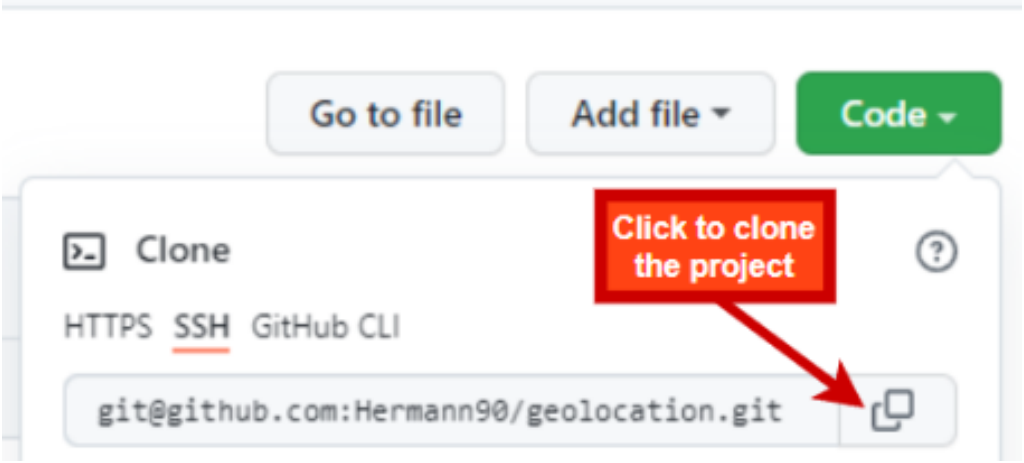
2- click on **New Item**

3- select **Maven Project**

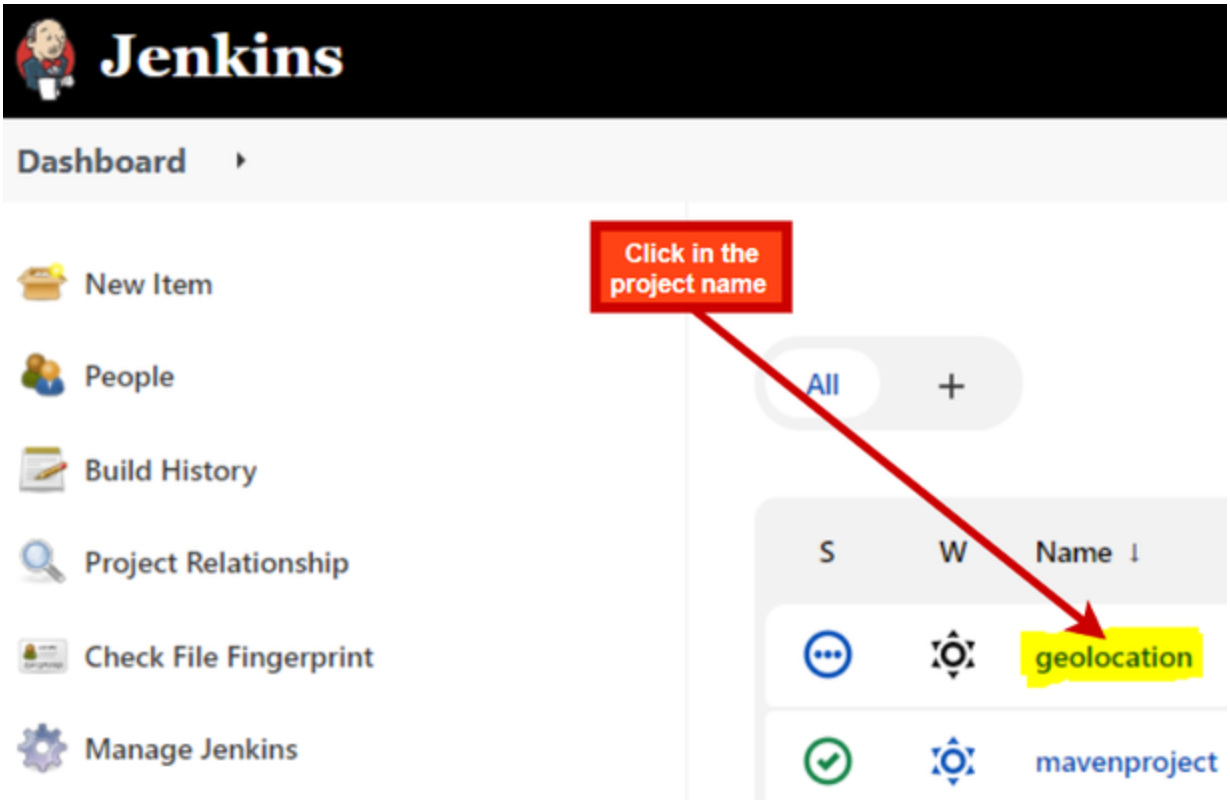
4- click in **Ok** to create new maven project in Jenkins.



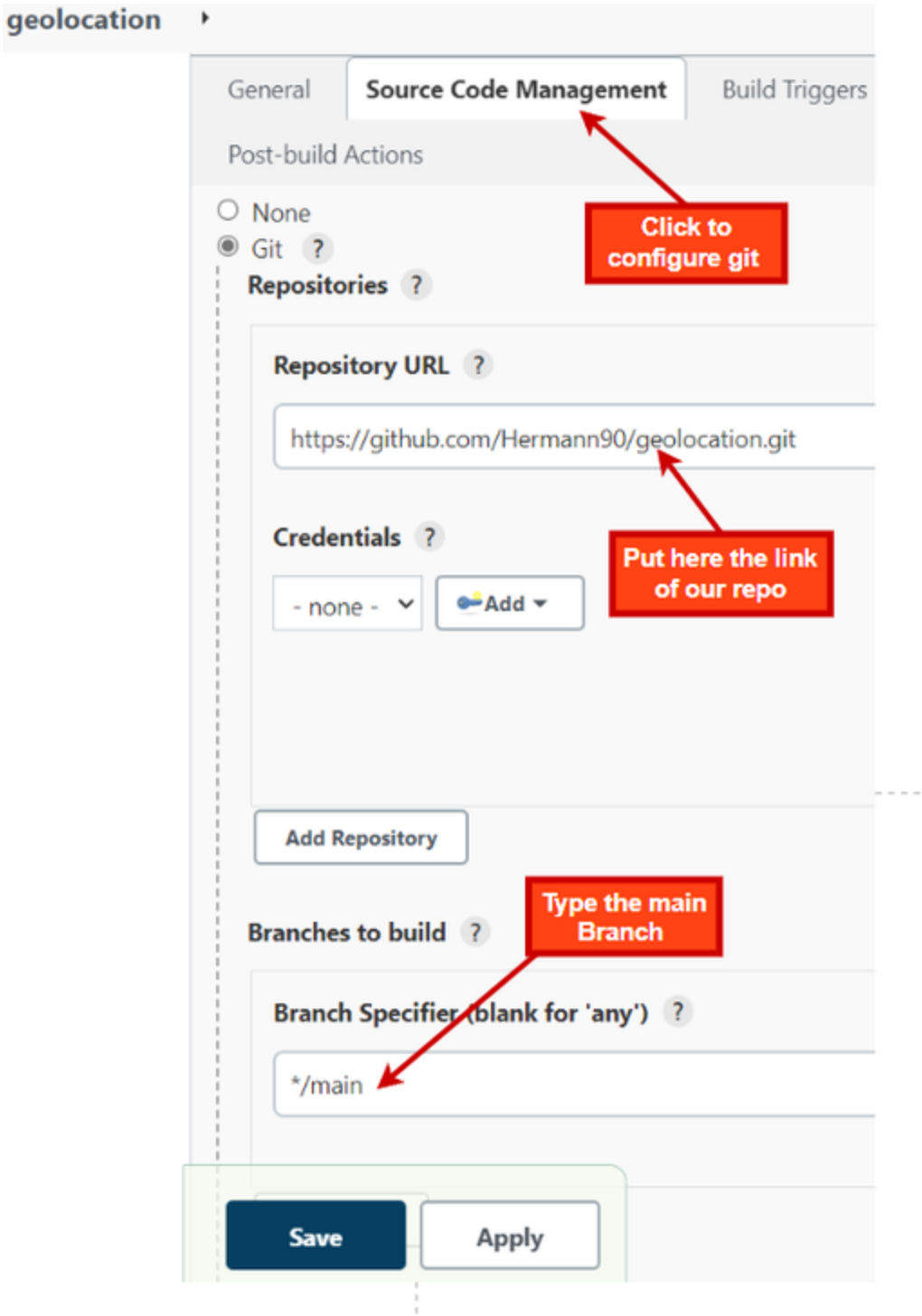
- Open your **GitHub account name** and choose the **geolocation repository**
- Click on **Code** and **copy the link** for clone creation



- In the **Jenkins tab of the browser** choose the maven project that we are created (see slide 25)



- click on **configure**, to configure the maven project
- In the **Source code management** tab, click on the radio button **Git**
- **Repository URL: Paste the link for clone creation**
- Allow it on the **main branch**
- click on **Apply,** then **save**
- **at this stage, the project is successfully built**



Step 5 : create an account in SonarCloud

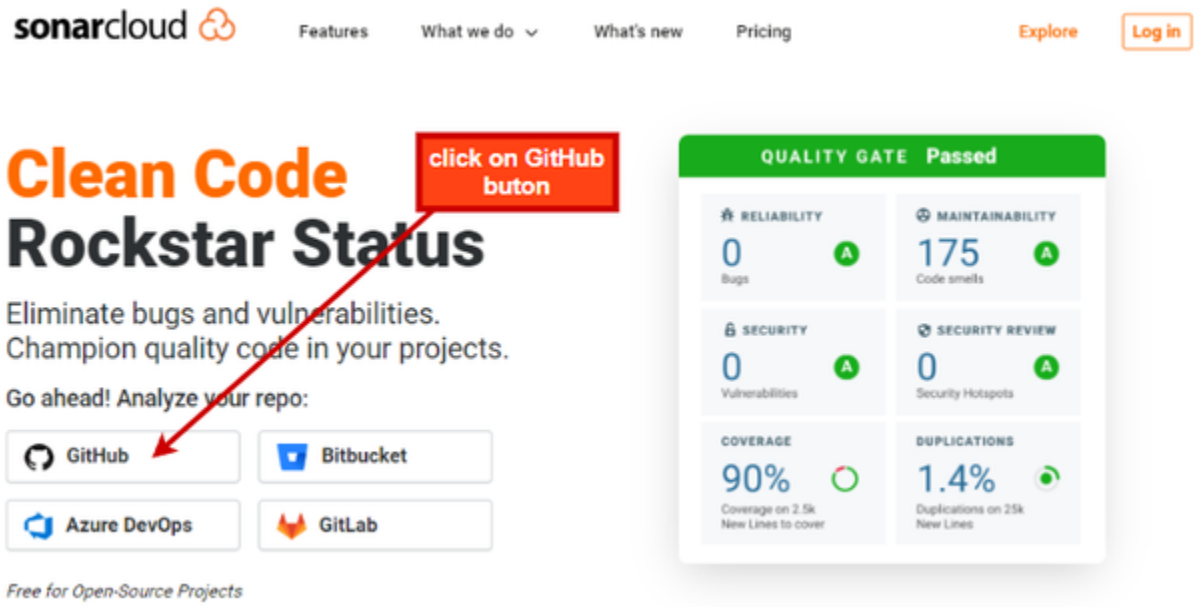
- **SonarCloud** is the leading online service to catch **Bugs** and **Security Vulnerabilities** in your code repositories.
- in this step, we will click on the **SonarCloud** link, create an account if we don't have one already, then select our repository for some configuration.

a- open the sonarcloud link by clicking on this link.

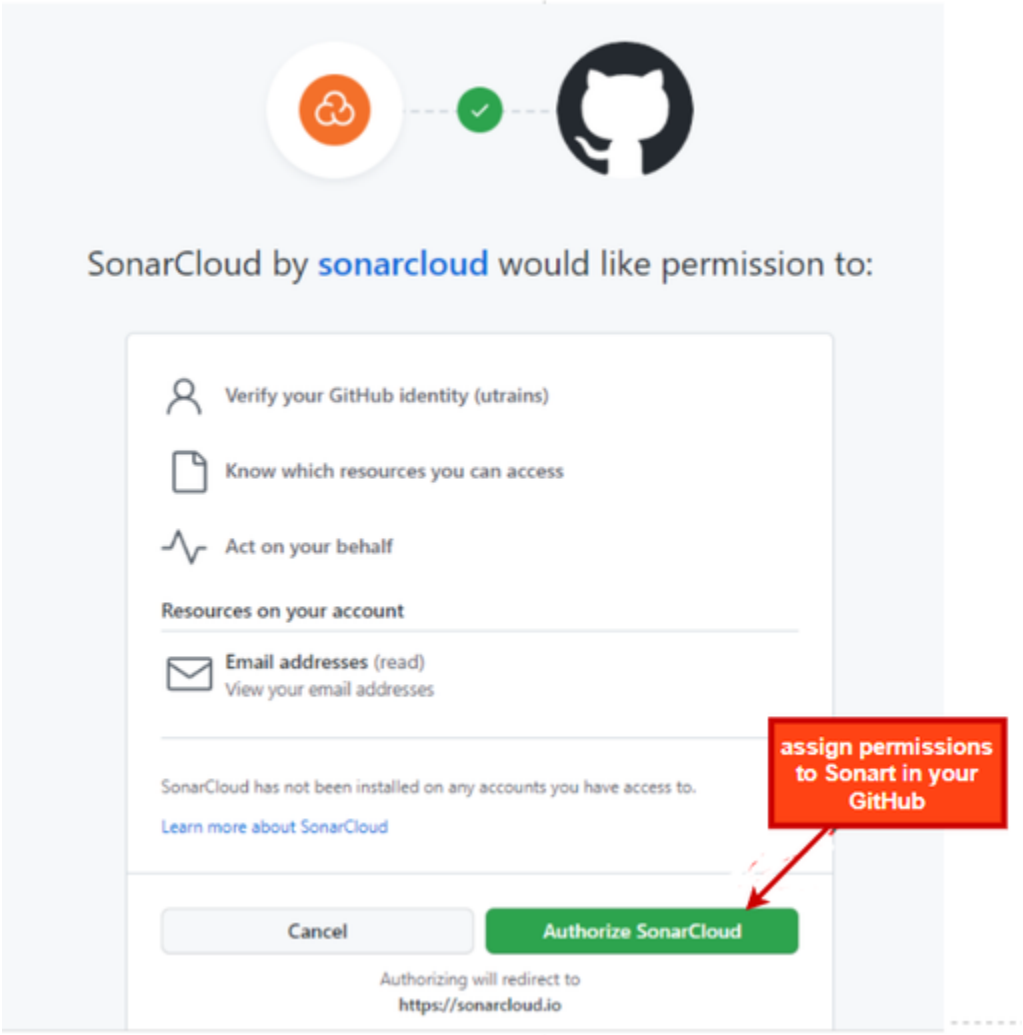
click in this link to open sonar creation account page

[Click here to Create Sonarcloud Account]

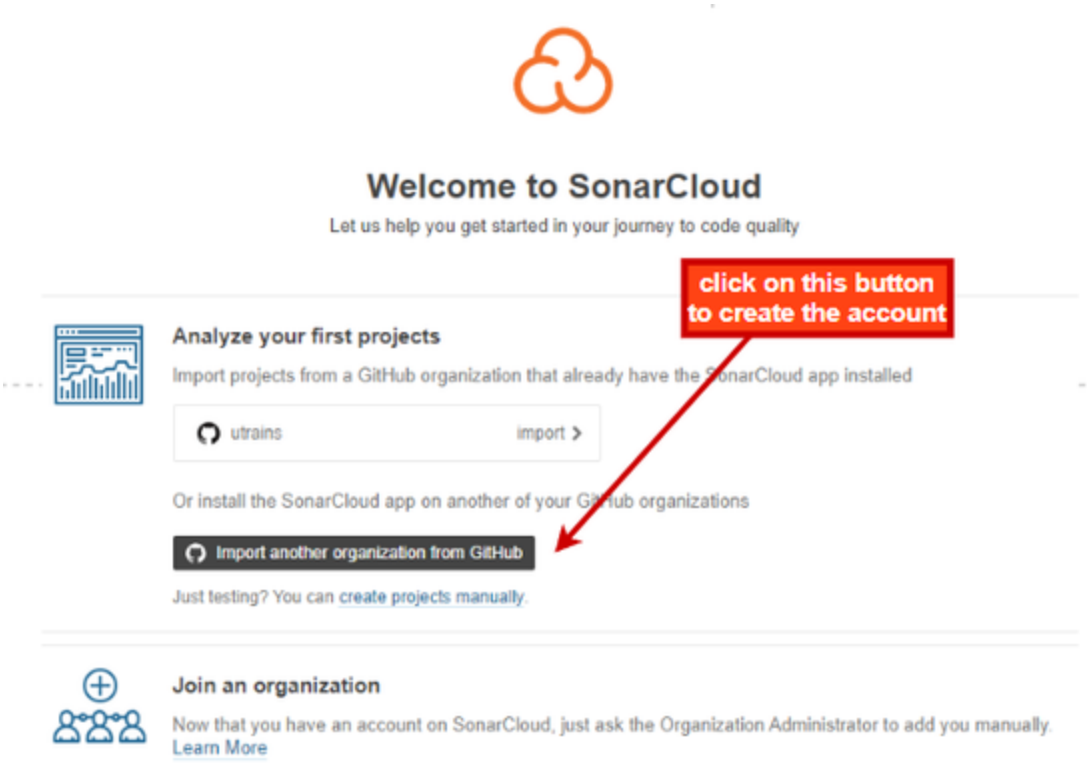
- if you have never created an account in SonarCloud, this page will appear. we will use our GitHub account to quickly create an account.



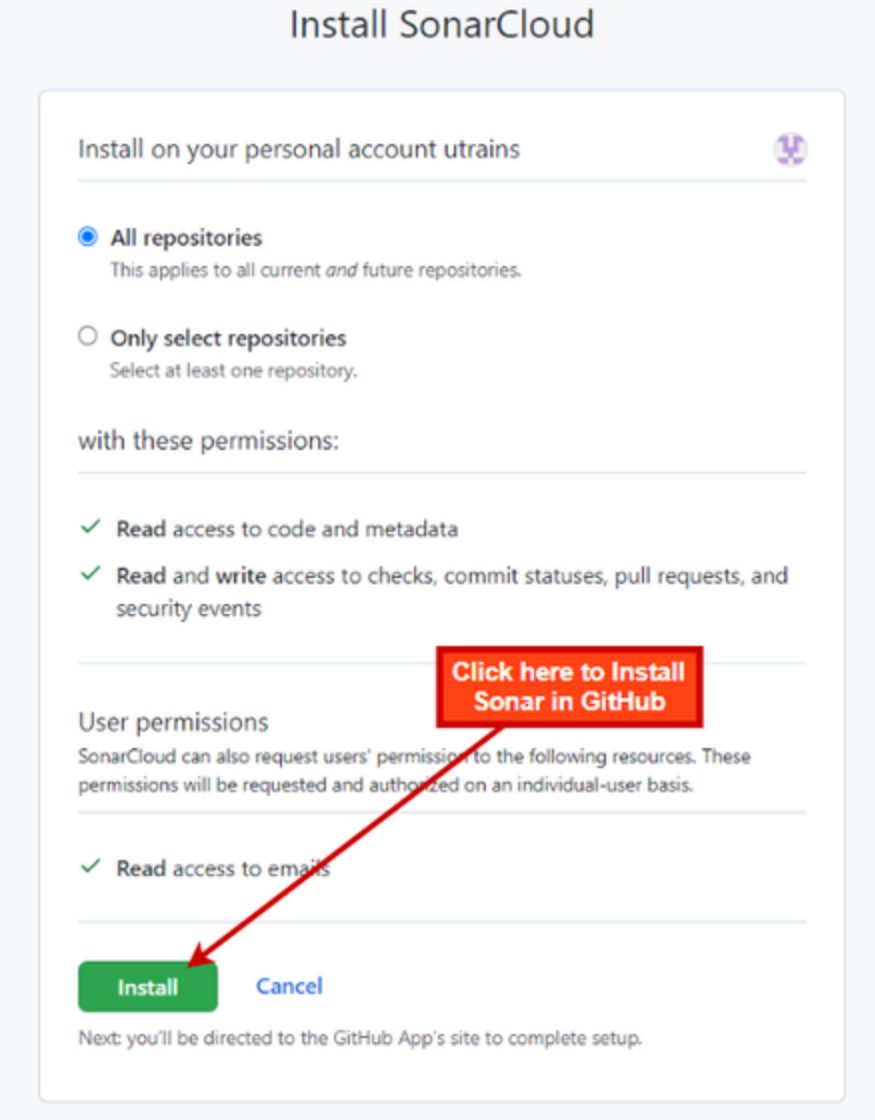
- **authorize Sonar to access your GitHub account, by clicking on the button below.**



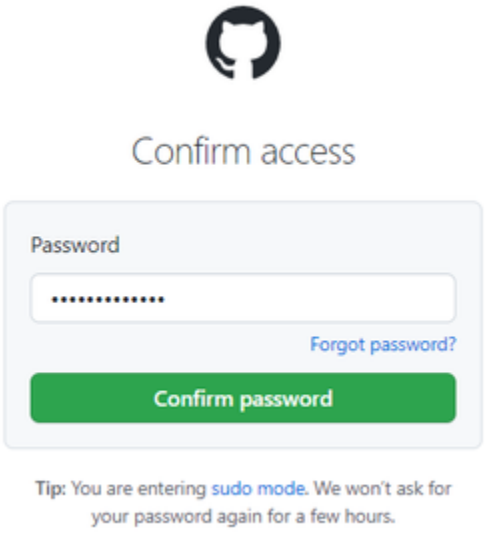
- a page appears with our GitHub account and asks us to import a new organization. in Sonar, accounts are linked to organizations.



b- install Sonar in your GitHub

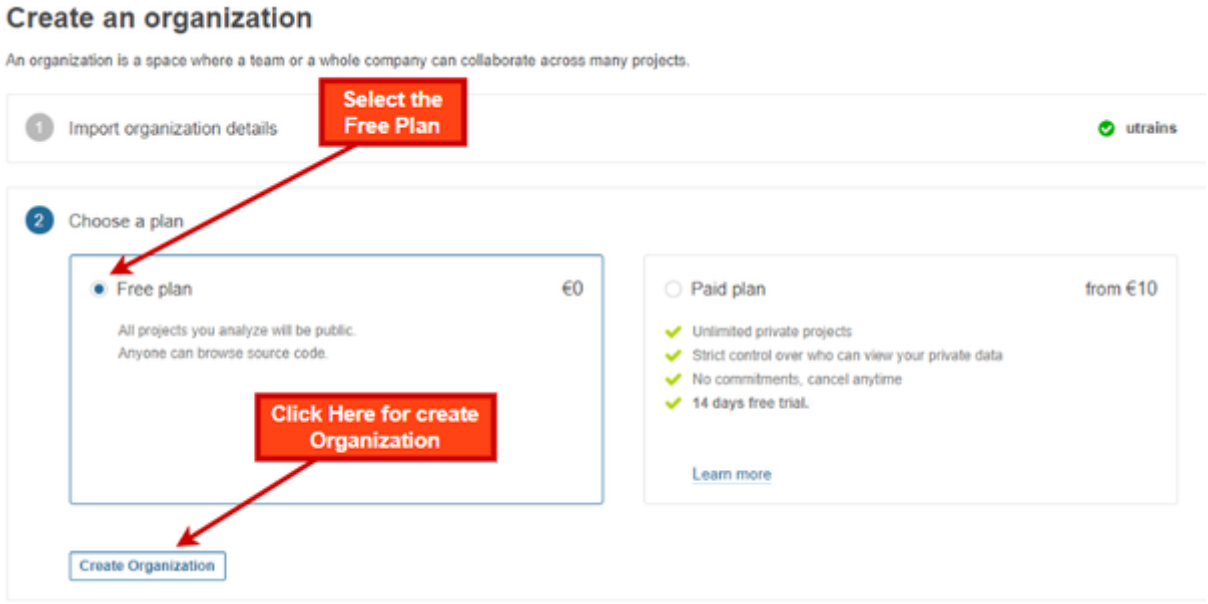


- type your GitHub Password then click in confirm button

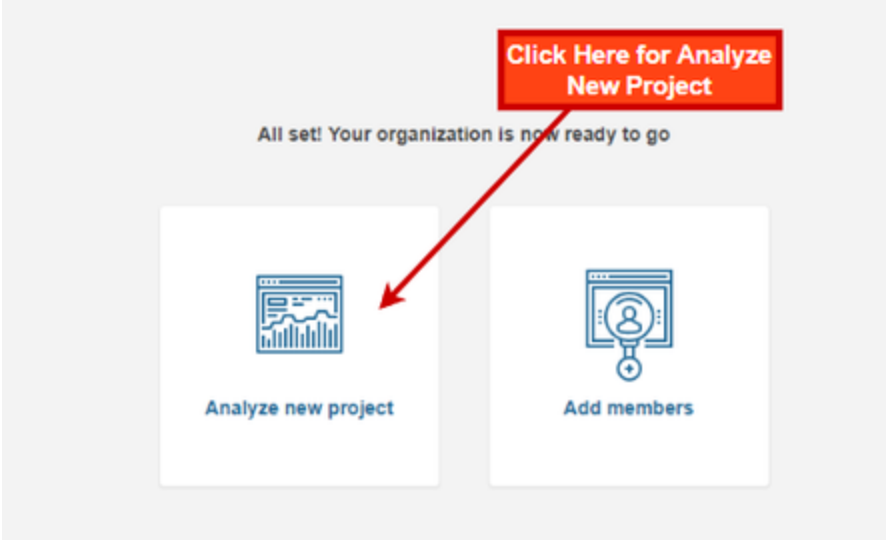


- create your Organization : here, just check that the fields are well filled in then click on continue to create your Organization

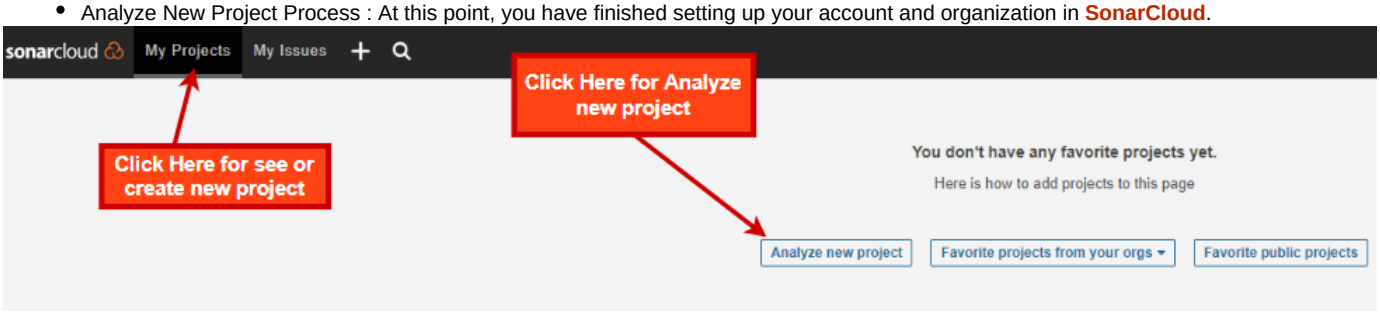
- payment plan : we choose here a free payment, but in company, most of the time, it is a paying plan for more functionality.



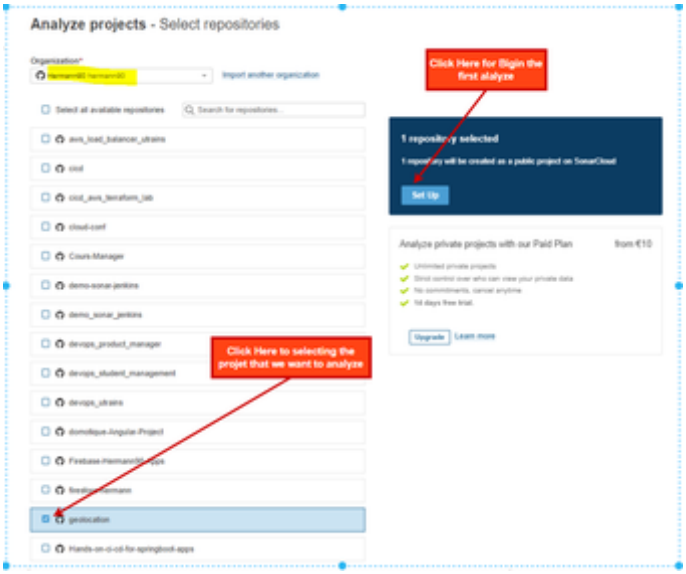
c- first analyze



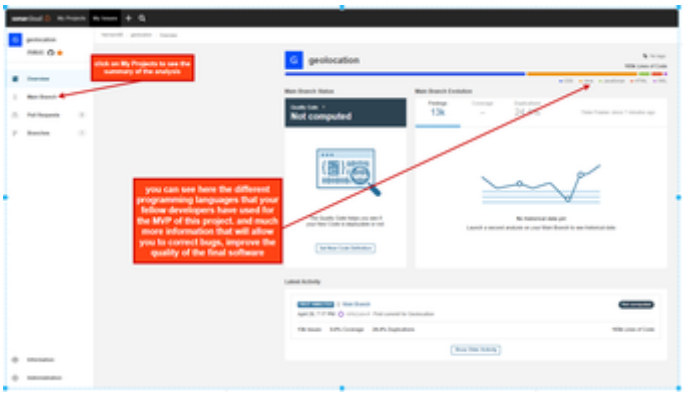
- Analyze New Project Process : At this point, you have finished setting up your account and organization in **SonarCloud**.



- The list of all the projects you have in your GitHub account is displayed here. just check the project we want to analyze. in our case, it would be **geolocation**.



- After about 5 minutes, you will see the results of the very first analysis of our project displayed in **SonarCloud**. Now we have to configure our project in collaboration with the developers:
  - provide them with the information to add in the pom.xml file to reference the sonar account in order to automate the analysis with our Jenkins server.
  - configure our Jenkins server so that the analysis is done automatically when the developers have done the push on the main branch
  - provide the analysis results to the developer so that they can improve the quality of the project source code.

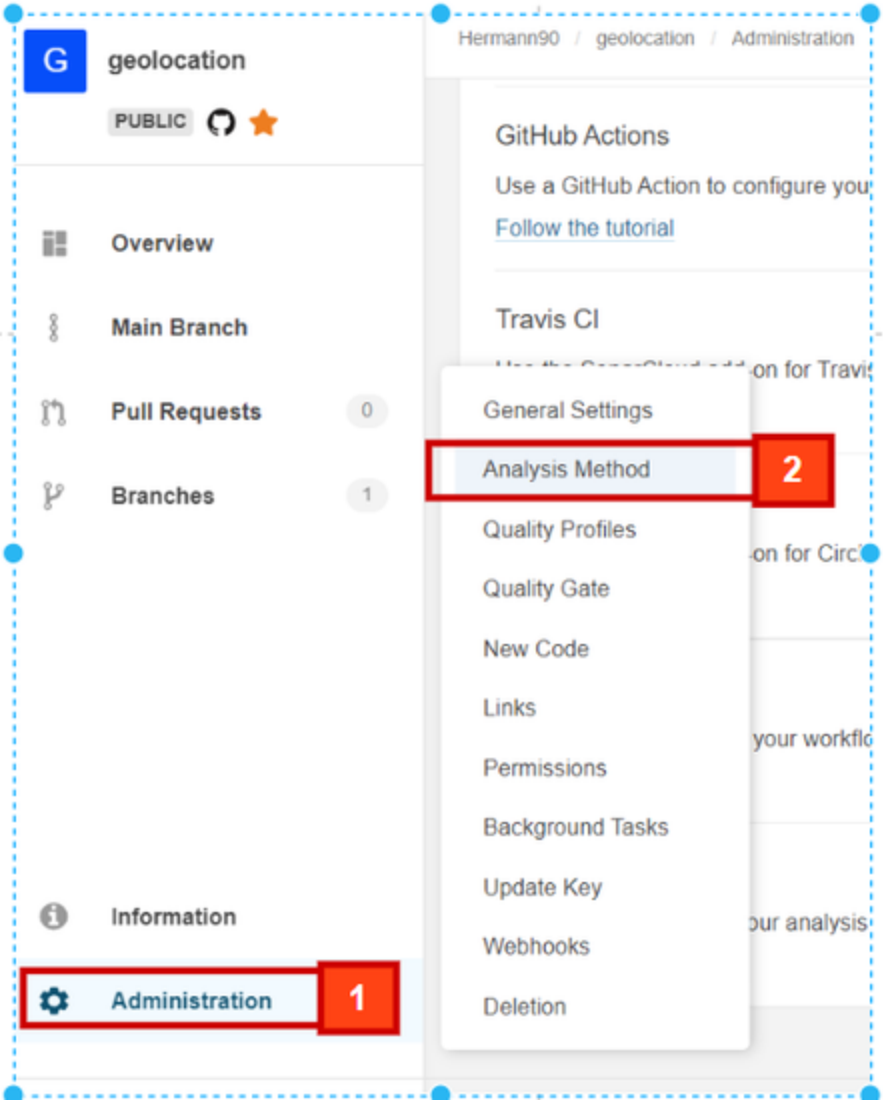


- **click on Main Branch to see the summary of the analysis :**
  Here clicking on each of the numbers will give the DEVs the details to solve each of the vulnerabilities.

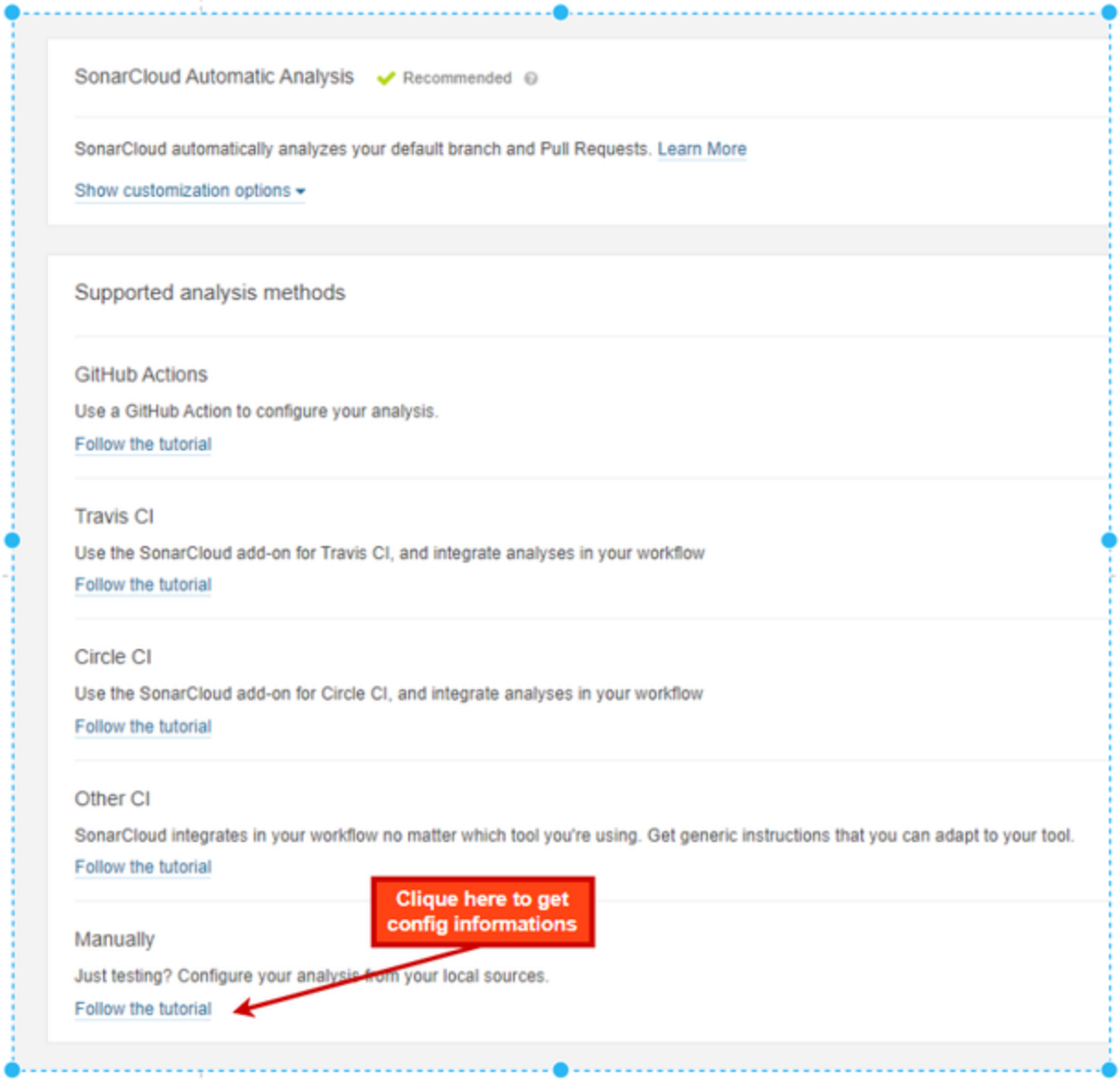
II- Now, the automation of the analysis after each push on the main branch by the developer

1- retrieving information for the automation of the analysis

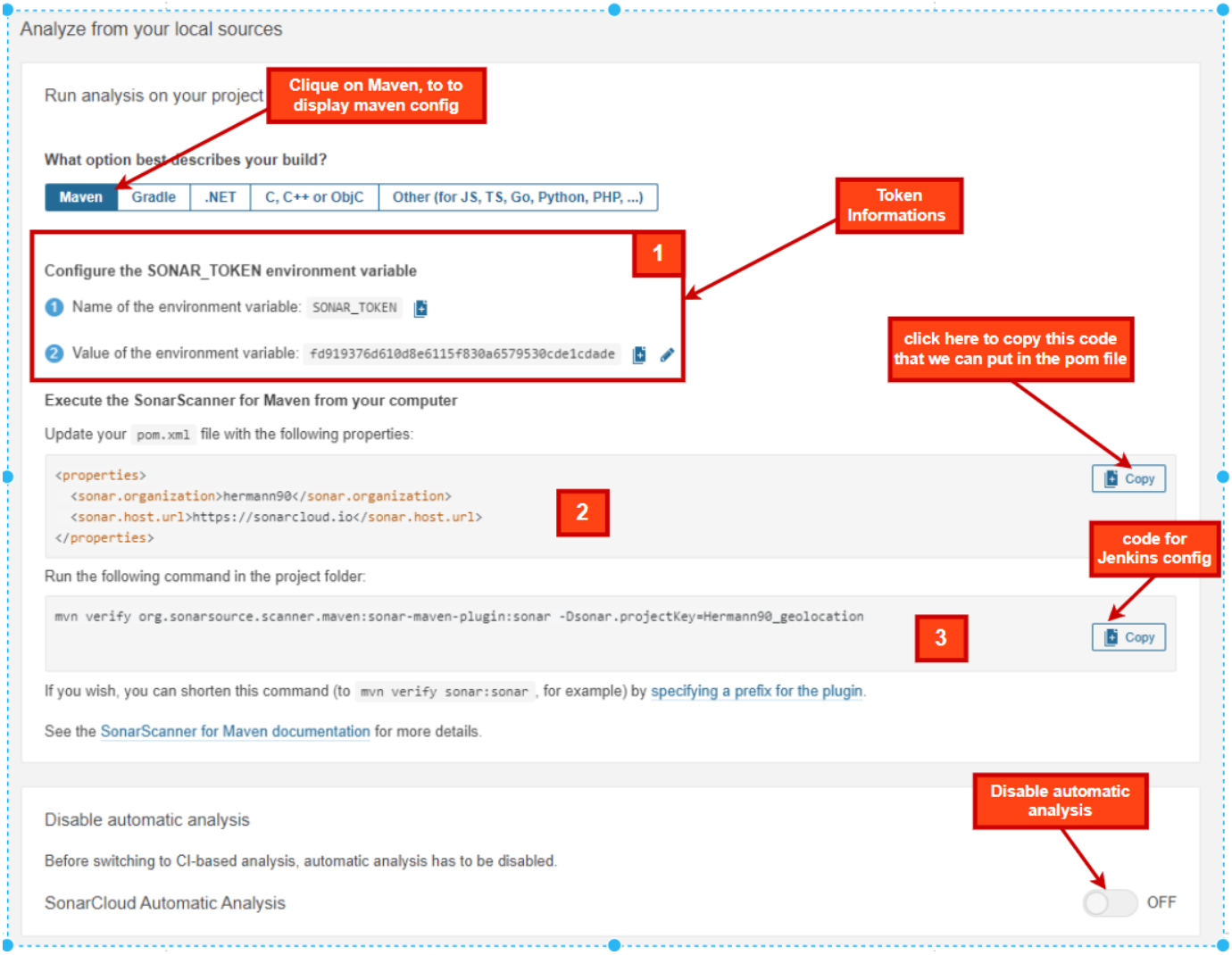
Process : **Administration Analysis Method**



The page that appears contains the information for the different methods of automation. Here, we will just choose the manual method, because Jenkins does not exist on the different methods.



All Config information's :



On this image, there are couple information needed to configure our project: the token, the maven properties, and the maven command to run.

1- Token Information :
this token will allow communication between **SonarCloud** and Jenkins or any other tools

2- Maven Properties :
here we have the configuration properties of our Maven project. these properties must be added in the pom.xml file. generally, it is the DevOps engineer who provides these config to the developer.

Example:

```
<properties>
  <sonar.organization>hermann90</sonar.organization>
  <sonar.host.url><https://sonarcloud.io</sonar.host.url>>
</properties>
```

3- Run Command :

command that will launch the analysis from Jenkins server.

```
mvn verify org.sonarsource.scanner.maven:sonar-maven-plugin:sonar -
Dsonar.projectKey=Hermann90_geolocation
```

But here we want jenkins to execute automatically not the developer running to command locally.

Step 6 : Integrate SonarCloud in Jenkins and perform the code analysis

Based on the information available in the image above, we will collect the information for the configuration in Jenkins. we need :

1- project key :
We have this information in **mvn command** in this image above. in my case we have **sonar.projectKey=Hermann90_geolocation**

2- project organization :

We have this information in the maven properties
sonar.organization=hermann90

3- Host URL :

Is the URL or address of the server where Sonar is located. We have this information in the maven properties
sonar.host.url=https://sonarcloud.io

4- Project Login :

For the login, we will use the token that is in the image above. for my case it would be :

sonar.login=fd919376d610d8e6115f830a6579530cde1cdade

5- Other information (source and java binaries)

Information to tell Sonar the directory where the source codes are located that it wants to analyze, and the directory where the binary files are located.

sonar.sources=src

sonar.java.binaries=.

summary of configuration information

```
sonar.projectKey=Hermann90_geolocation
sonar.organization=hermann90
sonar.host.url=https://sonarcloud.io
sonar.login=fd919376d610d8e6115f830a6579530cde1cdade
sonar.sources=src
sonar.java.binaries=.
```

Project configuration in Jenkins

1- copy the properties you got from sonarcloud. in my case it is the code below

```
sonar.projectKey=Hermann90_geolocation
sonar.organization=hermann90
sonar.host.url=https://sonarcloud.io
sonar.login=fd919376d610d8e6115f830a6579530cde1cdade
sonar.sources=src
sonar.java.binaries=.
```

2- open the Jenkins Web interface.

- open the Jenkins Web interface.
  - login in the web interface of jenkins. As a reminder, the information is :
    - **login : utrains-root**
    - **password: school1**

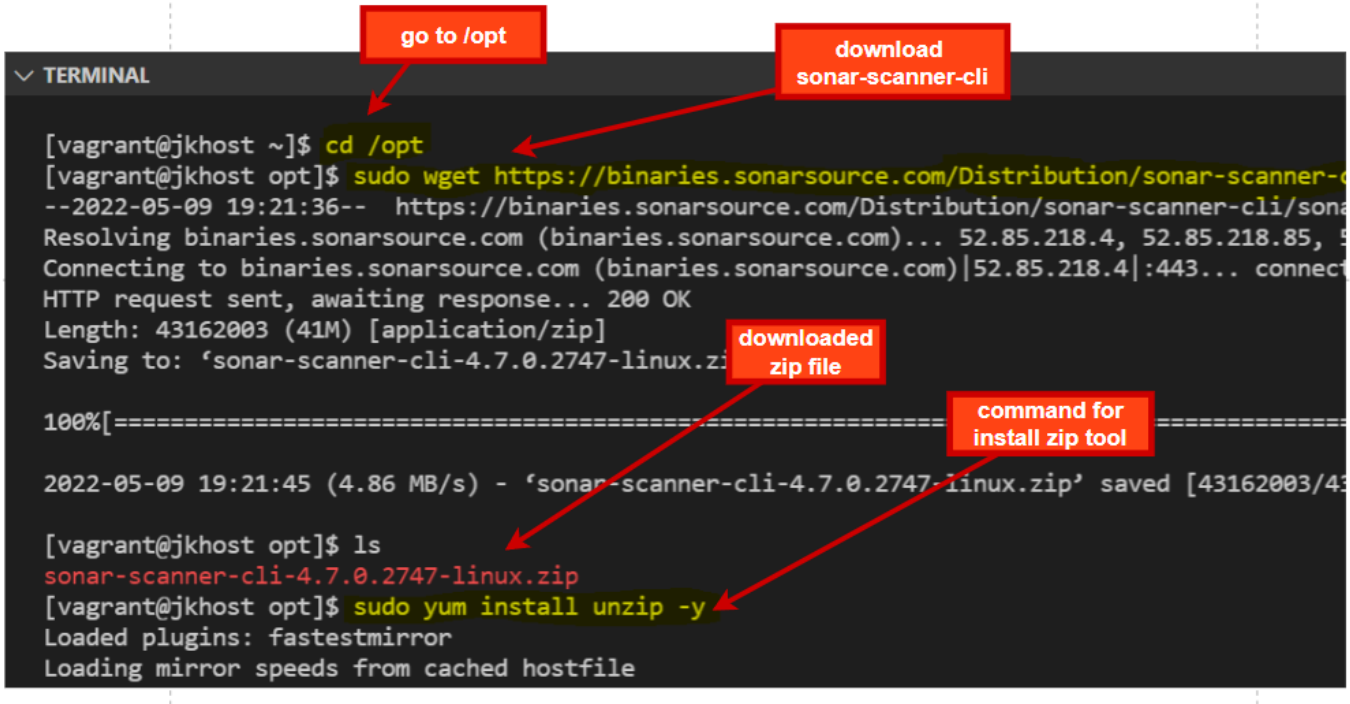
3- install the sonar-scanner tool and the plugin in Jenkins

two step for this installation.

step 1 : sonar-scanner installation in Jenkins

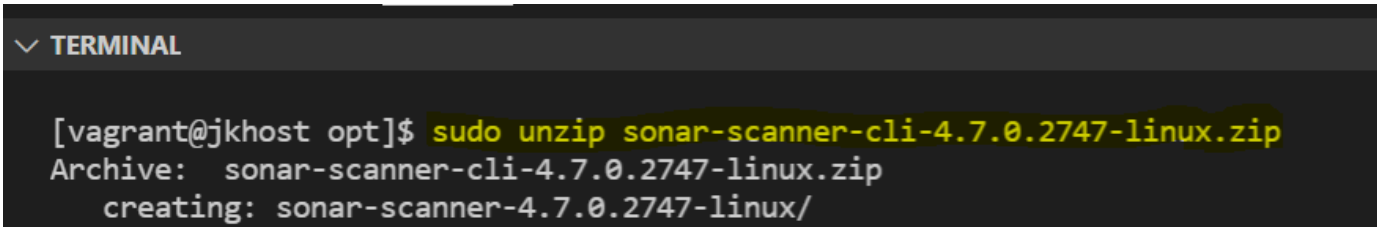
- connect to the terminal using **ssh** to the **jenkins server**
- download **sonar-scanner-cli using wget**
- install the zip tool, using the yum command. this tool can help you to **unarchive zipped** files

```
ssh vagrant@192.168.58.177
sudo wget https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-4.7.0.2747-linux.zip
sudo yum install unzip -y
```



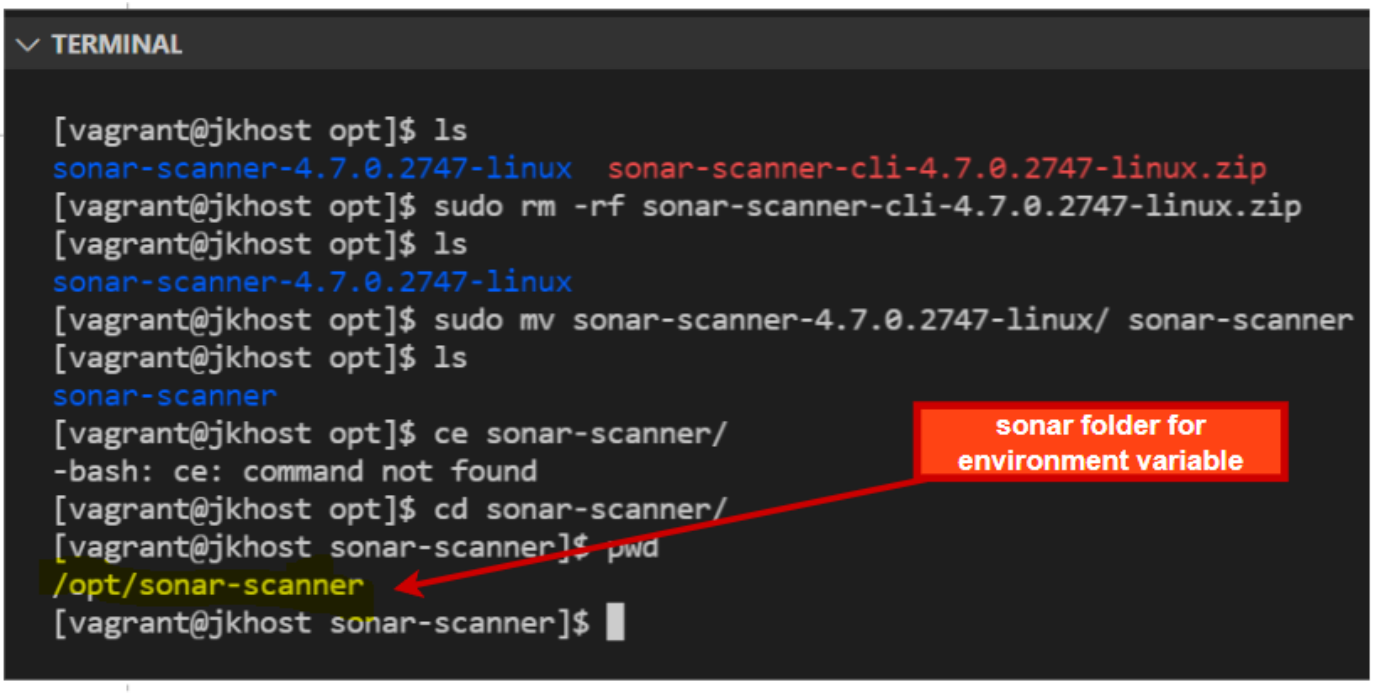
- unzip the

```
unzip sonar-scanner-cli-4.7.0.2747-linux.zip
```



- make ls see the folder that unzip are create
- delete the downloaded zip file, then move the folder created by the unzip command to a folder called **sonar-scanner**

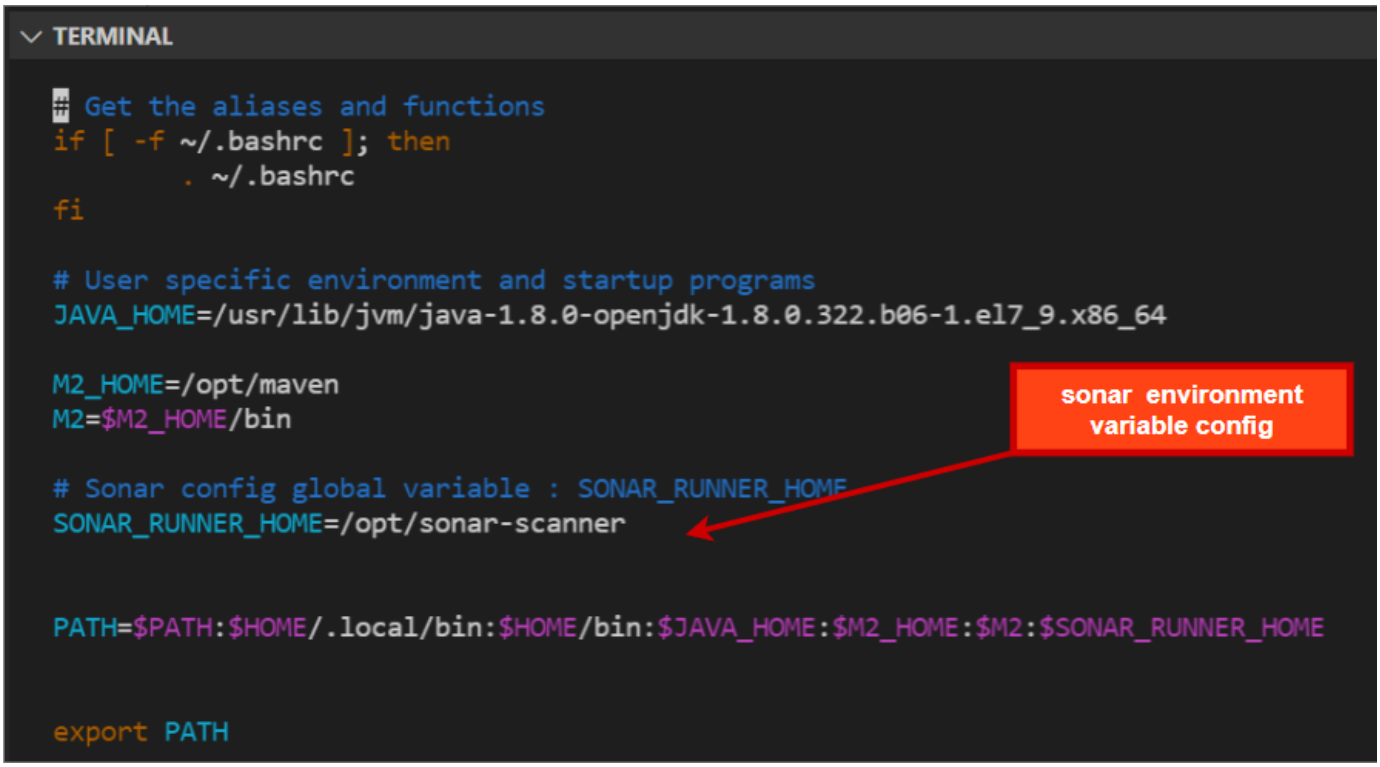
```
ls
sudo rm -rf sonar-scanner-cli-4.7.0.2747-linux.zip
ls
sudo mv sonar-scanner-4.7.0.2747-linux/ sonar-scanner
```



- configure the **sonar-scanner** environment variable

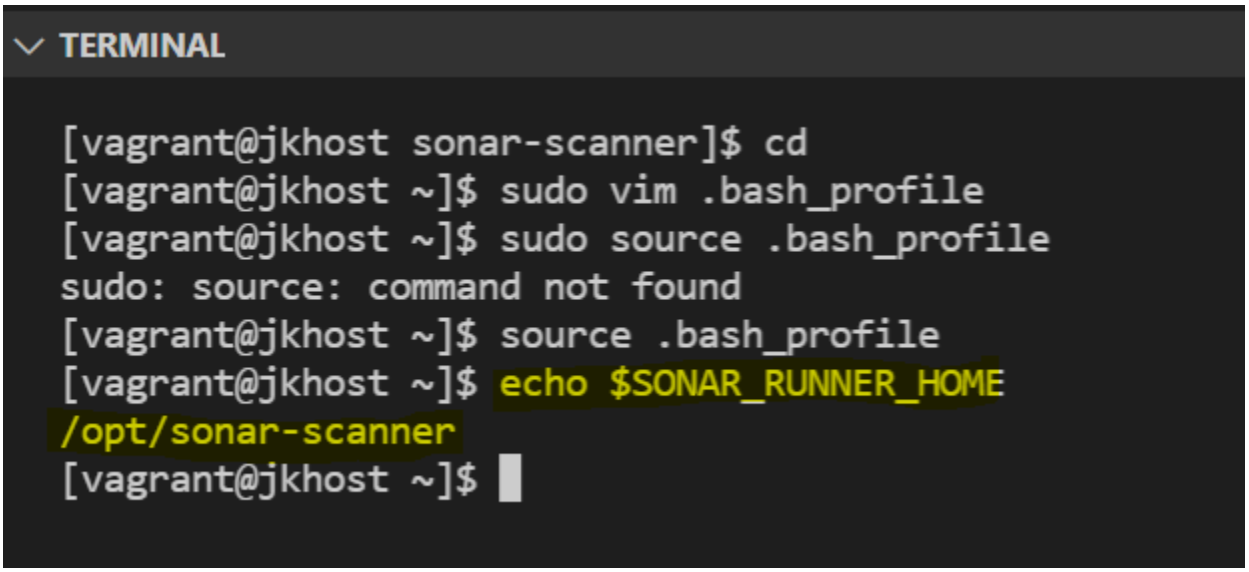
copy the path **/opt/sonar-scanner/** then open the ~.**bash_profile** and set the sonar environment variable

```
sudo vim ~/.bash_profile
```



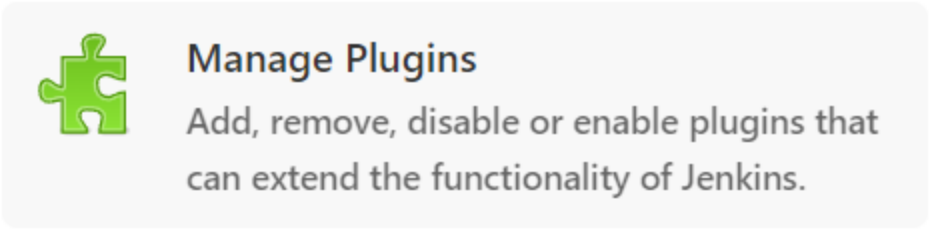
- save the modification then check if the environment variable is correctly set

```
source ~/.bash_profile
echo $SONAR_RUNNER_HOME
```

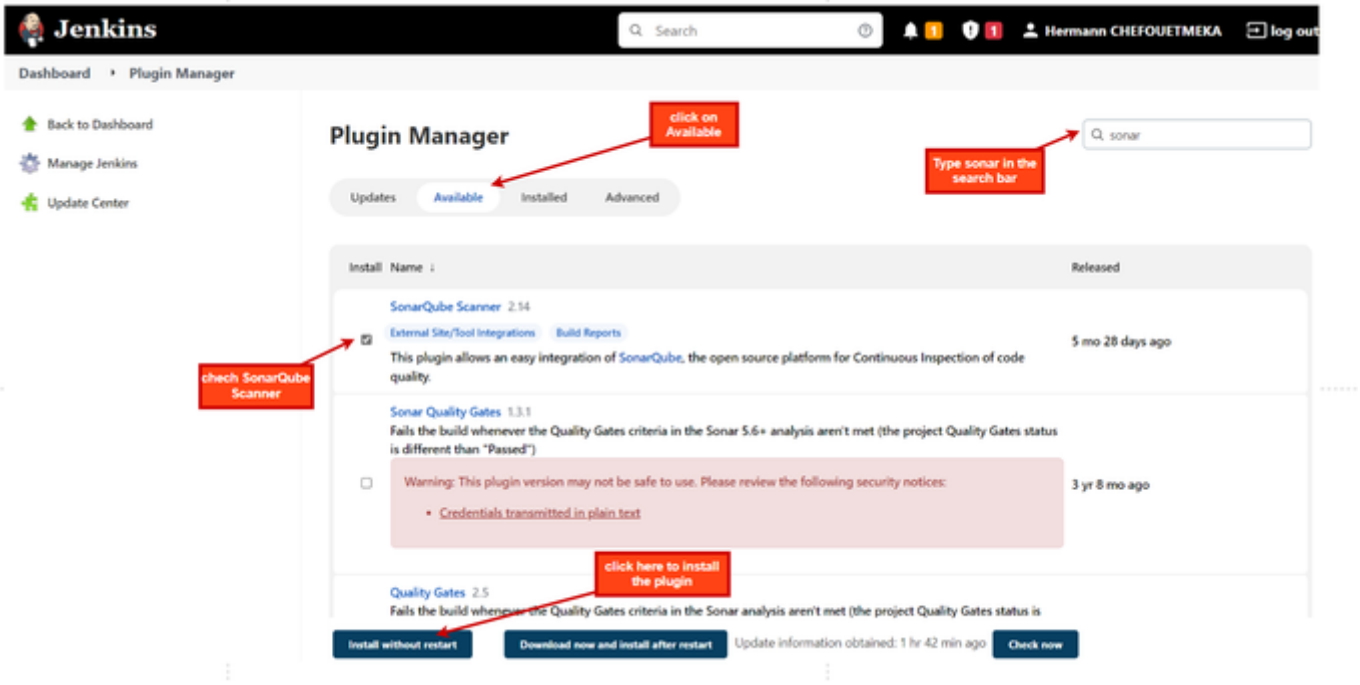


step 2 : sonar-scanner configuration in Jenkins

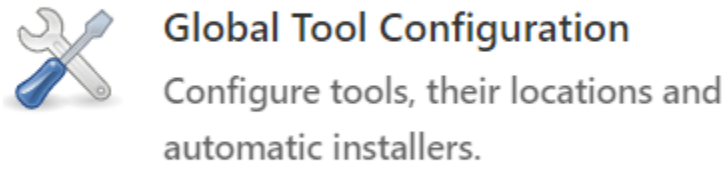
- login to the jenkins web interface
- click on manage Jenkins
- then click on manage plugin



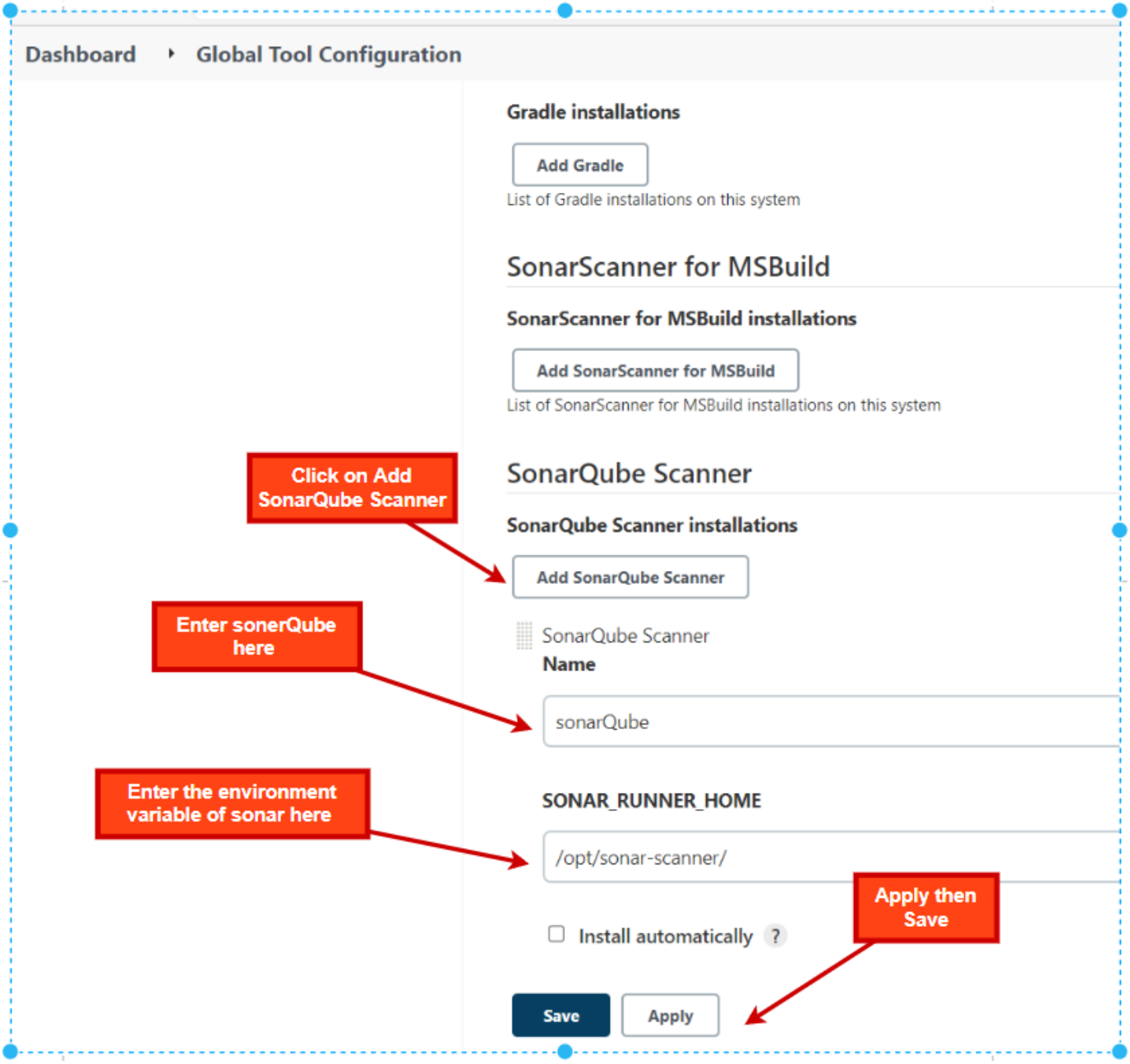
- search **sonarQube** scanner plugin and install it



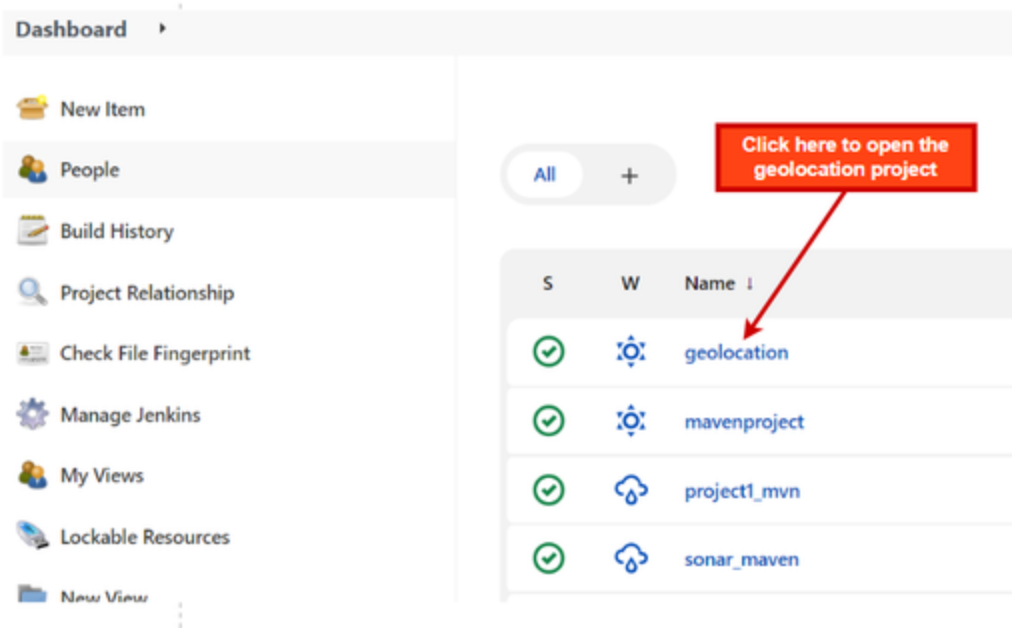
- go to manage jenkins global Tools Configuration



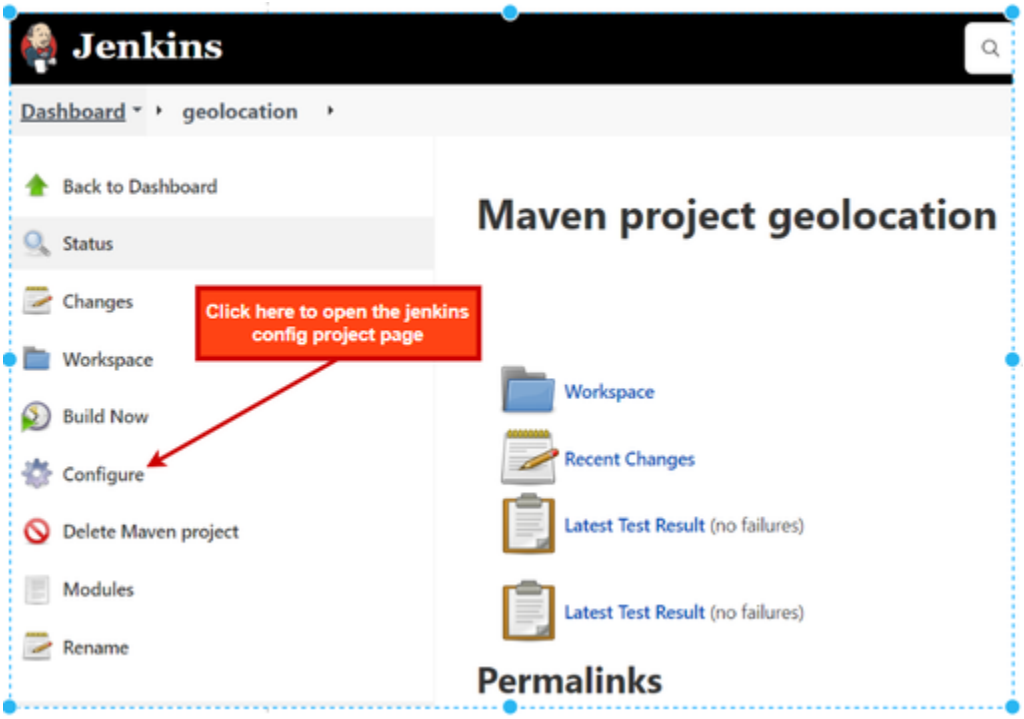
- configure the sonar environment variable.



4-open the geolocation project we have created.

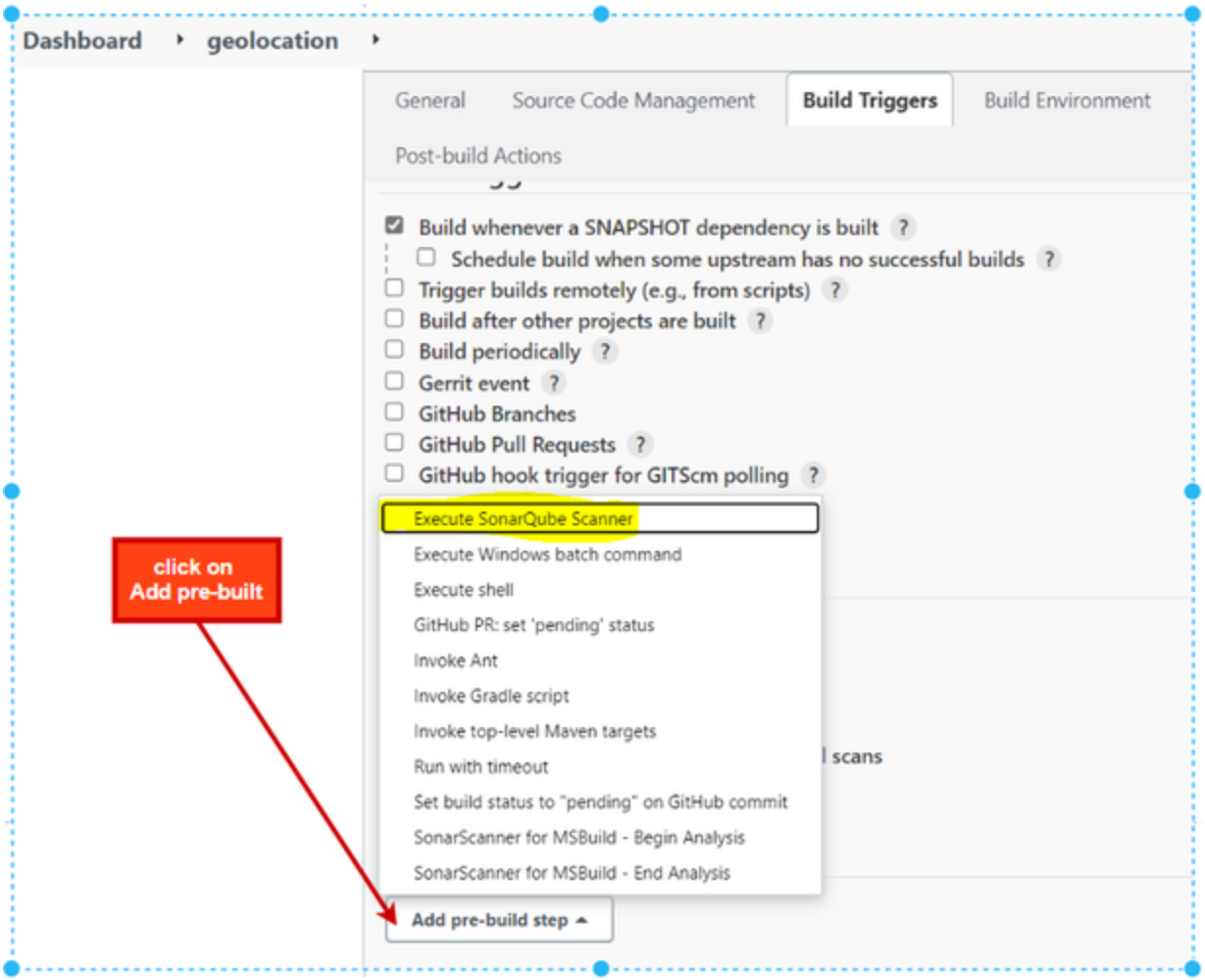


4- project config page in Jenkins



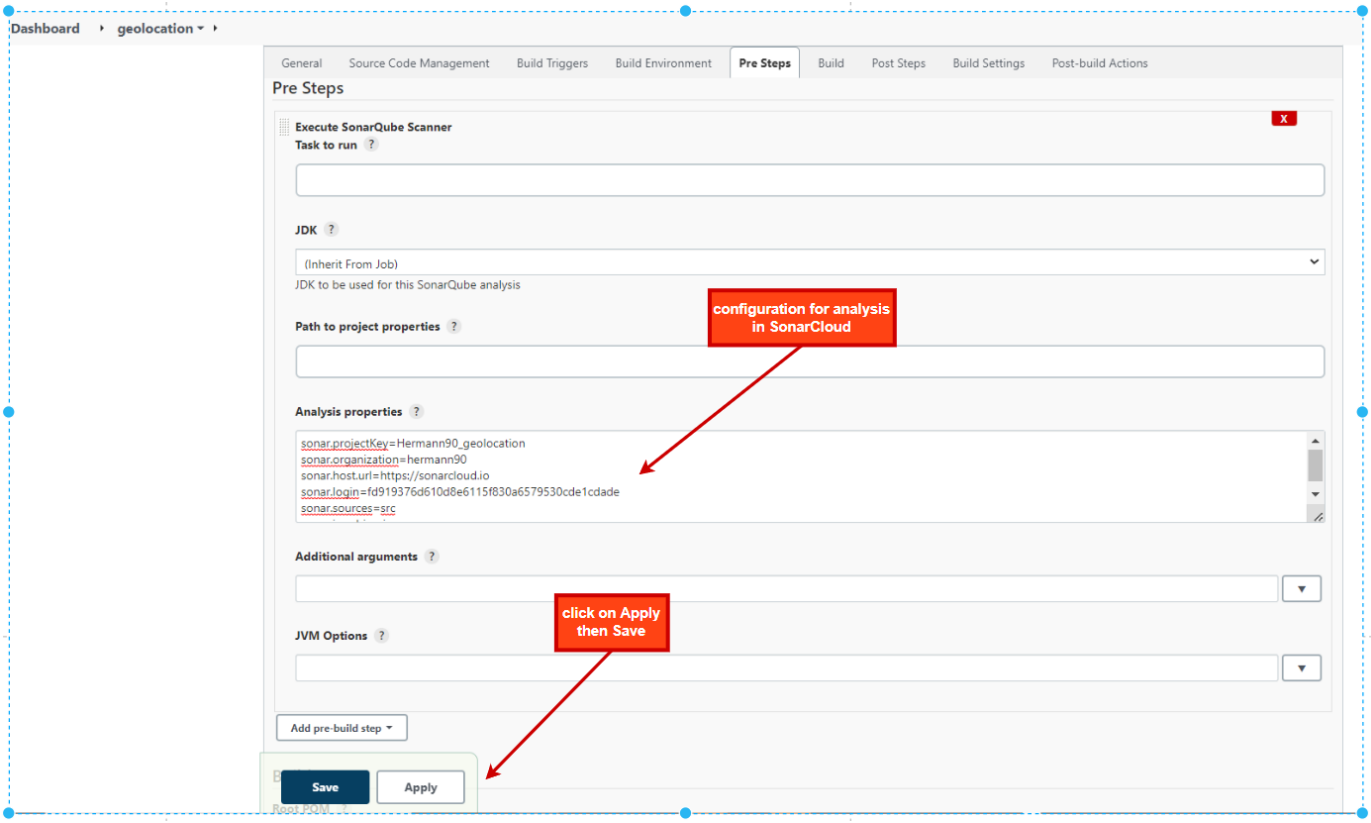
5- add sonarcloud config in the Geolocation Project

- click on **Add pre-built step** then in the popup that appears, choose **Execute SonarQube Scanner**



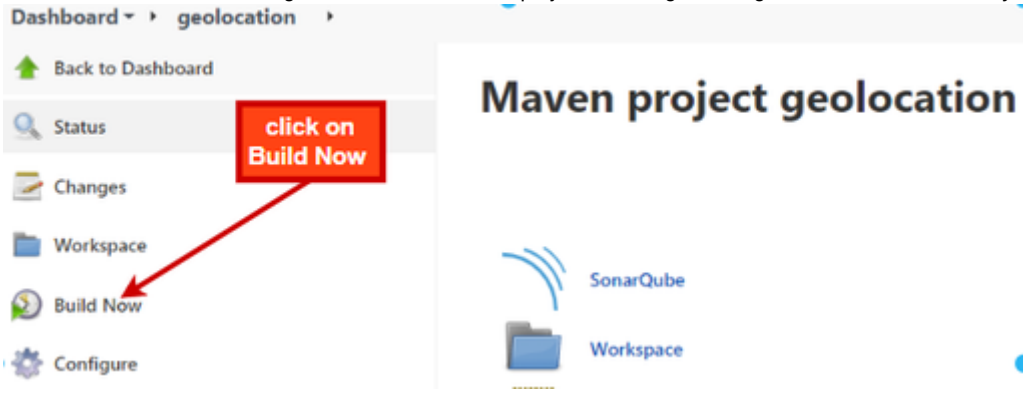
- **Analysis properties** in Pre Step Section

a section appears where we will put the Sonar analysis settings. here, copy and paste in the Analysis properties sub-section all the Sonar properties (projectkey, organization, ...) that we have explained above.

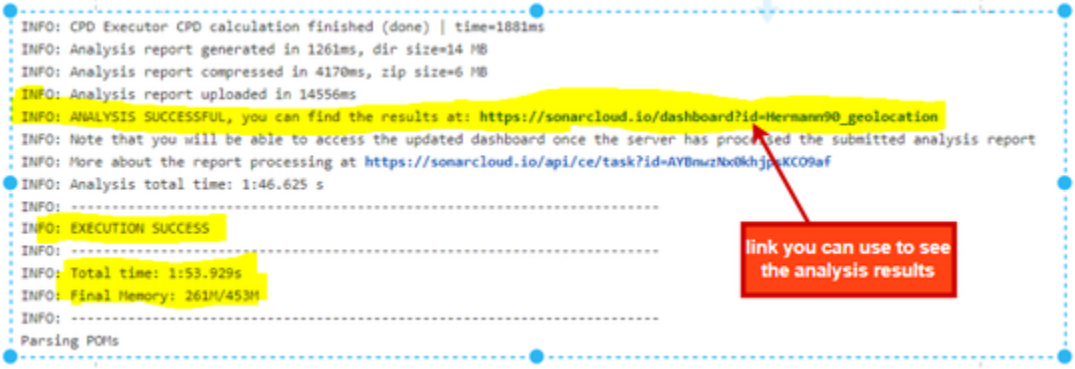


Step 7 : Build and check the Result

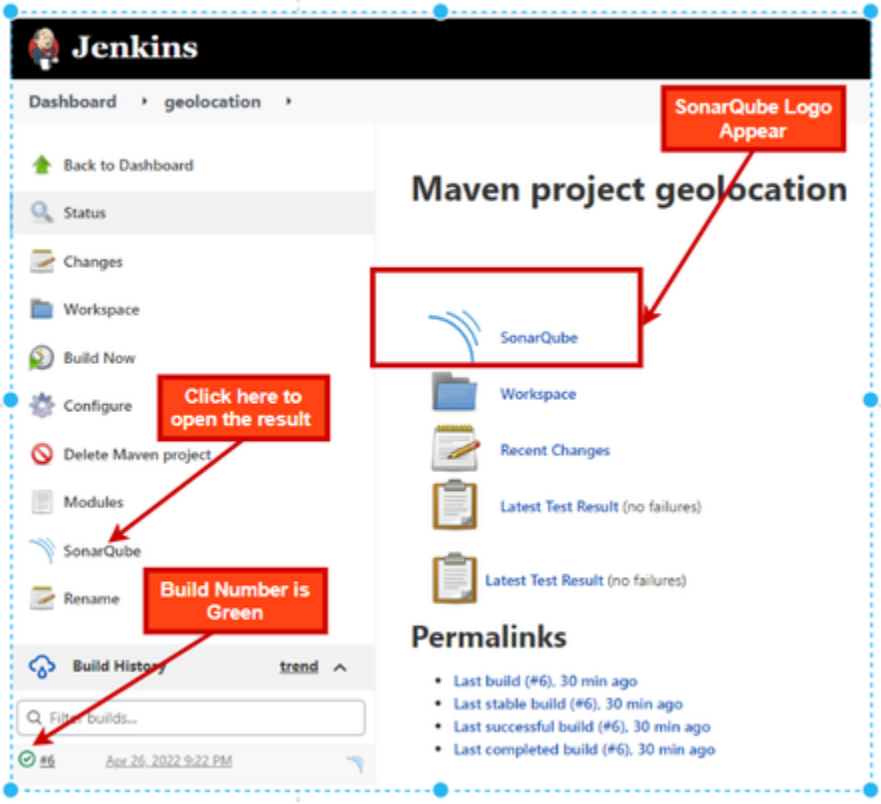
- click on Build Now to generate the build of our project. Before generating this build, the sonar analysis is performed.



- When you open the Jenkins web console, you can see the different analysis actions that are performed before the build. At one step, you have the capture below that reflects the end of the sonar analysis execution with success.

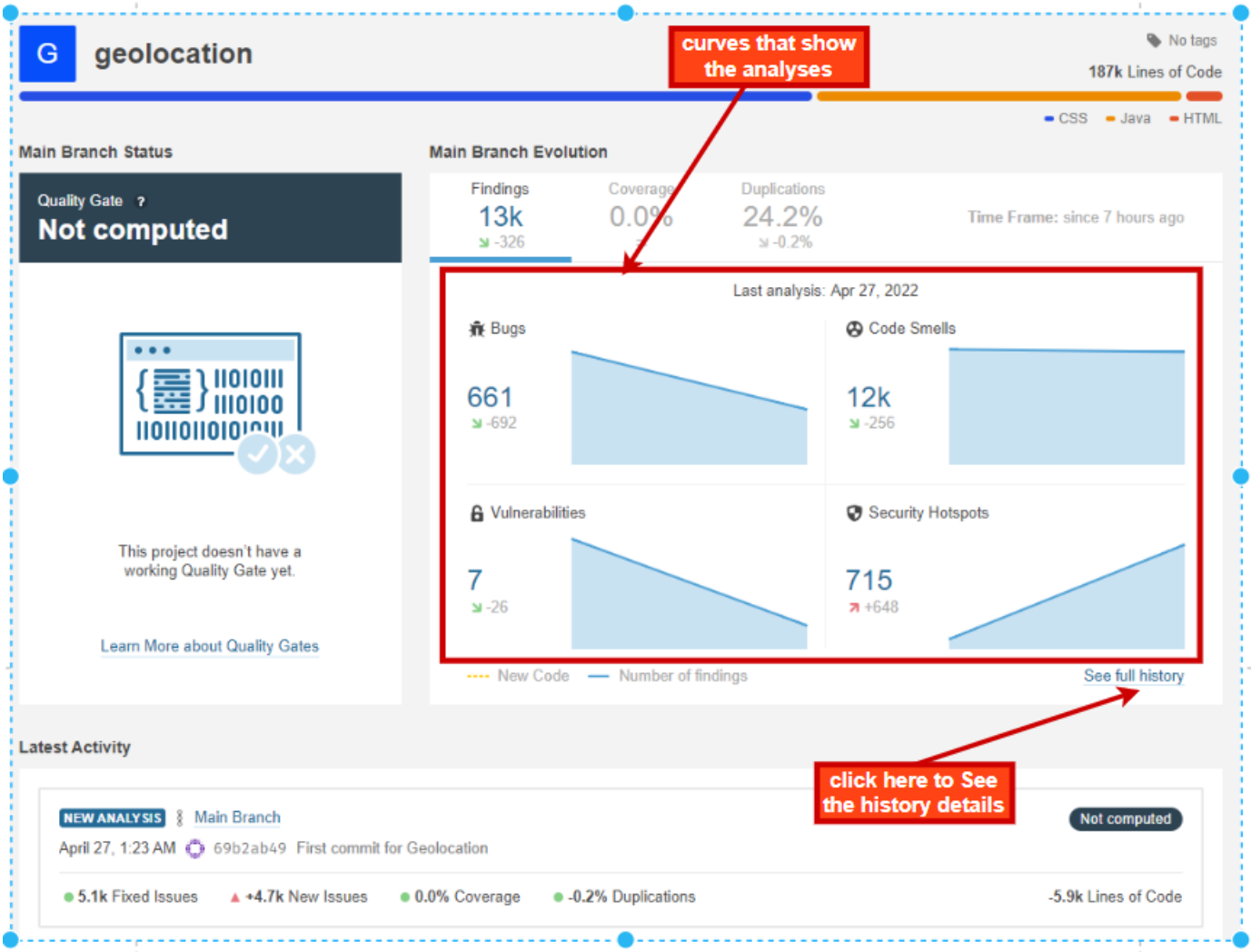


- **when everything has gone well, we notice :**
  - **the appearance of Sonar icons:** this means that the project integrates the Sonar analysis
  - the number (6 in my case) of build in green color.



- **let's display the analysis results in Sonar.**

After clicking on the Sonar icon, we are redirected to the sonar analysis page where we can see the curve of the code quality evolution.

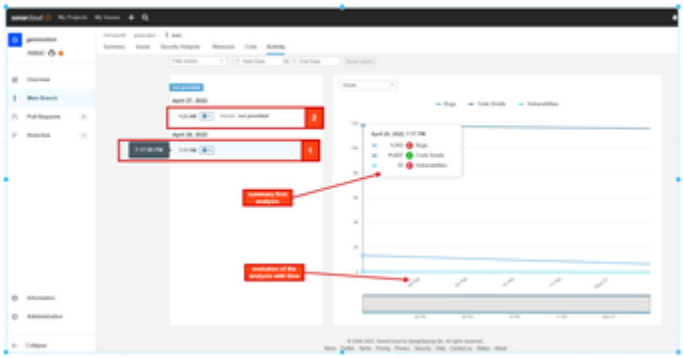


- we can see the evolution of the analysis curve in time.

here we are already at two analyses: the first one on the 26th and the second one on the 27th which correspond well to the dates so we edit this project :

1- first analysis made when creating the project in SonarCloud, done on **April 26, 2022**

2- second analysis by **Jenkins,** done on **April 27, 2022**



Note : If you click on the analysis date or approach the graph with the cursor, the summary of this analysis is displayed.

---

extra content: see how developers manually deploy the application

- Let's see how the developer deploys the application in order to proceed to tests and bug fixes.

- After that, we will go through our second mission within this project.

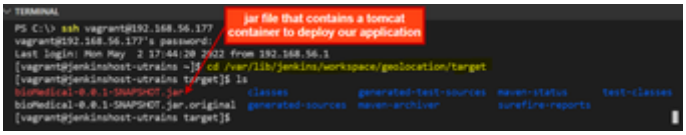
Traditional method to deploy the application:
1- After the sonar analysis, a build of the application is done and a jar file is generated. The jar file, embeds the executable codes of the application and a **Tomcat Server** that will allow to **deploy the application.**

traditional method to deploy the application:
1- After the sonar analysis, a build of the application is done and a jar file is generated. The jar file, embeds the executable codes of the application and a Tomcat Server that will allow to deploy the application.
Deployment process :

- if all went well at the sonar analysis stage, then you can deploy the application on the Jenkins server (not recommended) to see what the application looks like.
  1- connect to the Jenkins server
  2- open the directory that was created after the sonar analysis (all the projects are in the **/var/lib/jenkins/workspace/** directory)

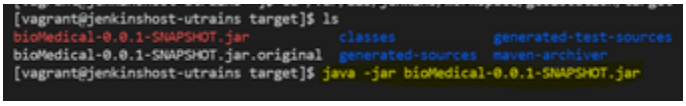
```
cd /var/lib/jenkins/workspace/geolocation/target
ls
```

  we can see our Jar files



- use the command **java -jar name of the jar file** to start the application. if you don't have enough memory, stop the Jenkins server before doing so. because in practice, the application has to be deployed on a different server where java and maven are installed.

```
java -jar bioMedical-0.0.1-SNAPSHOT.jar
```

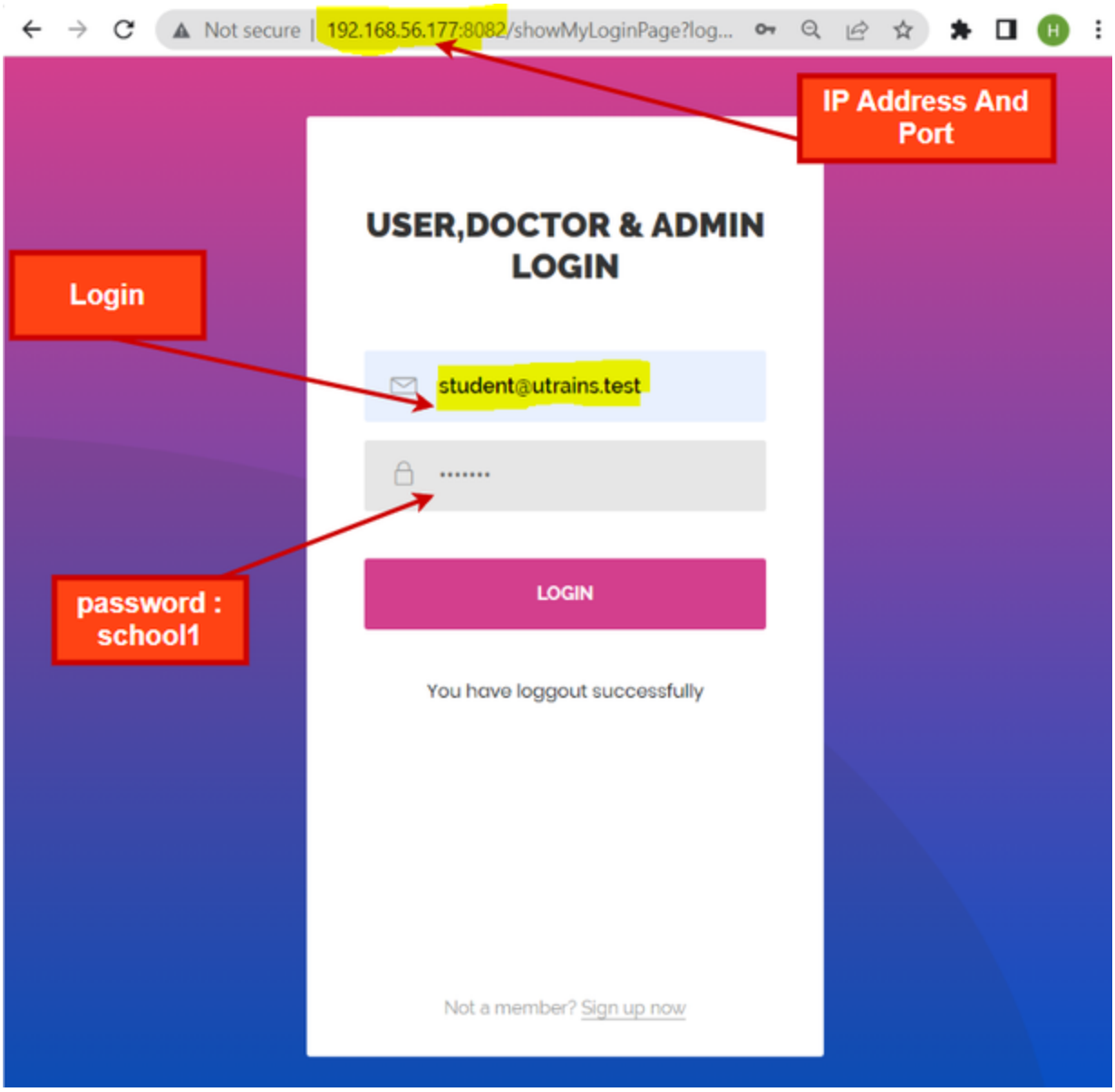


- start the java application

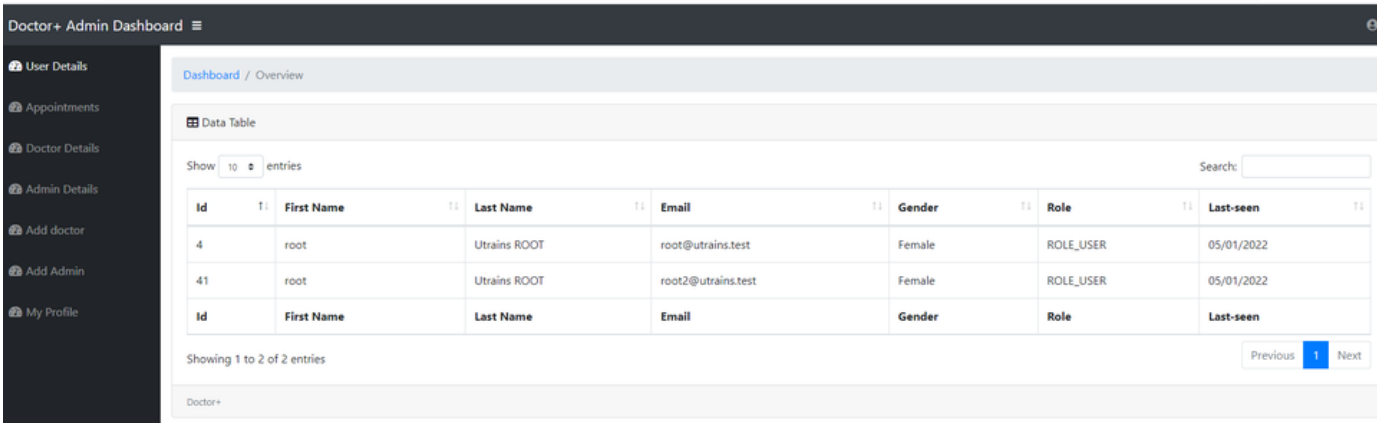
- open the browser, then enter the IP address of the server, followed by the port to open the application

192.168.56.177:8080

- login page



- **App Admin Dashboard**



- log out from the administration page, then log in with a user profile **(email: root@utrains.test password: root_pass)**.
- here we can see that the page that the users see is different, this is the role management that our **Dev** friends have already implemented in this application.
- main page for simple Patients

- admin Details and Database

this application uses a **MySql** database. here, the developers have created a database for the tests which is stored in the application's memory and it is this database that displays the data.