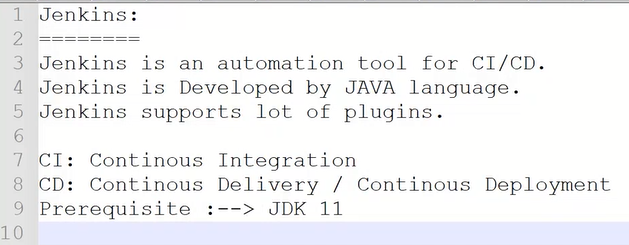
Jenkins

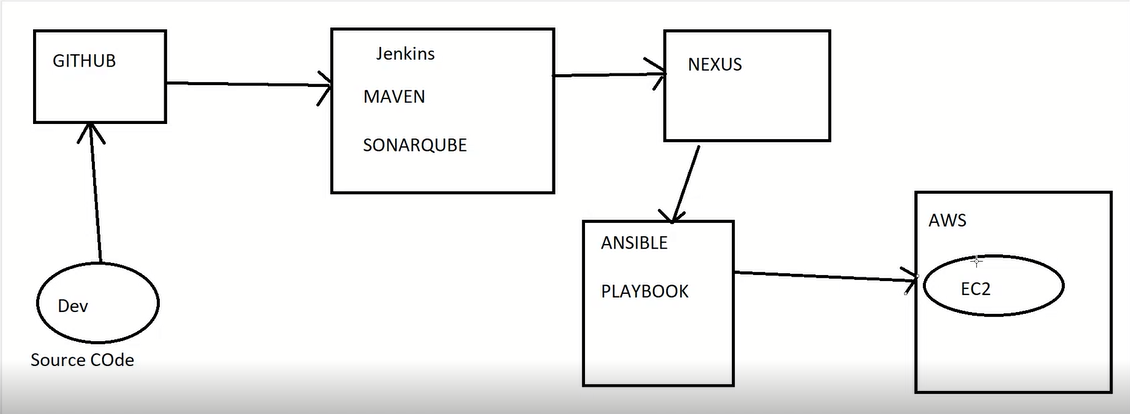
What is Jenkins?

Jenkins is an open source Continuous Integration server capable of orchestrating a chain of actions that help to achieve the Continuous Integration process (and not only) in an automated fashion. Jenkins is free and is entirely written in Java.

Jenkins is a widely used application around the world that has around 300k installations and growing day by day. It is a server-based application and requires a web server like Apache Tomcat. The reason Jenkins became so popular is that of its monitoring of repeated tasks which arise during the development of a project.

For example, if your team is developing a project, Jenkins will continuously test your project builds and show you the errors in early stages of your development. By using Jenkins, software companies can accelerate their software development process, as Jenkins can automate build and test at a rapid rate. Jenkins supports the complete development lifecycle of software from building, testing, documenting the software, deploying and other stages of a software development lifecycle.





Prerequisite to Install jenkins is java.

To install Java

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

Yum install java 11

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

Start the Jenkins service:

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

systemctl start Jenkins

Enable the Jenkins service:

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

systemctl enable jenkins

now copy the password for admin from home directory under /var/lib/jenkins/secrets/initialAdminPassword

Important

1) Default port : 8080

2) Jenkins home directory : /var/lib/jenkins

3) Jenkins port config file : /etc/sysconfig/jenkins

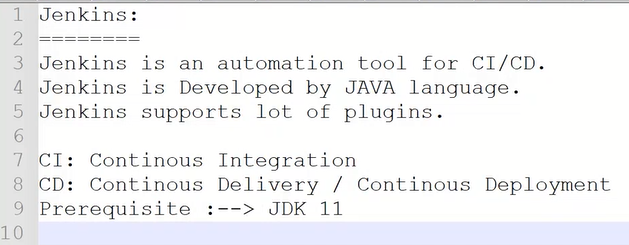
4) Jenkins secrets key : /var/lib/jenkins/secrets/initialAdminPassword

5) Jenkins Task logs : /var/lib/jenkins/logs

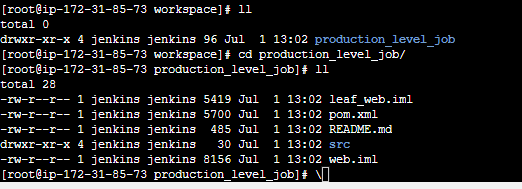
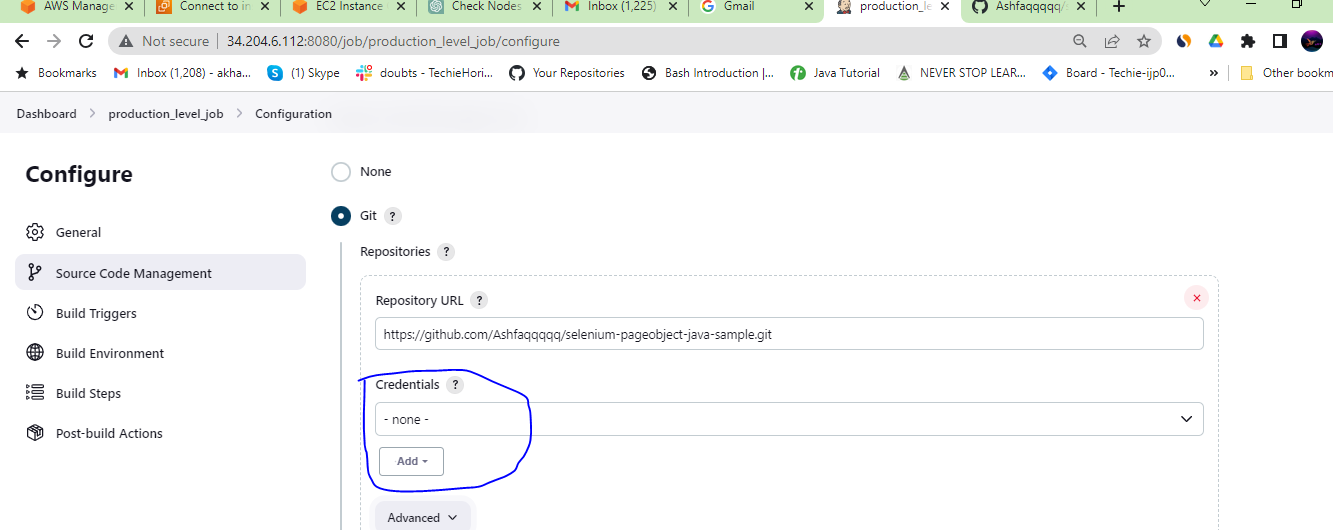
6) Jenkins app logs: /var/log/jenkins/jenkins.log

7) Jenkins jobs: /var/lib/jenkins/jobs

8) Jenkins workspace: /var/lib/jenkins/workspace



Note: packages need to install based on the requirement in Jenkins, like git, java, we need to install git in Jenkins server (ec2 machine) inorder to access source code from git hub repository.

* Yum install git –y -------> to install git in Jenkins server,
* Whatever we created jobs and build in Jenkins it will be stored in workspace folder in Jenkins (home directory : /var/lib/Jenkins/workspace)
* 
* While configuring jobs in Jenkins, make sure provide git repository to access source code which have pom.xml file, provide username to provide access to private repo.
* 

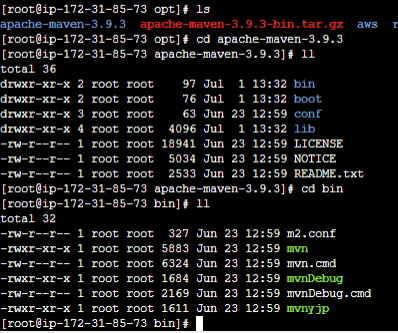
To build the code

We need to install maven in Jenkins servers

Download maven from website

Wget <https://dlcdn.apache.org/maven/maven-3/3.9.3/binaries/apache-maven-3.9.3-bin.tar.gz>

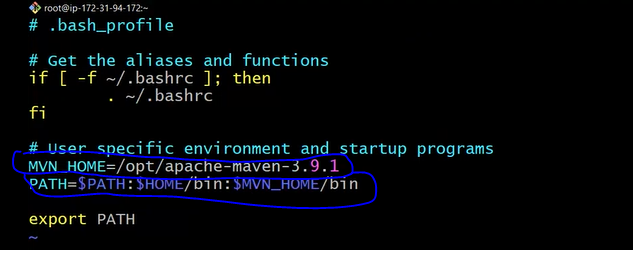
Untar the file using tar command : tar –xvf <filename>



Goto /apache-maven-3.9.3/bin/mvn/

We need to add global variables , for that # vi .bash\_profile

Edit file, add highlighted, what we have given name in Jenkins gui





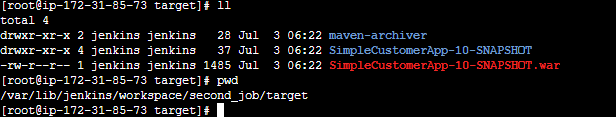
To restart file , source ~/.bash\_profile

To check the path has updated in bash file, use echo $maven (name can be vary depend upon what we have mentioned tools of Jenkins.



It will show location of maven downloaded as show above

After build the job in Jenkins , .war file created in the Jenkins server in under workspace/jobname/target/ location.



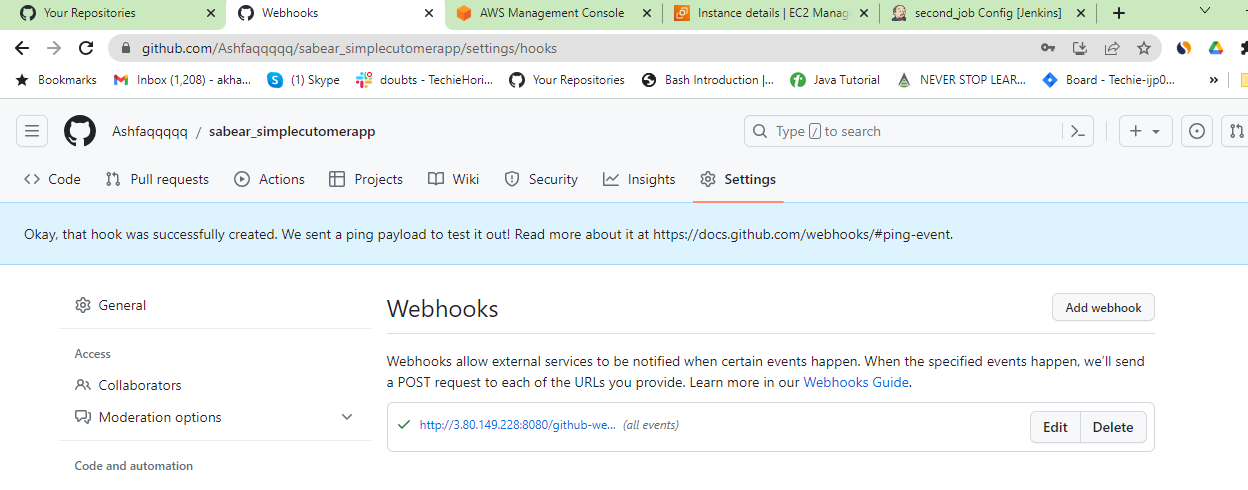
Webhooks: to automate the process or pull the changes into the jenkins. We need to configure the webhooks.

Note: check the configure the java in Jenkins or not!!

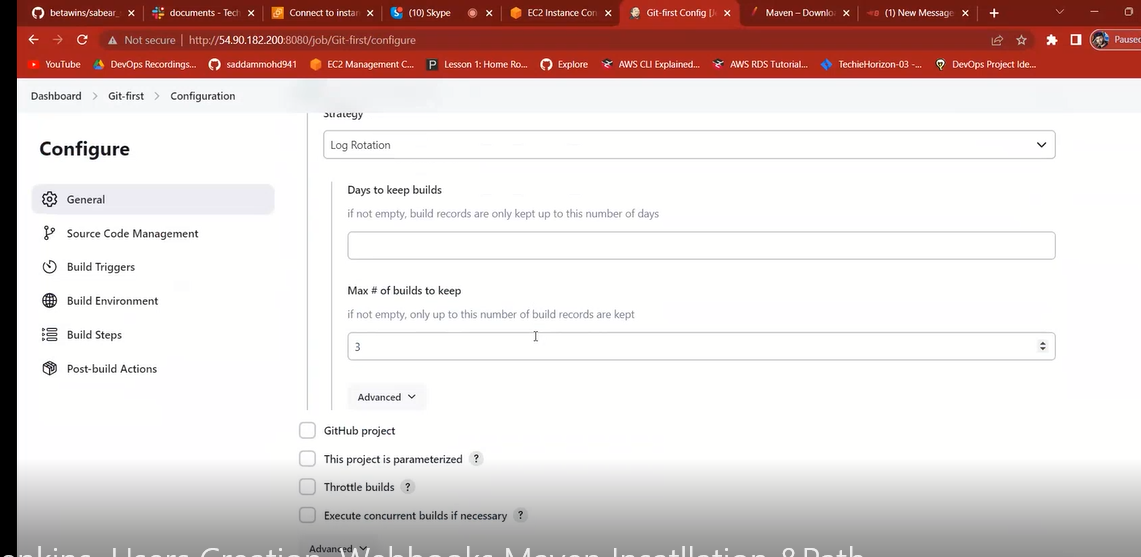
Webhooks:

Add url to the webhooks of respository, <http://3.80.149.228:8080/github-webhook/>

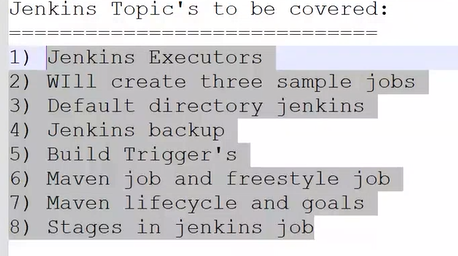
, select everything and update.



Inorder to limit the jobs log we can add or remove the old builds in configuration settings by enabling the option.



3rd video class

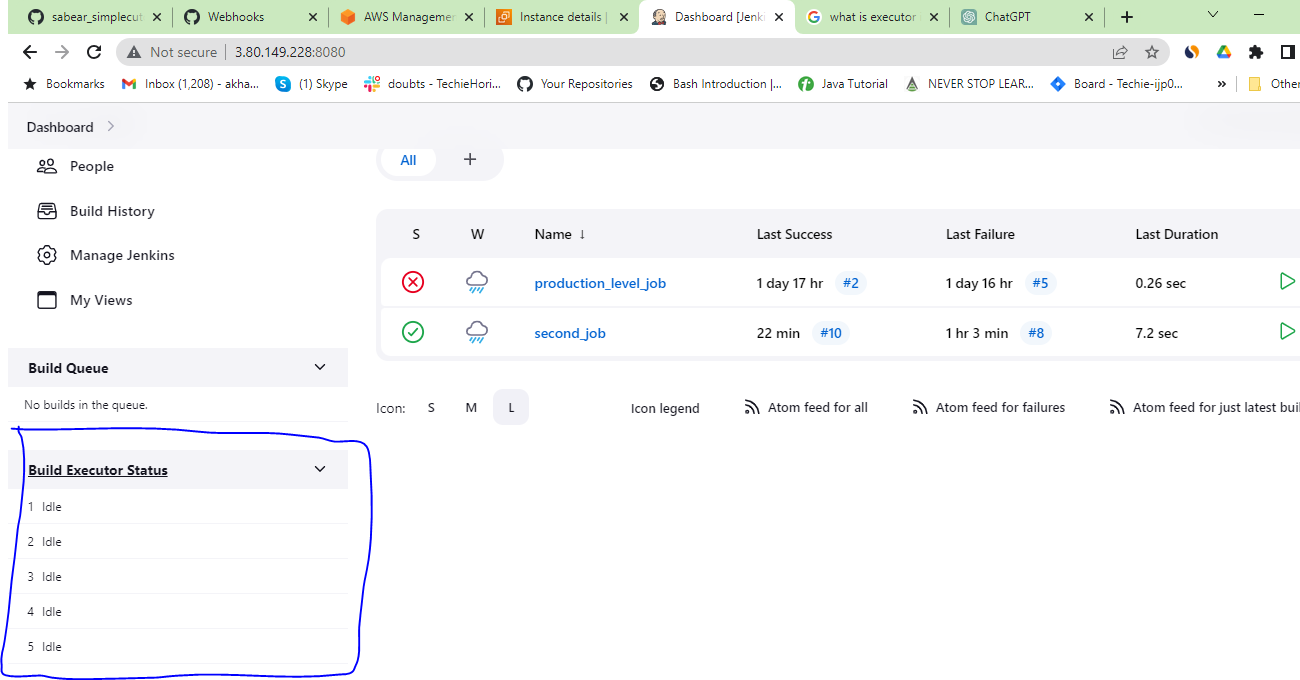


1. Jenkins executors:

In Jenkins, executors refer to the individual worker processes or threads that are responsible for executing build jobs and performing other tasks. Jenkins is a popular open-source automation server that allows you to automate various aspects of software development, including building, testing, and deploying applications.

When you create a Jenkins job, it gets assigned to an executor for execution. Executors run concurrently and can execute multiple jobs simultaneously, depending on the configuration and available system resources. Each executor is essentially a separate execution environment that can be configured with specific resources, such as CPU, memory, and environment variables.

The number of executors available in Jenkins depends on the configuration of your Jenkins instance and the resources of the host machine. You can configure the number of executors either globally for the Jenkins instance or on a per-node basis if you have a distributed Jenkins setup with multiple worker nodes.



We can avoid typing password by editing file of config.xml in server,

#vi config.xml

Make it false at security password.

Then save it.

Jenkins Backup: are two types

1. Use thin backup, like with bash script
2. Manually with scp command to copy the Jenkins files (workspace,jobs)

Manually,

Tar the Jenkins folder and create backup, use scp command to copy the file in new Jenkins server.

Note:in jobs,

**Build periodically options**

if we want to build the job every minute or to automate the process , in job, configure setting add build periodically options Like \*\*\*\*\* or we can add particular time to automate the process.

**Poll SCM**

If we provide same chron job as above, if we enable this option then it will look to the repository every time whether any changes any made and then execute.

Plugins:

Install maven integration in Jenkins to ease the jobs execution…

Maven integration in Jenkins allows you to automate the build and deployment processes of Maven-based projects. Maven is a popular build automation tool primarily used for Java projects. Jenkins, on the other hand, is a continuous integration and continuous delivery (CI/CD) tool.

When you configure Maven integration in Jenkins, you can set up Jenkins jobs that perform various tasks related to your Maven project, such as:

1. Building: Jenkins can automatically trigger builds for your Maven projects whenever changes are pushed to the source code repository. It can fetch the latest code, compile it, run tests, and generate build artifacts (e.g., JAR, WAR files).
2. Dependency Management: Maven has a powerful dependency management system. Jenkins can fetch the required dependencies specified in your project's Maven configuration file (pom.xml) and ensure that they are available during the build process.
3. Testing: Jenkins can execute Maven goals to run automated tests for your project. This allows you to incorporate test suites, such as JUnit or TestNG, into your build pipeline and receive test reports.
4. Code Analysis: Jenkins can integrate with various code analysis tools, such as SonarQube or Checkstyle, to analyze your Maven project's code quality, identify bugs, and enforce coding standards. It can generate reports or trigger build failures based on the analysis results.
5. Deployment: Jenkins can automate the deployment of your Maven-built artifacts to different environments, such as development, staging, or production. It can copy the generated artifacts to specific locations, execute deployment scripts, or trigger deployment pipelines.
6. Release Management: Maven provides release management capabilities through plugins like the Maven Release Plugin. Jenkins can invoke these plugins to automate the process of releasing new versions of your Maven project, including versioning, tagging in version control, and updating project metadata.

Overall, Maven integration in Jenkins simplifies the setup and management of your Maven-based build and deployment pipelines. It helps streamline the software development process, ensuring consistent builds, automated testing, and efficient release management.

In Jenkins we can create group also.

In Jenkins what “clean install” do , will clean existing directory whenever we build a job,

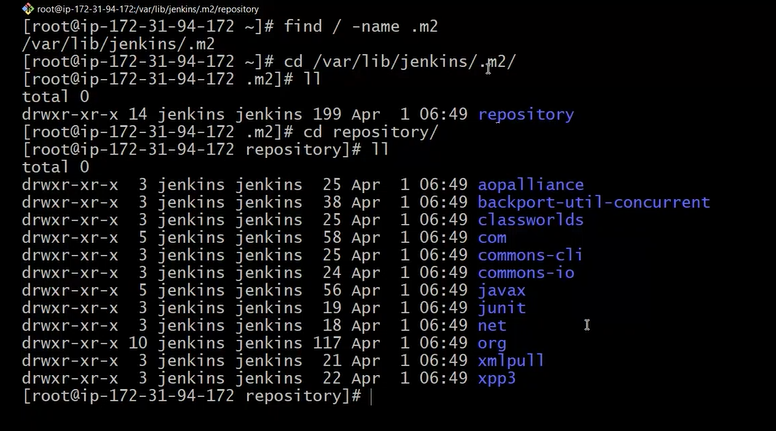
1. how many repo for maven to download dependencies? explain

Maven can download dependencies from multiple repositories. By default, Maven uses the Maven Central Repository (https://repo.maven.apache.org/maven2/) as its primary repository for downloading dependencies. Maven Central is a public repository that hosts a vast collection of open-source libraries and artifacts.

In addition to the Maven Central Repository, Maven supports the concept of repository groups and allows you to configure multiple repositories to search for dependencies. When Maven needs to resolve a dependency, it sequentially checks each configured repository in the order they are specified until it finds the required dependency.

Here are the different types of repositories that Maven can use to download dependencies:

1. Local Repository: This is the local cache on your machine where Maven stores the downloaded dependencies. It is typically located in the `.m2` directory in your user's home directory. When Maven encounters a dependency, it first checks the local repository to see if it is already available. If not, it proceeds to download it from other repositories.



2. Central Repository: As mentioned earlier, Maven Central Repository is the default repository used by Maven. It hosts a vast collection of open-source libraries and artifacts that are widely used in the Java ecosystem. Maven Central Repository is a reliable and comprehensive source for most common dependencies.

3. Remote Repositories: Apart from the Central Repository, you can configure additional remote repositories to search for dependencies. These repositories can be private or public and may contain specialized or proprietary libraries. You specify the remote repositories in the `<repositories>` section of your project's `pom.xml` file, along with their URLs and other configuration details.

4. Repository Groups: Maven allows you to create a repository group, also known as a repository aggregator, which acts as a logical collection of multiple repositories. A repository group combines multiple repositories into a single virtual repository, allowing Maven to search for dependencies in all the included repositories simultaneously. This provides flexibility and the ability to use multiple repositories as if they were a single repository.

You can configure repository groups and remote repositories in your `pom.xml` or in your Maven settings file (`settings.xml`), allowing you to define specific repositories for your project or globally across your development environment.

By leveraging these different repository options, Maven provides flexibility in resolving dependencies, allowing you to download dependencies from multiple sources based on your project's requirements and constraints.

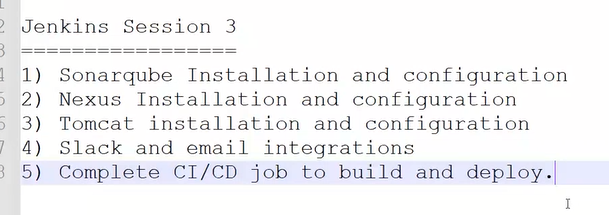
Check in website for maven details about life cycles,pom…..

In order make continuous integration, there stages need to implemented in environment.

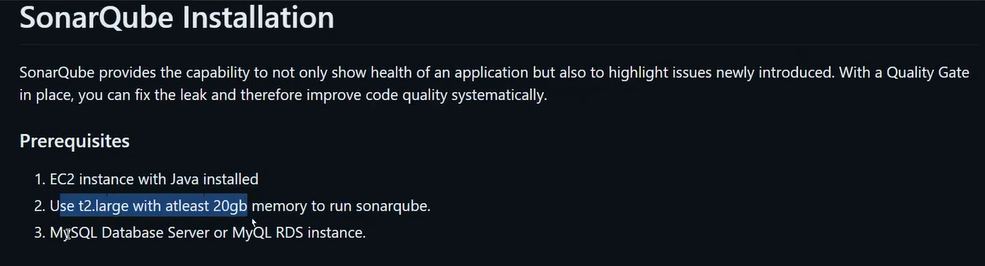
Different stages in Jenkins

1. Gitclone –to download the source code
2. Maven build- to compile the source code and create package for us.
3. Sonarqube to check the quality of code
4. Nexus artifactory-storing the artifactory in the central location
5. Slack integration- informing the teams for internal communication
6. Email integration

Class-04



To install sonarqube in another server, java and Jenkins is required



Java and Jenkins are not prerequisites for installing SonarQube. However, Java is a prerequisite for running SonarQube, as it is a Java-based application.

1. Install Java: Ensure that you have Java Development Kit (JDK) installed on your system. SonarQube requires Java 11 or later versions. You can download and install the appropriate JDK for your operating system from the official Oracle Java website or use OpenJDK.

2. Download SonarQube: Go to the official SonarQube website (https://www.sonarqube.org/downloads/) and download the latest version of SonarQube Community Edition or SonarQube Developer Edition.

3. Extract SonarQube: Once the download is complete, extract the SonarQube package to a directory on your system.

4. Configure SonarQube: Navigate to the SonarQube installation directory and configure the `sonar.properties` file according to your environment requirements. This file contains various configuration settings such as database connection details, server port, and authentication settings.

5. Start SonarQube: Run the startup script for SonarQube, which is typically called `sonar.sh` (Unix) or `StartSonar.bat` (Windows). This will start the SonarQube server.

6. Access SonarQube Web Interface: Open a web browser and access the SonarQube web interface using the URL `http://localhost:9000` or the appropriate server address and port you configured in the `sonar.properties` file.

Note that Jenkins can be integrated with SonarQube to automate code quality analysis as part of your CI/CD pipeline. However, Jenkins itself is not a prerequisite for installing SonarQube. You can install and use SonarQube independently for code quality management.

Make mysql as database to store logs

**Procedure**

Install java 1.8 version

yum install java-1.8\*

**Add mysql rpm Repository**

yum update

sudo wget https://dev.mysql.com/get/mysql57-community-release-el7-11.noarch.rpm

sudo yum localinstall mysql57-community-release-el7-11.noarch.rpm

rpm --import https://repo.mysql.com/RPM-GPG-KEY-mysql-2022

sudo yum install mysql-community-server

sudo systemctl start mysqld.service

**Start MySQL and Enable Start at Boot Time**

systemctl start mysqld

systemctl enable mysqld

```

**Check if mysql is running or not**

netstat -na | grep 3306

```

**Configure the MySQL Root Password**

**You will see default MySQL root password**

grep 'temporary' /var/log/mysqld.log

```

Login to mysql using the default password

mysql -u root -p

**Now replace the default password with a new and strong password**

ALTER USER 'root'@'localhost' IDENTIFIED BY 'Admin@123';

flush privileges;

**Test Using new password**

mysql -u root -p

**Download stable SonarQube version from below website.**

- Website: https://www.sonarqube.org/downloads/

- Note: This Article written for SonarQube6.0

**Download & unzip SonarQube 6.0**

# cd /opt

# wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-6.6.zip

# unzip sonarqube-6.6.zip

# mv /opt/sonarqube-6.6 /opt/sonar

After installing sonarqube, login to database add two users (remote and local) and database

Login to mysql

```sh

mysql -u root -p

```

Create a new sonar database

```sh

CREATE DATABASE sonar CHARACTER SET utf8 COLLATE utf8\_general\_ci;

```

Create a local and a remote user

```sh

CREATE USER sonar@localhost IDENTIFIED BY 'Sonar@123';

CREATE USER sonar@'%' IDENTIFIED BY 'Sonar@123';

```

Grant database access permissions to users

```sh

GRANT ALL ON sonar.\* TO sonar@localhost;

GRANT ALL ON sonar.\* TO sonar@'%';

```

check users and databases

```sh

show databases;

SELECT User FROM mysql.user;

FLUSH PRIVILEGES;

QUIT

```

So for you have configured required database information on mysql. Let’s Jump back to your EC2 instance and enable SonarQube properties file to connect his Database.

**### ON EC2 Instance**

Edit sonar properties file to uncomment and provide required information for below properties.

- File Name: /opt/sonar/conf/sonar.properties

  - sonar.jdbc.username=`sonar`

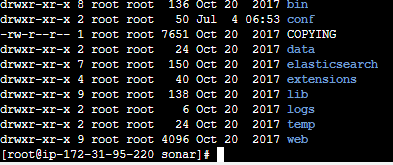
  - sonar.jdbc.password=`Sonar@123`

  - sonar.jdbc.url=jdbc:mysql://`localhost:3306`/sonar?useUnicode=true&characterEncoding=utf8&rewriteBatchedStatements=true&useConfigs=maxPerformance&useSSL=false

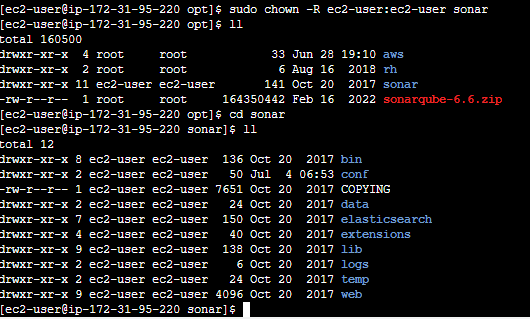
  - sonar.web.host=`0.0.0.0`

  - sonar.web.context=`/sonar`

After this , all the files are in root permissions, inorder to access by ec2-user, change permissions as below mentioned.



Changed permission to ec2-user to root



# Sonar version 7 and higher cannot be run using root user,please switch to any other user and change the permissions to sonar directory and start sonar.

Create new user

```sh

# useradd sabear

# passwd sabear

```

Add user sabear to sudoers file

```sh

vi /etc/sudoers

## Same thing without a password

# %wheel ALL=(ALL) NOPASSWD: ALL

sabear ALL=(ALL) NOPASSWD: ALL

```

Restart the sshd service

```sh

# systemctl restart sshd

```

Switch to newly created user

```sh

# sudo su sabear

Or

**just got the file in bin and check version is running , start that file inorder to start**

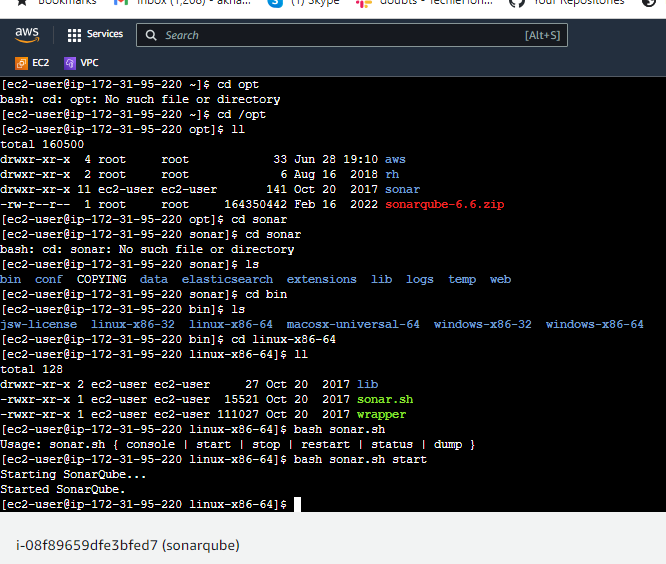
Start SonarQube service

```sh

# cd /opt/sonar/bin/linux-x86-64/

# ./sonar.sh start

```



To check the sonarqube is running, use bash sonar.sh start



##### Run SonarQube as a default service

Below red content complete on requirement.

Implement SonarQube server as a service

Copy sonar.sh to etc/init.d/sonar and modify it according to your platform.

# sudo cp /opt/sonar/bin/linux-x86-64/sonar.sh /etc/init.d/sonar

# sudo vi /etc/init.d/sonar

```

Add below values to your /etc/init.d/sonar file

Insert/modify below values

SONAR\_HOME=/opt/sonar

PLATFORM=linux-x86-64

WRAPPER\_CMD="${SONAR\_HOME}/bin/${PLATFORM}/wrapper"

WRAPPER\_CONF="${SONAR\_HOME}/conf/wrapper.conf"

PIDDIR="/var/run"

```

Start SonarQube server

```sh

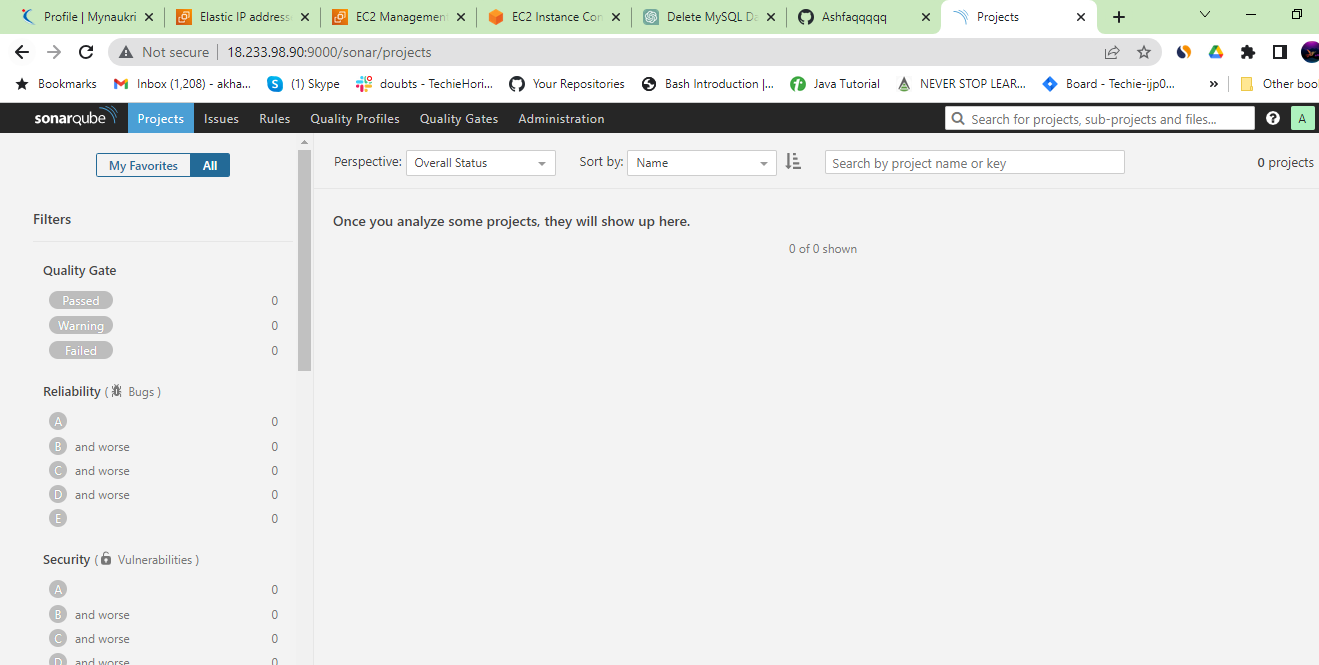
# service sonar start

```

SonarQube application uses port 9000. access SonarQube from browser

```sh

http://<EC2\_PUBLIC\_IP>:9000/sonar



```

Default credentials for sonarqube.

Username : admin

Password: admin

Token is creation when anonymous access is disabled

Token Is created : 1234: **c6a8a5f1a12aafd2da94130c8984bb76343864bc**

**Copy to maven**

mvn sonar:sonar \

-Dsonar.host.url=http://18.233.98.90:9000/sonar \

-Dsonar.login=c6a8a5f1a12aafd2da94130c8984bb76343864bc

### NOTES

1) Check whether you enabled port 9000 in EC2 instance security group

### Important Points:

1) mysql port number : 3306

2) sonarqube port: 9000

3) sonarqube logs : /opt/sonar/logs

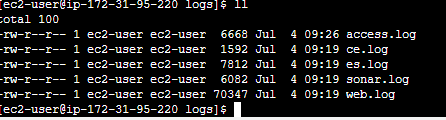
4) We can find four different logs

a)access.log

b)es.log aka elastic search

c)sonar.log

d)web.log



### Videos for reference:

### sonarqube installation:

https://www.youtube.com/watch?v=zRQrcAi9UdU

#### mysql rds instance aws:

https://www.youtube.com/watch?v=vLaW6b441x0

after installing and ping sonar qube , install sonarqube scanner in Jenkins

goto config

goto plugins

search sonarqube, install without restart.

What is sonarcube scanner?

SonarQube Scanner is a tool used in the Jenkins CI/CD pipeline to perform static code analysis on software projects. It integrates with SonarQube, which is a popular open-source platform for continuous code quality inspection.

When SonarQube Scanner is used in Jenkins, it analyzes the source code of a project and provides detailed reports on various aspects of code quality, including code duplication, code smells, potential bugs, security vulnerabilities, and more. It helps developers and teams identify areas of improvement, maintain code quality standards, and ensure that best practices are followed.

Here's how SonarQube Scanner works within Jenkins:

Integration: The SonarQube Scanner is configured as part of the Jenkins build process. It can be added as a build step or as a post-build action, depending on the specific Jenkins configuration.

Analysis Configuration: The SonarQube Scanner is configured with the necessary settings, such as the SonarQube server URL, authentication tokens, project key, and any additional analysis parameters.

Code Analysis: During the build process, SonarQube Scanner collects the source code files and sends them to the SonarQube server for analysis. The SonarQube server applies a wide range of rules and metrics to evaluate the code quality.

Analysis Results: Once the analysis is complete, SonarQube provides detailed reports, metrics, and visualizations through the SonarQube user interface. These reports help identify code quality issues, potential bugs, security vulnerabilities, and maintainability concerns.

Quality Gate: SonarQube allows the definition of quality gates, which specify thresholds for code quality metrics. The SonarQube Scanner can fail the Jenkins build if the code doesn't meet the defined quality gate criteria, ensuring that only code of acceptable quality is deployed.

By integrating SonarQube Scanner into Jenkins, developers and teams can automate the code quality analysis process and get immediate feedback on the health and maintainability of their codebase. It promotes continuous code quality improvement and helps identify and resolve issues early in the development lifecycle.

Install sonarqube scanner in Jenkins

wget <https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-4.6.2.2472-linux.zip>

unzip the file and rename

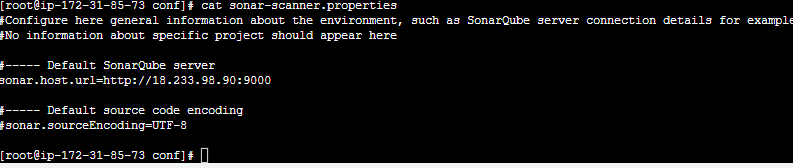
unzip sonar-scanner-cli-4.6.2.2472-linux.zip

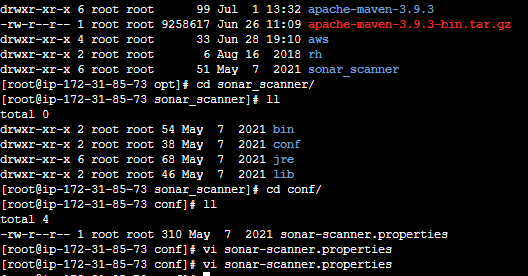
mv sonar-scanner-4.6.2.2472-linux /opt/sonar\_scanner

**Set SonarQube server details in sonar-scanner property file**

- Sonar properties file: /opt/sonar\_scanner/conf/sonar-scanner.properties

- sonar.host.url=http://`<SONAR\_SERVER\_IP>`:9000





Login to Jenkins GUI console and install " SonarQube scanner" plugin

- `Manage Jenkins` > `Manage Plugins` > `Avalable` > `SonarQube scanner`

Configure SonarQube scanner home path

- `Manage Jenkins` > `Global Tool Configuration` > `SonarQube Scanner`

- Name : `sonar\_scanner`

- SONAR\_RUNNER\_HOME : `/opt/sonar\_scanner`

Configure SonarQube server name and authentication token

- `Manage Jenkins` > `Configure Systems` > `SonarQube Servers`

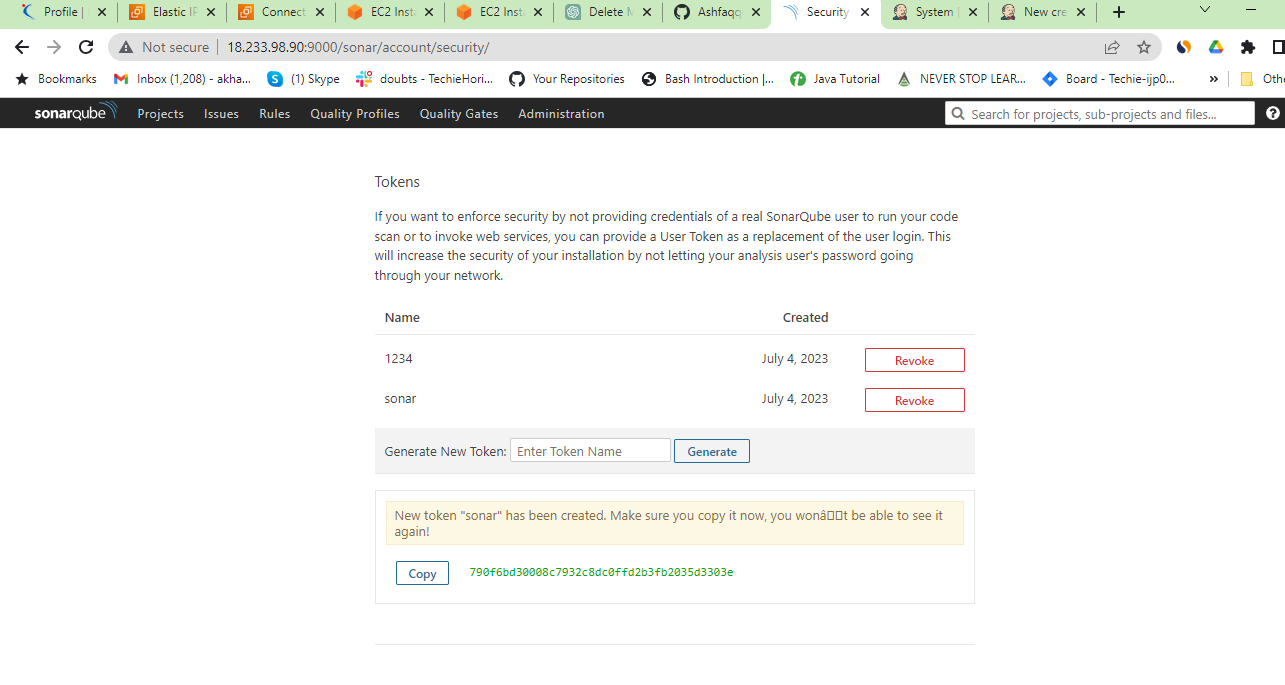
- Name : `SonarQube`

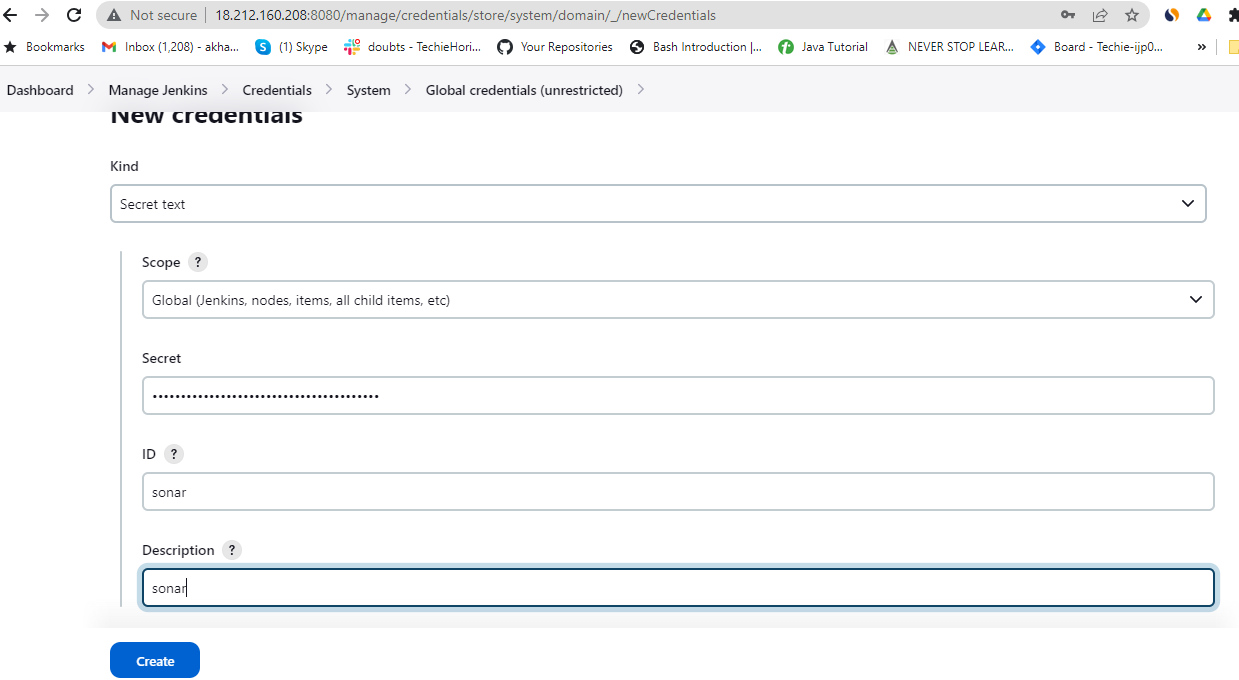
- ServerURL : `http://<Sonarqube\_server>:9000/sonar`

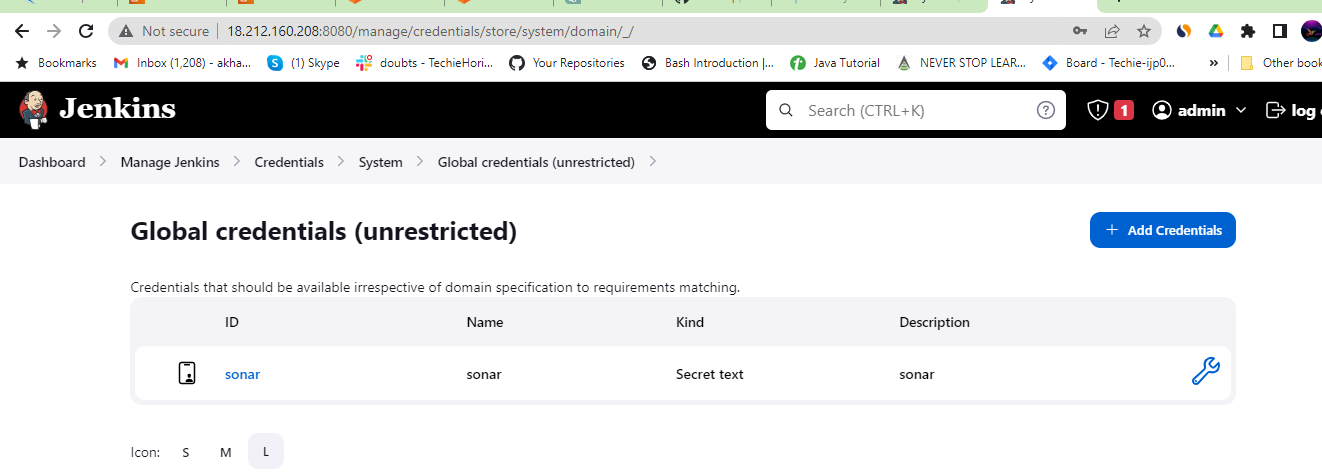
- Server `authentication token`

To Get Authentication code follow below steps.

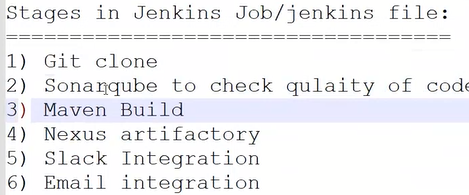
Login to SonarQube server as a admin `My Account` > `Security` > `Generate Token`



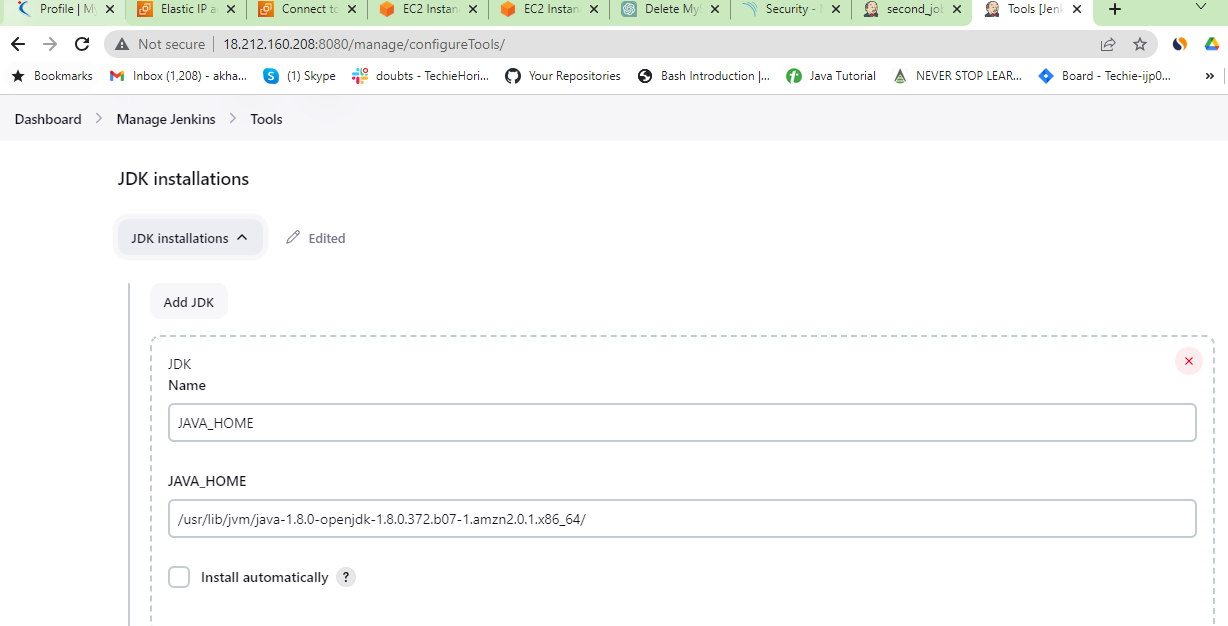




Goto config of sonarqube sever in Jenkins, add credentials as sonar



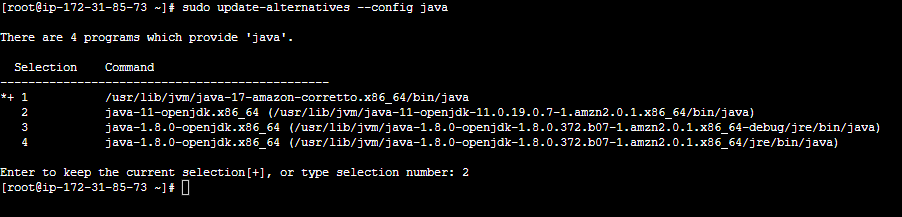
See java 1.8\* is installed in Jenkins server, for that check whether java 11 must be configured first.

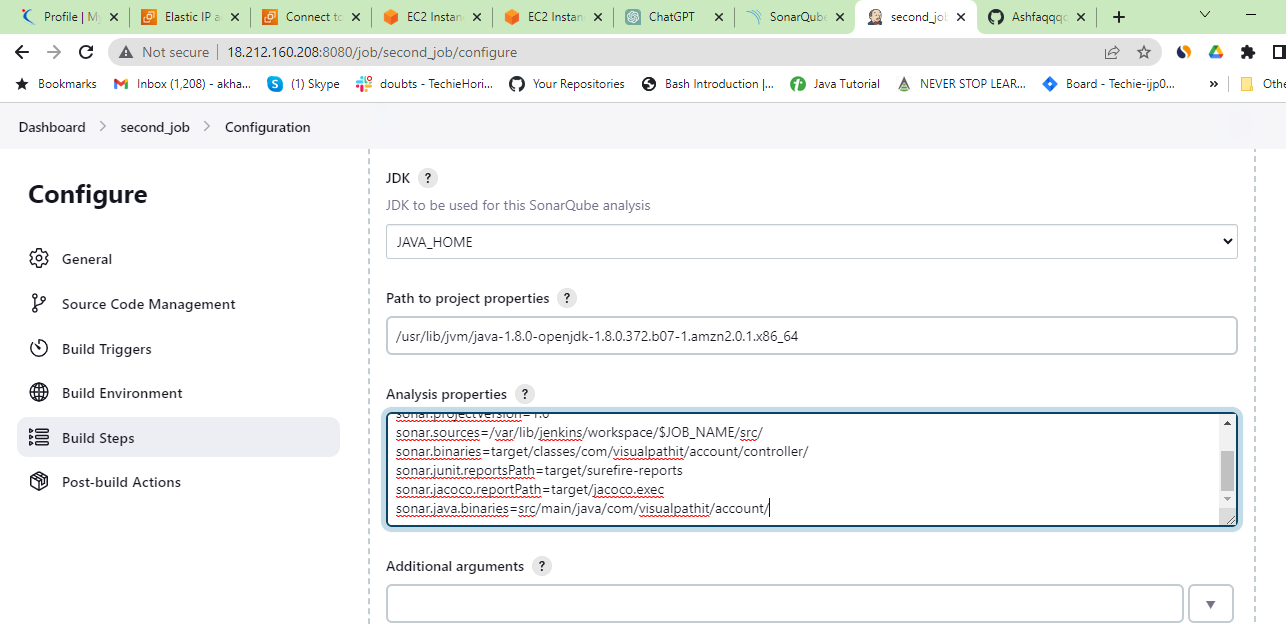


Here java version 1.8\* used by Jenkins job, java 11 used by Jenkins.

Inorder to update in Jenkins server which java version shall use by java server so

Sudo update-alternatives - -config java





Add below, in Jenkins jdk

sonar.projectKey=Ncodeit

sonar.projectName=Ncodeit

sonar.projectVersion=2.0

sonar.sources=/var/lib/jenkins/workspace/$JOB\_NAME/src/

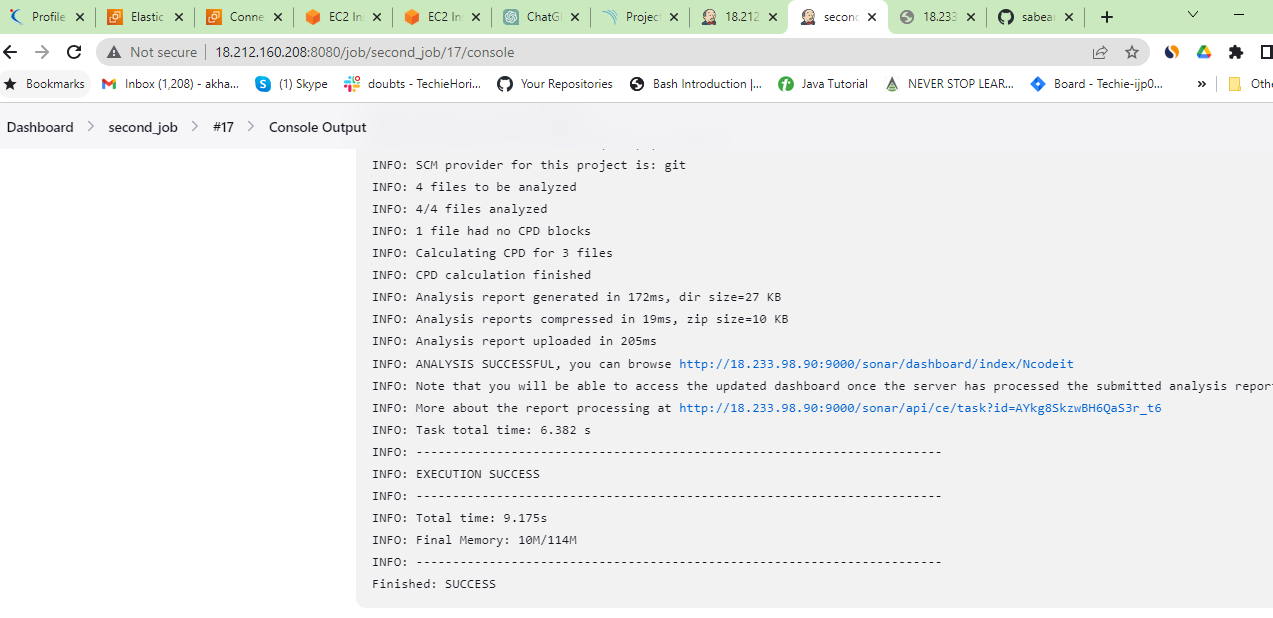
sonar.binaries=target/classes/com/visualpathit/account/controller/

sonar.junit.reportsPath=target/surefire-reports

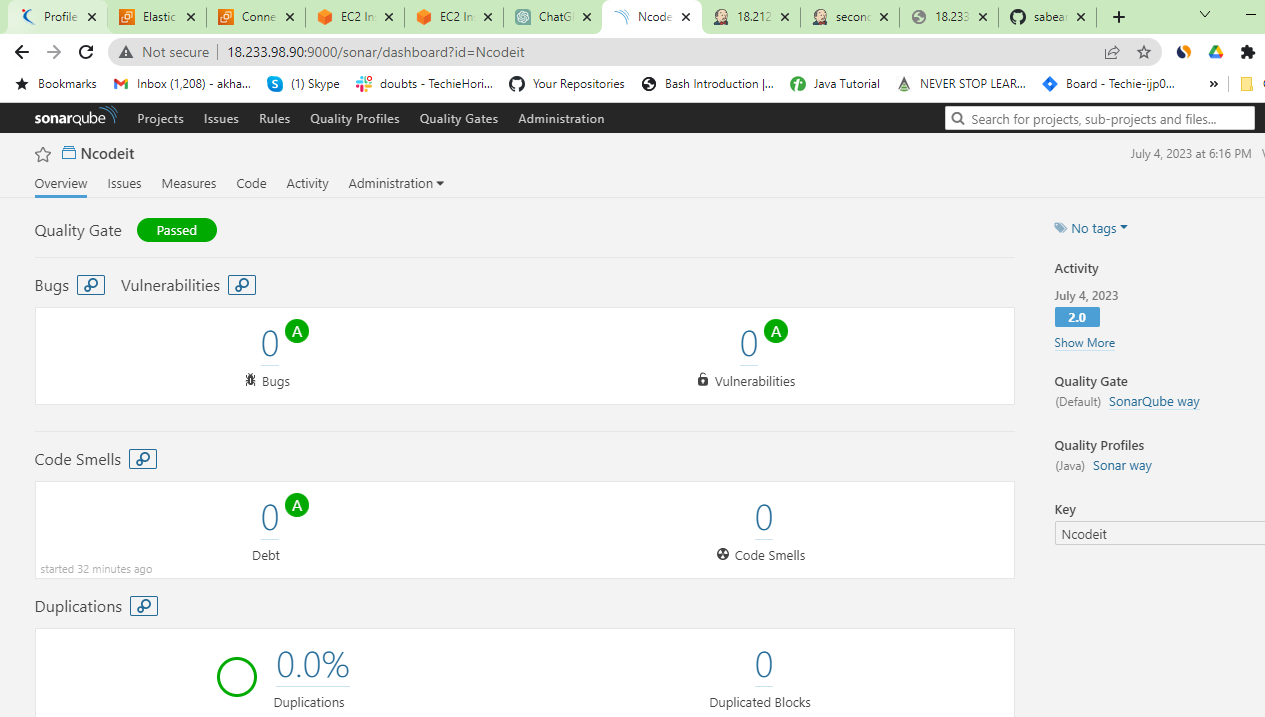
sonar.jacoco.reportPath=target/jacoco.exec

sonar.java.binaries=src/com/room/sample

now build the job, it will success.

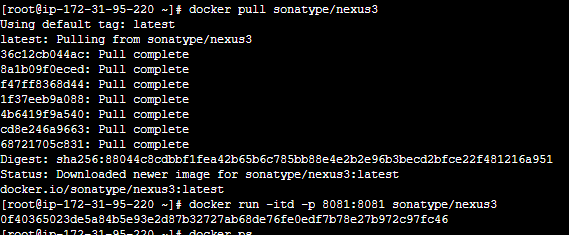


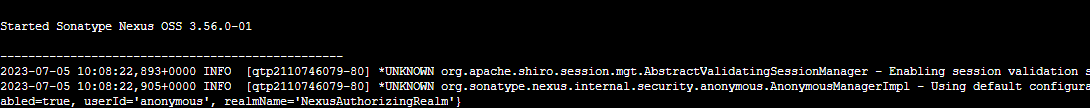
Now we will check on sonarqube projects



Nexus installation:

Installing nexus with traditional way or with docker images.





Ping with publicip:8081



Login in to the container

Docker exec –it containerid /bin/bash

Cat the password in and paste to login,

Create repo, select maven 2, select release, select redeploy and create

# Nexus Installation

Nexus is one a artifact repository which helps to store your build outcomes.

### Follow this article in \*\*[Youtube](https://www.youtube.com/watch?v=83AGz9huJGo)

### Prerequisites

1. EC2 instance with java

### Implementation steps

Download and setup nexus stable version

```sh

cd /opt

wget https://sonatype-download.global.ssl.fastly.net/nexus/3/nexus-3.0.2-02-unix.tar.gz

tar -zxvf nexus-3.0.2-02-unix.tar.gz

mv /opt/nexus-3.0.2-02 /opt/nexus

```

As a good security practice, it is not advised to run nexus service as root. so create new user called nexus and grant sudo access to manage nexus services

```sh

sudo adduser nexus

# visudo \\ nexus ALL=(ALL) NOPASSWD: ALL

sudo chown -R nexus:nexus /opt/nexus

```

Open /opt/nexus/bin/nexus.rc file, uncomment run\_as\_user parameter and set it as following.

```sh

vi /opt/nexus/bin/nexus.rc

run\_as\_user="nexus" (file shold have only this line)

```

Add nexus as a service at boot time

```sh

sudo ln -s /opt/nexus/bin/nexus /etc/init.d/nexus

```

Login as a nexus user and start service

```sh

su - nexus

service nexus start

```

Login nexus server from browser on port 8081

http://<Nexus\_server>:8081

Use default credentials to login

username : admin

password : admin123

### Troubleshooting

service is not starting?

- make sure you are trying to start service with nexus user.

- check java installation

Unable to access nexus URL?

- make sure port 8081 is opened in security group.

### Next Steps

- [x] [Configure Users & Groups in Jenkins](https://youtu.be/jZOqcB32dYM)

- [x] [Secure your Jenkins Server](https://youtu.be/19FmJumnkDc)

- [x] [Jenkins Plugin Installation](https://youtu.be/p\_PqPBbjaZ4)

- [x] [Jenkins Master-Slave Configuration](<https://youtu.be/hwrYURP4O2k>)

Goto Jenkins, add plugins

