Setup and install Maven :

**Step 1: Download and install JDK (if previously not downloaded)**

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

* Set the **JAVA\_HOME** environment variable to point to the base directory

location where Java is installed on your machine.

* Set the environment variable JAVA\_HOME to C:\ProgramFiles\Java\jdk1.7.0\_60
* Append the string “;C:\Program Files\Java\jdk1.7.0.60\bin” to

the end of the system variable, Path.

**Step 2: Download Maven Archive**

[**http://maven.apache.org/download.cgi#**](http://maven.apache.org/download.cgi)

**Download** [apache-maven-3.5.3-bin.zip](http://www-eu.apache.org/dist/maven/maven-3/3.5.3/binaries/apache-maven-3.5.3-bin.zip) (binary zip archive file) and extract it to the program files (c:\)

**Step 3: Set Maven Environment variables**

Go to Control Panel 🡪 System 🡪 Advanced System settings 🡪 Environment variables 🡪System variable 🡪 new

|  |  |
| --- | --- |
| **Variable** | **Value** |
| M2\_HOME | Location of Maven folder ( C:\Program Files (x86)\apache-maven-3.5.3) |
| MAVEN\_HOME | Location of Maven folder (C:\Program Files (x86)\apache-maven-3.5.3) |
| PATH | Location of Maven bin folder (append using ; to the existing path)  (----------------------- ; C:\Program Files (x86)\apache-maven-3.5.3\bin)\*\* |

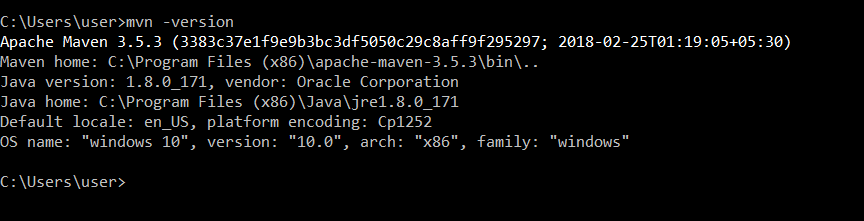
\*Although M2\_HOME suffice your requirement, but sometimes some program do search for MAVEN\_HOME variable too. Therefore, set it also.

\*\* C:\Program Files (x86)\apache-maven-3.5.3\bin == %M2\_HOME%\bin

and we set this variable, so that we can trigger maven executable from command prompt and it points to the right executable file which is in the bin folder

**Check the installation :**

>mvn –version



**Step 4: Define the project structure:**

Go to cmd 🡪 go to directory/folder where u want to build the project

D:\DevOps\Maven\My-maven-first-app>**mvn archetype:generate**

**\***this command will start downloading all the plugins that are required by the maven to run this mvn archetype:generate command. Before running any command, maven requires a set of plugins that are stored in a repository on internet. These plugins do not come as a part of maven installation.

\*When we try running any command for the first time on a system, maven will try to look for the required plugins to run that command on the following location on your system: C:\Users…\.m2\repository/. If it will not find it here, then it will download it from the internet and will store at this location, so that next time before running this command it does not have to download the associated plugins.

1Choose a number: 1 (you can type : maven-archetype-quickstart)

Normally, we just use the following two templates:

* **maven-archetype-webapp** : Java Web Project (WAR)
* **maven-archetype-quickstart** : Java Project (JAR)

Choose a version: 4

2Define value for property ‘groupId’: com.anushasoftware

3Define the value for property ‘artifactId’: maven-demo

Define the value for property ‘version’: 1.0-SNAPSHOT

4Define the value for property ‘package’: com.anushasoftware.demos

**Step 5: Build project –**

**To compile :** Go to cmd 🡪 go to the location/folder containing pom.xml

>mvn compile

Maven will take care of downloading all the related dependencies that has been mentioned in the pom.xml file and going to do the compilation of Java classes in your application directory.

**Test :**

>mvn test **//executes the test cases**

**Packaging :**

>mvn package // **Package it into a jar file (artifact) in the target folder.**

**Install :**

**>**mvn install //**install the artifact in the local rpository**

**Step 6: To look at the default configurations of the super POM**

Go to cmd 🡪 location/root directory(with name same as artifactId) and type

>**mvn help:effective-pom**

1. Choosing the type of architecture for the project.
2. By convention, groupId is the reverse of the domain name of the organizations web address.
3. Whenever a project is developed it produces an outcome, which can be a JAR,WAR or any other distribution archive file. When Maven generates the outcome/distribution archive file, it gives the name :

artifactId-version.extension

e.g maven-demo-1.0-SNAPSHOT.jar

1. Whatever project maven is going to create for you, it is going to put some sample Java source files and JUnit files. It asks in which package to store sample code.
2. SNAPSHOT – this string in the version property of any project means, that project is still under development and not completed.
3. Whatever value you will provide to the artifactId == name of the root folder of your project.

Adding Dependencies

**Step 1: Clean the target folder**

>**mvn clean** //clean is the phase that removes all the classes and distributions generated before

**Step 2: Add dependency**

* Go to the java file (App.java) and add a slf4j logging framework API.

Use logger object to print “hello world” using a logger file:

**Logger logger =LoggerFactory.getLogger(App.class);**

**logger.info(“hello world”);**

* Also import slf4j dependency in the code

**import org.slf4j.\*;** // imported all the classes of this jar

* Ask maven to compile: it will return an error message (package slf4j does not exist). It couldn’t import slf4j. How to tell maven to pull up slf4j jar file :
* Assuming this jar file is not I the local repository, we can tell maven to connect to the central repository and pull up a jar file
* Go to the pom.xml and edit it by adding dependency tag for slf4j. Get the co-ordinates (GroupId, ArtifactId,Version) through various online Maven repository.
* Go to browser 🡪 Maven repository 🡪 slf4j API module 🡪 Click on the release version & copy the co-ordinates
* Paste them in pom.xml file under dependency tag.
* Build the project

\*If not doing with maven, then we would have to download the slf4j jars and add it to the class path in order to have dependency & then only would be able to compile any class file which has dependency on this jar. Maven does it automatically for us.

\*By default the <scope> would be compile phase, which means that this jar would be available during compile time.

Creating a web application using Maven

**Step 1: Create an archetype of the project**

- Go to the folder location on cmd

>mvn archetype:generate

- Choose webapp archetype (e.g. webapp-j2ee14) 269

- Choose version, groupId, artifactId, package, version

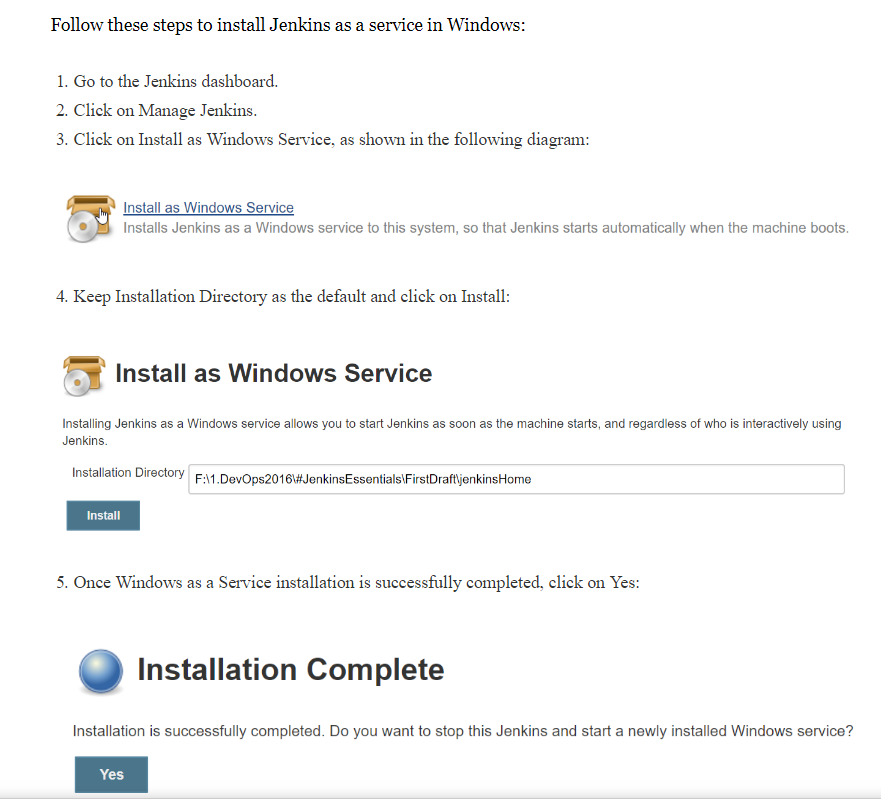
- mvn compile (at the folder where pom.xml is) : it will download all the dependencies and compile it

- mvn package – war file will be ready

Step 2:

Jenkins

1. **Installing Jenkins as window service:**



1. **Installing Plugins:**
2. Go to Jenkins dashboard 🡪Manage Jenkins 🡪 Manage Plugins
3. Go to Available tab 🡪 plugin to install 🡪 Install without restart
4. **Verify the successful plugin installation:**

Go to browser 🡪 localhost:8080/updateCenter/

\*If installation is pending with restart, then restart Jenkins

1. **Install and add plugin to Jenkins:**
2. Go to <https://updates.jenkins-ci.org/download/plugins/> and download the required plugin (e.g. copyartifact.hpi – permalink to the latest).
3. Go to Jenkins dashboard 🡪Manage Jenkins 🡪 Manage Plugins 🡪 Advanced tab 🡪 choose the file and upload plugin.
4. Check the success of installation <http://localhost:8080/updateCenter/>
5. **How to change the default 8080 port for Jenkins**

(to avoid any conflict with any other app using the same port)

On command prompt:

>java -jar jenkins.war –httpPort=9090

OR

Go to Program files/Jenkins directory 🡪 Jenkins.xml 🡪change the port no.

1. **How to configure Maven In Jenkins**
2. Go to Jenkins Dashboard 🡪Manage Jenkins 🡪Manage plugins 🡪 Available 🡪Maven Integration 🡪 Install
3. Go to Manage Jenkins🡪Global tool configuration 🡪 Maven 🡪 Add Maven\_home variable value (i.e. path of the maven file on your system).
4. Go to Jenkins Dashboard 🡪 New Item 🡪 Maven Project option will be available
5. **Build a simple Maven Project in Jenkins**
6. Go to Jenkins Dashboard 🡪 New Item 🡪Choose name for the Maven Project .
7. On Configure Page, set the following:

a). Discard Old builds

Days to keep builds: 1

Max of builds to keep: 5

b). JDK (to be used for this project) Java-1.8

c). Build 🡪 adavanced 🡪 enable the following

- Resolve Dependencies during Pom parsing

- Use custom workspace (add path of the folder containing pom.xml)

Build 🡪 Goals & Options = Clean Compile Test

1. Apply & Save

\*Error : Compilation error - No compiler is provided in this environment. Perhaps you are running on a JRE rather than a JDK?

Probable cause: not able to connect to/find jdk.

Solution:

1. Go to Jenkins dashboard🡪 Manage Jenkins 🡪Configure System 🡪Global properties 🡪 Add Environment Variable :

JAVA\_HOME and value

1. Also check in the system of your windows :

System 🡪 advanced settings 🡪environment variables

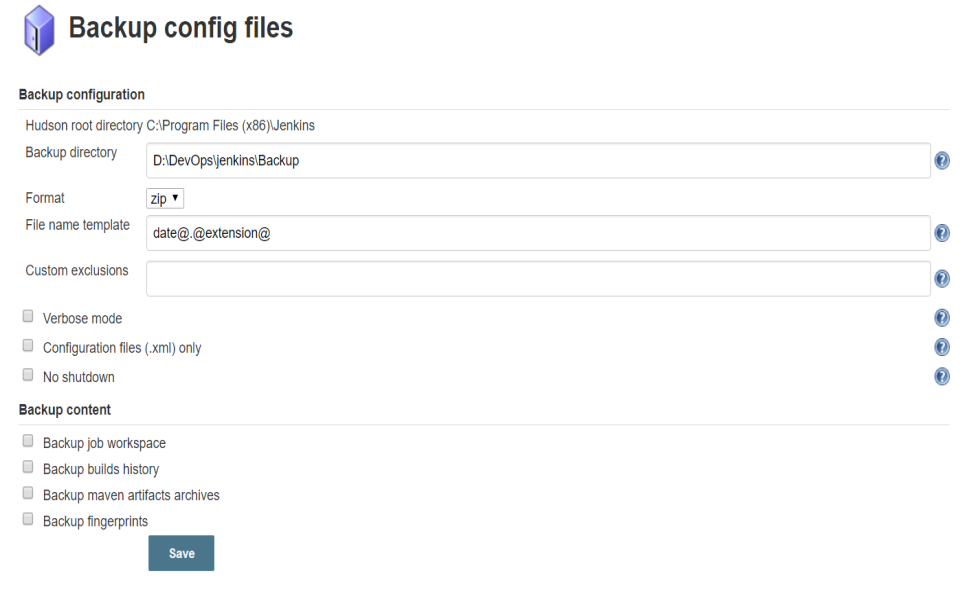
Set the PATH : Append with the folder path of the jdk

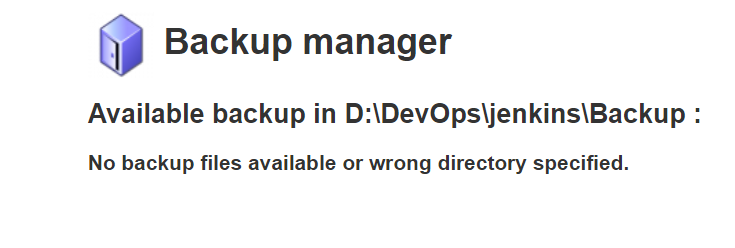
If, user variable is pointing towards JRE, then update it to point towards java path.

1. **Backing Up and Restoring Jenkins**

Jenkins has a backup plugin which can used to backup critical configuration settings related to Jenkins. A core task for the smooth running of Jenkins is the scheduled backing up of its home directory. This is not necessarily all the artifacts, but is at least its configuration and the history of testing, which plugins will need to make reports.

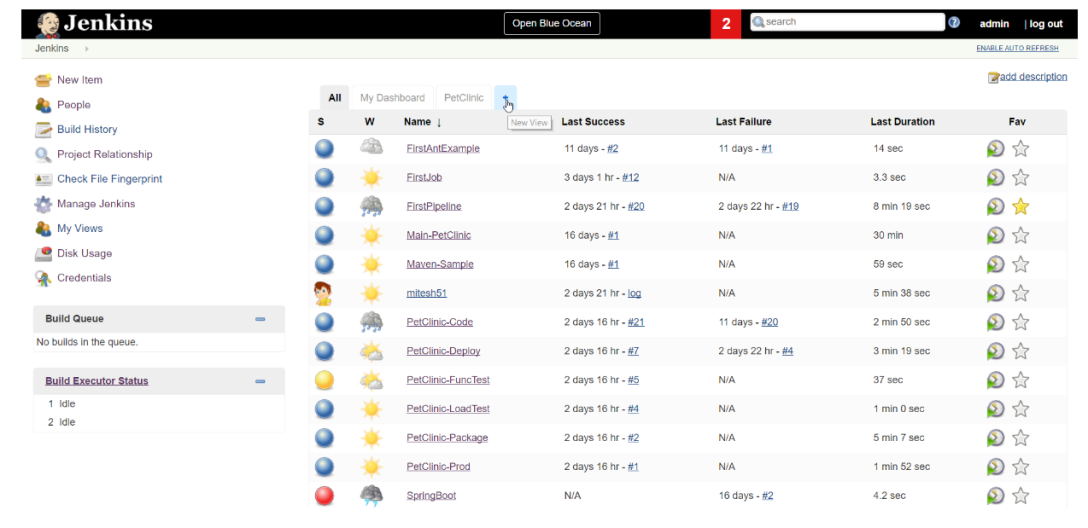
1. Go to Manage Jenkins 🡪 Manage Plugin 🡪 Available 🡪install without restart the backup plugin.
2. Manage Jenkins 🡪 Backup Manager 🡪 Set Up



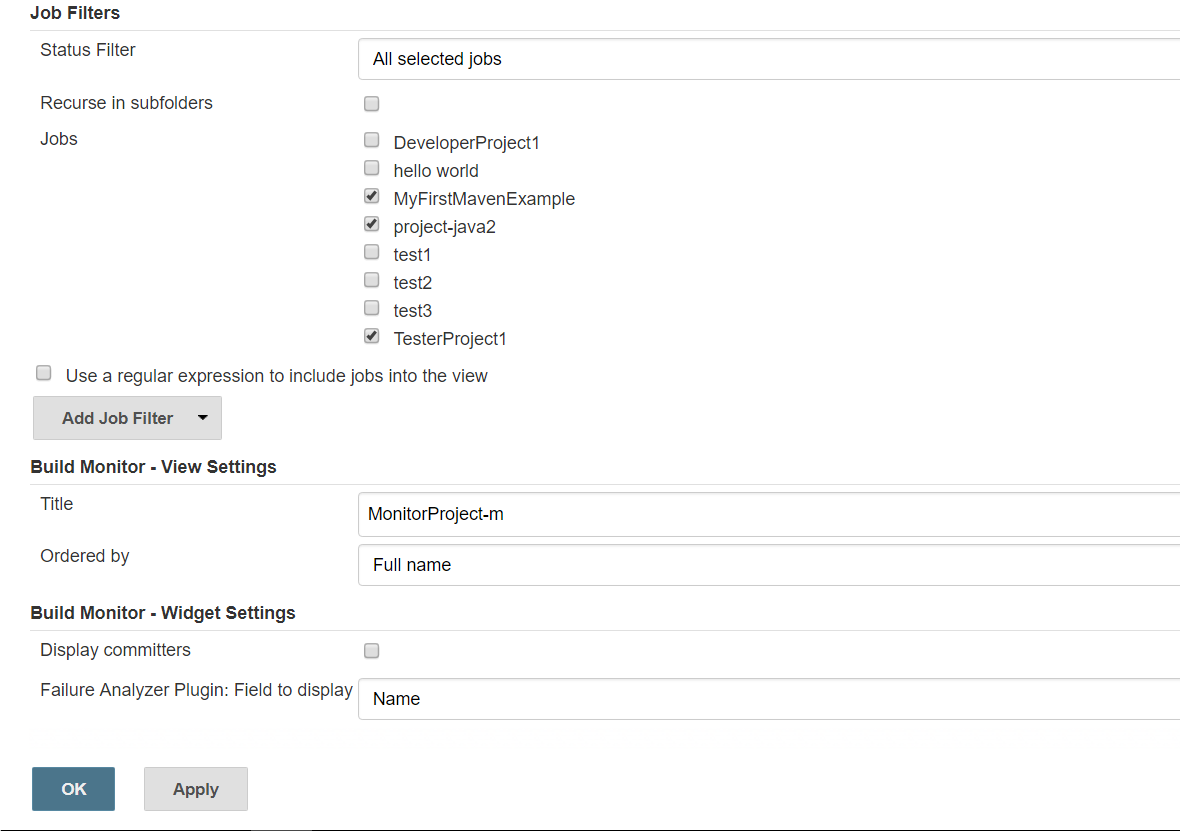
1. Backup Manager 🡪 Backup Hudson Configuration
2. Go to Restore Hudson Configuration
3. **Jenkins monitoring through JAVA melody**
4. Go to Jenkins dashboard 🡪 manage Jenkins 🡪 manage plugins 🡪 install monitoring plugin.
5. Go to Manage Jenkins 🡪 Monitoring of Jenkins Master
6. **Monitoring a Jenkins job using Build Monitor View**

The Build Monitor plugin provides a visualization of the status and progress of selected Jenkins jobs.

1. Go to Manage Jenkins 🡪 Manage Plugins 🡪install Build Monitor plugin.
2. Go to Jenkins Dashboard 🡪 Click on New View

****

1. Type the name of view (e.g. MonitorProject-m) and click on Build Monitor View.
2. From the view window select the jobs to be monitored.



1. Click on apply and ok, and the monitor screen will appear.



1. **Configuring e-mail notifications**

It is essential to know when something fails so that we can take corrective measures at the right time. You need to have an email account to configure email notifications in Jenkins and SMTP details.

Go to Manage Jenkins 🡪 Go to Email Norification

1. **Archiving :**
2. Log in Jenkins as an Administrator.
3. Go to Manage Jenkins 🡪Script Console link 🡪 type the following groovy script and run it.

**def warning='[You can ARCHIVE this Job]'**

**def now=new Date()**

**for (job in hudson.model.Hudson.instance.items) {**

**println "\nName: ${job.name}"**

**Run lastSuccessfulBuild = job.getLastSuccessfulBuild()**

**if (lastSuccessfulBuild != null)**

**{ def time = lastSuccessfulBuild.getTimestamp().getTime()**

**if (now.month.equals(time.month)){**

**println("Project has same month as build");**

**}else**

**{ if (job.description.startsWith(warning)){ println("Description has already been changed"); }**

**else{**

**job.setDescription("${warning}") } } } }**

1. Successful execution of the script will give output based on the build jobs available in your Jenkins.



1. Any project that had its last successful build in another month than this will have the description you can archive this job added to its description.
2. Click on Configure to check the description as well.

How does this script works??

* A warning string is defined and the current date is now stored. Each job in Jenkins is programmatically iterated via the for statement.
* Jenkins has a class to store build run information. The runtime information is retrieved via job.getLastSuccessfulBuild() and is stored in the lastSuccessfulBuild instance. If no successful build has occurred, then lastSuccessfulBuild is set to null; otherwise, it has the runtime information.
* The time of the last successful build is retrieved and then stored in the time instance via lastSuccessfulBuild.getTimestamp().getTime().
* The current year is compared with the year of the last successful build and, if they are different and the warning string has not already been added to the front of the job description, then the description is updated.

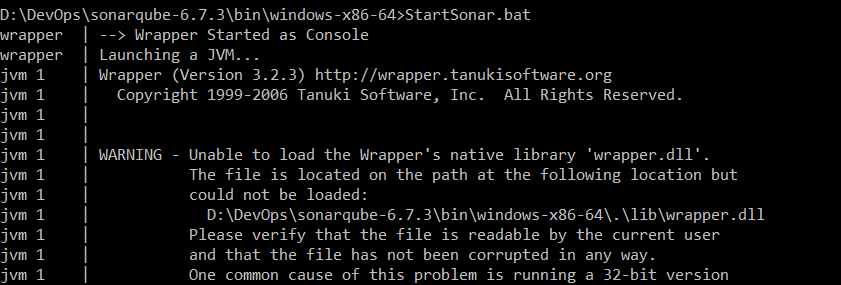
1. **Integrating Jenkins with SonarQube**

Sonar is a specialized tool that collects software metrics and breaks them down into understandable reports. Java8 is a pre-requisite.

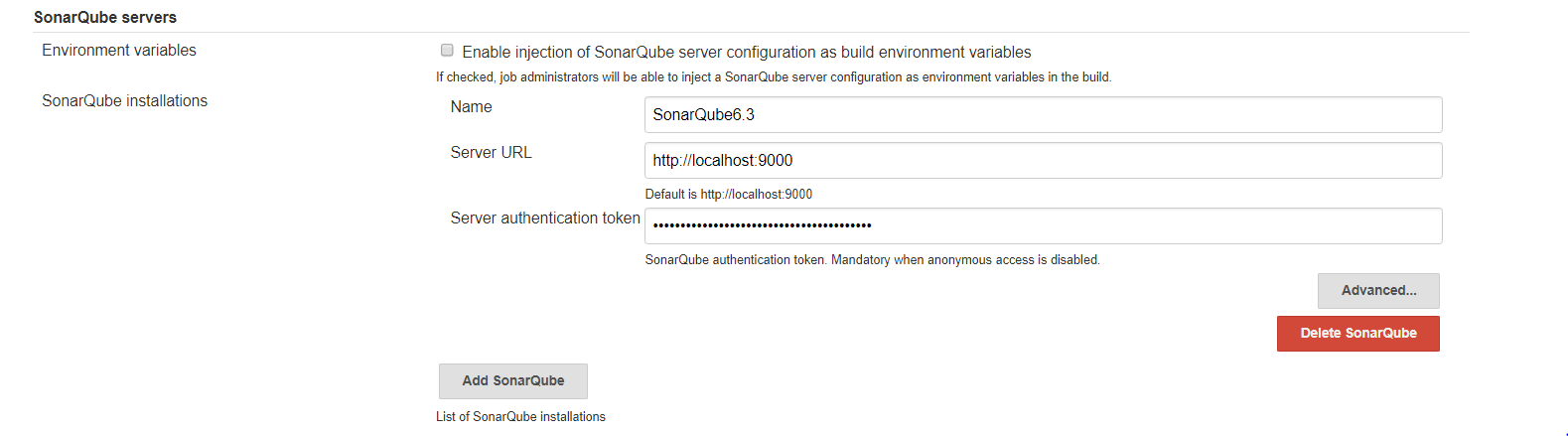
1. Download SonarQube: <https://www.sonarqube.org/downloads/>

Download the latest stable version and extract the .zip in the system. (e.g. D:\).

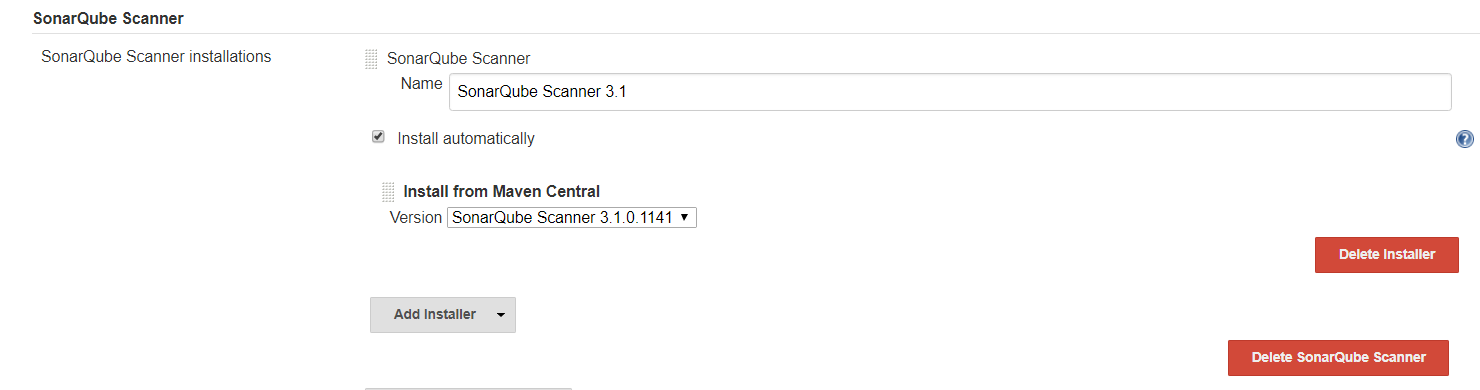
1. Go to the cmd 🡪directory containing 64 bit version
2. Type : StartSonar.bat



1. Once the SonarQube is up, go to the browser : <http://localhost:9000> to access SonarQube dashboard.
2. Login with default credentials (admin as userId and Password).
3. Go to Administration 🡪 Security 🡪 Generate token (copy the token).
4. Go to Jenkins dashboard 🡪 Manage Jenkins 🡪 Manage plugins 🡪 Available 🡪 SonarQube Scanner & Sonar Quality Gates (Install without restart).
5. Go to Manage Jenkins 🡪 Configure System 🡪SonarQube section and enter the details.



1. Go to Manage Jenkins 🡪 Global Tool Configuration 🡪 SonarQube Scanner .



1. Now you are ready for the static code analysis of the project.
2. **Perform static code Analysis on sample application(e.g. JavaScript, HTML,CSS) using SonarQube:**
3. Go to Jenkins dashboard 🡪New Item (SonarQube-Demo) 🡪 Freestyle project.
4. Provide the repository URL in the SCM section of the project configuration window.
5. Go to Build 🡪 Execute SonarQube Scanner .
6. Provide the location of sonar-project.properties or provide details directly for static code analysis.

# Required metadata

sonar.projectKey=java-sonar-runner-simple

sonar.projectName= project analyzed with the SonarQub Runner

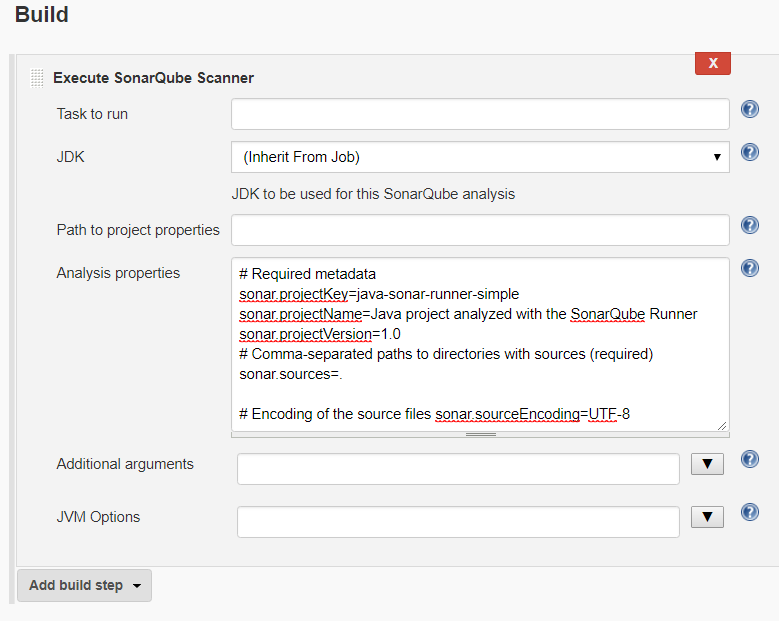
sonar.projectVersion=1.0

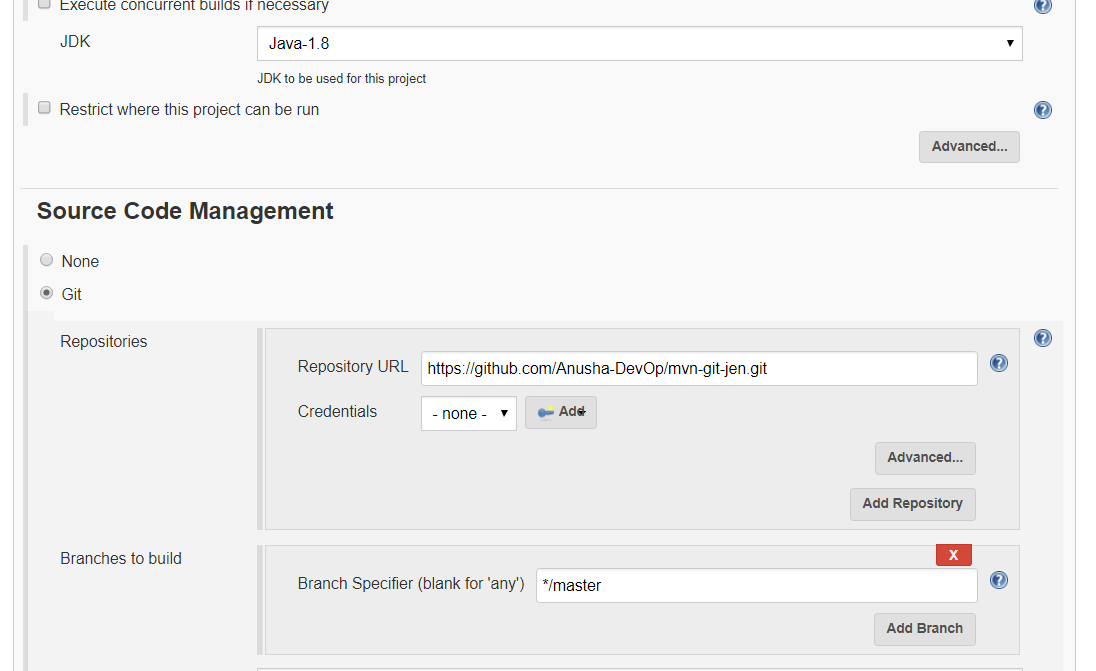
# Comma-separated paths to directories with sources (required)

sonar.sources=.

# Encoding of the source files

sonar.sourceEncoding=UTF-8

1. sonar.sources is the main property for static code analysis. With this property, you inform SonarQube which directory needs to be analyzed:
2. Go to Jenkins project 🡪 Build Now
3. Go to the SonarQube dashboard (<http://localhost:9000>) and check whether static code analysis for our project is available or not.
4. Click the project link and check the related details.
5. **Performing static code analysis (SonarQube scanner) on a Maven Project in Jenkins**
6. Go to Jenkins dashboard 🡪 New Item 🡪 Enter an item name and click on Maven Project.
7. On the configure page of the project:
8. Choose JDK to be used for the project
9. Provide the repository URL (e.g. of your github repository) in the SCM section.

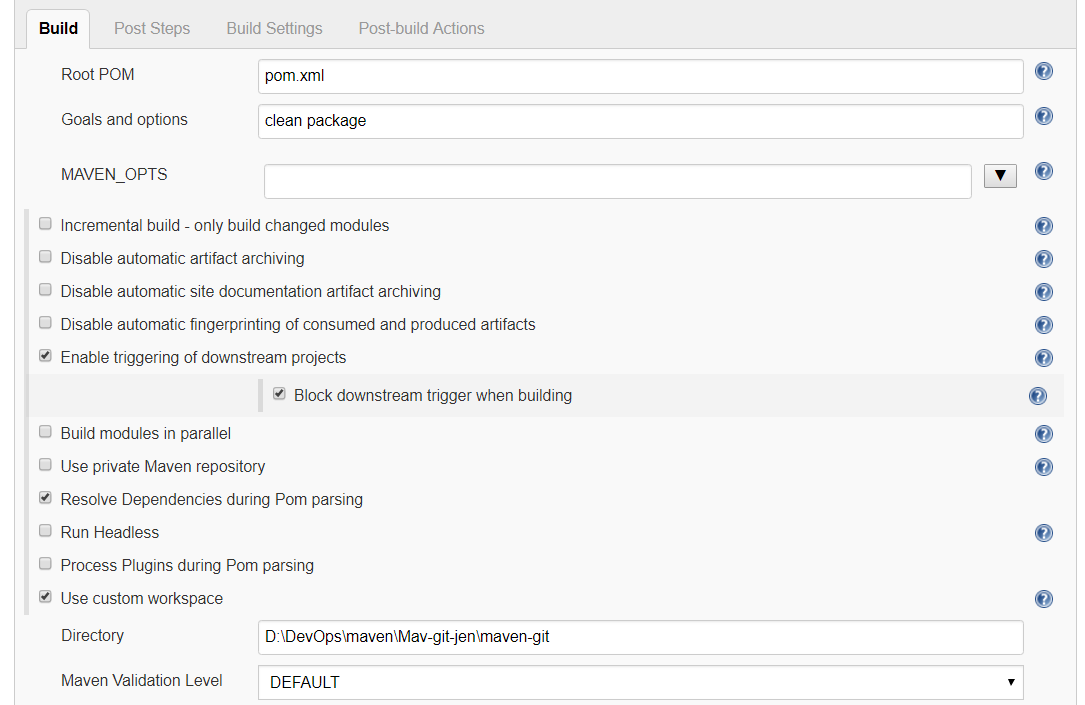
 c).

1. Go to Build 🡪 advanced :

**-**Write the goals/phases you want maven to run

-Check the “Resolve Dependencies during Pom parsing

-Check the “Use custom workspace” and give the path of your directory containing pom.xml file of your project.

****

1. Go to Add post built step 🡪 choose the “Execute SonarQube scanner” 🡪 Type the analysis properties

# Required metadata

sonar.projectKey=Sonar-Maven-demo

sonar.projectName= Maven project analyzed with the SonarQub Runner

sonar.projectVersion=1.0

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

sonar.sources=src/main/java

#Path for binaries

sonar.binaries=target/classes

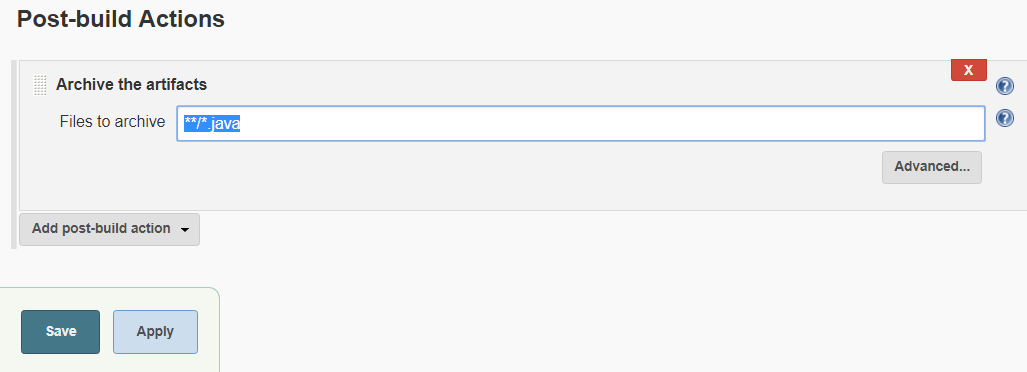
#Language

sonar.language=java

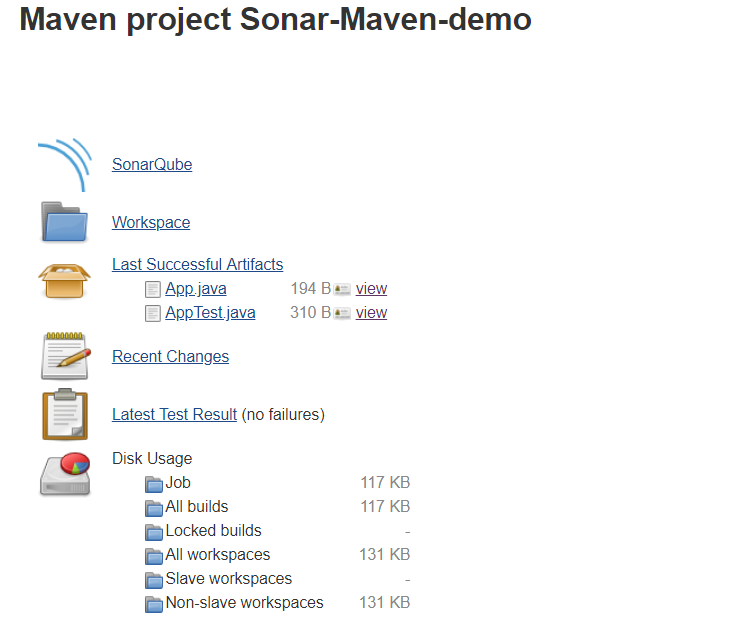
# Encoding of the source files

sonar.sourceEncoding=UTF-8

1. **Go to Post-built Actions 🡪 Archive the artifacts 🡪**

****

1. **Save and click on Build Now. After successful build, you can click on SonarQube to check the details. Also check the artifacts that has been created.**

****

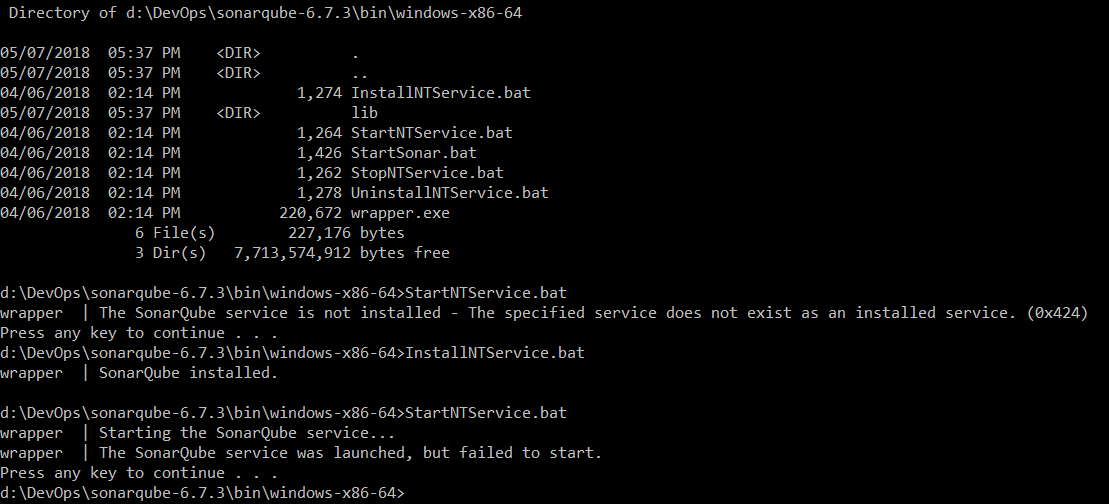
1. **Configuring SonarQube as a Window Service:**

Go to cmd as an administrator

Go to SonarQube | bin | Specific OS Directory | Execute following commands:

>InstallNTService.bat

>StartNTService.bat



Error: SonarQube service was launched but failed to start.

**XVII. How to compile JSP pages based on the maven-jetty-jspc-plugin**

To create a simple java web application, we will use **maven-archetypewebapp**

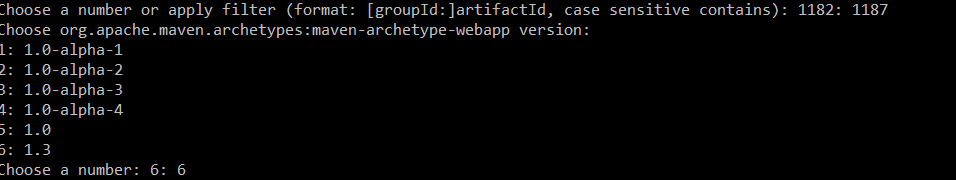
plugin.The compiled code will work with the Jetty server, which is often used for integration tests.

1. Create a folder : ch3.building\_software
2. Go to cmd 🡪 go to the above directory 🡪 and execute the following command to create a WAR project

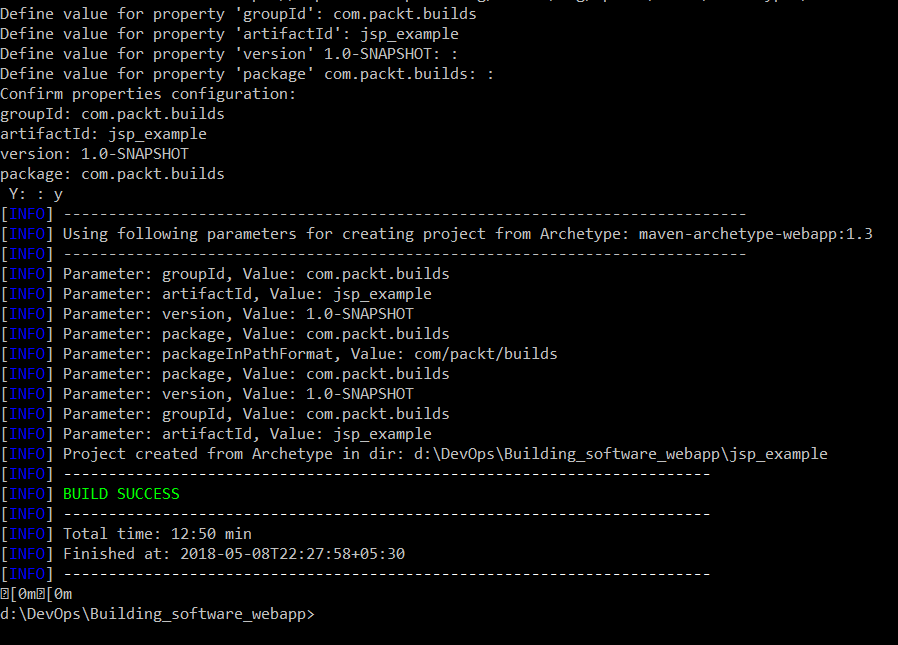
>mvn archetype:generate -DarchetypeArtifactId=maven-archetype-webapp

OR

>mvn archetype:generate

And choose the number corresponding to archetype webapp (by default it is set for archetype quickstart e.g. 1182). Also choose the latest version.

1. Configure the properties (maven co-ordinates): GroupId, ArtifactId, Version and package.



1. **Remotely triggering jobs/builds through Jenkins API using security tokens:**

This will allow you to run other jobs from within maven.

**Pre-requisites:**

**a). Jenkins security is turned on.**

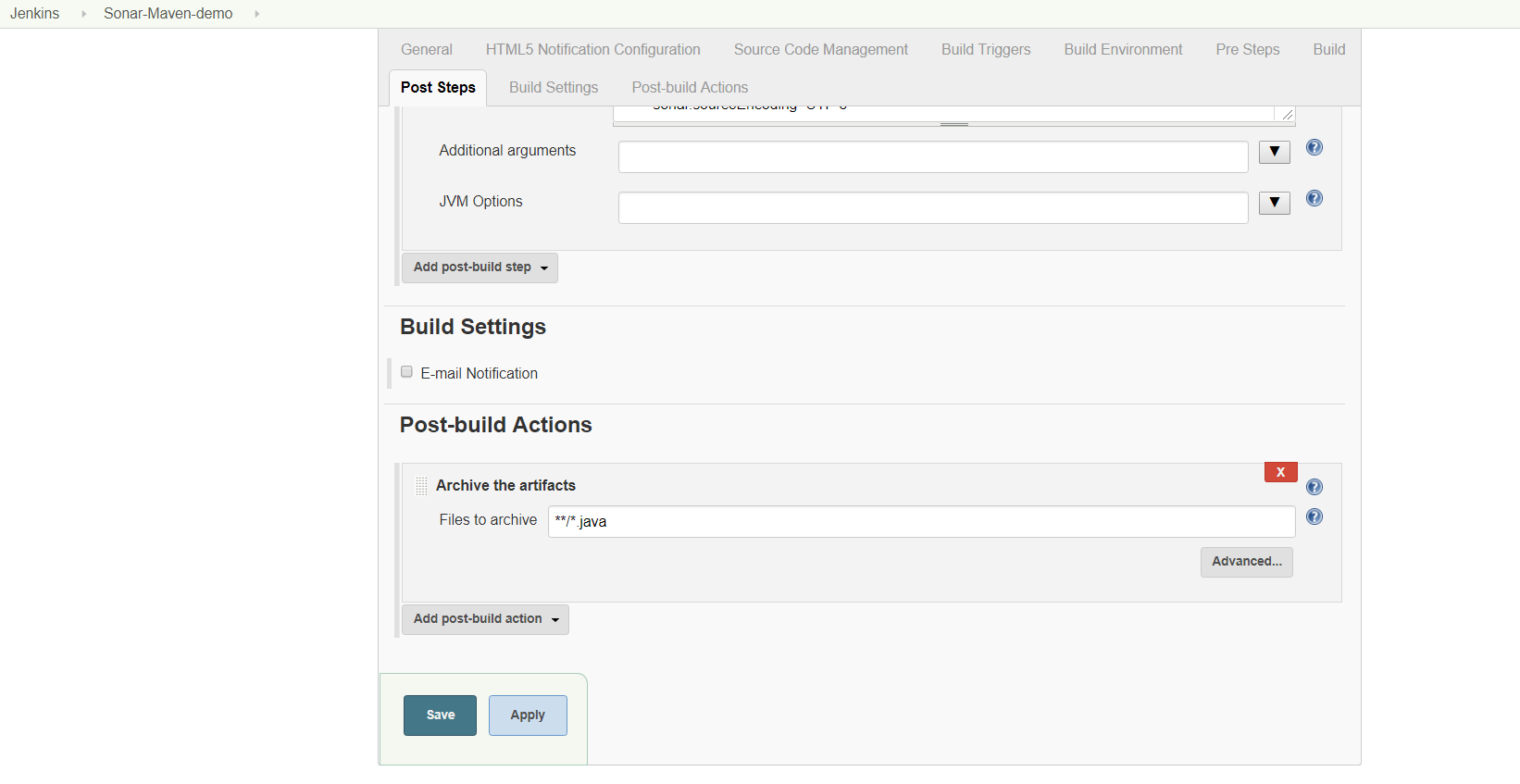
**b). Modern version of wget (**[**http://www.gnu.org/software/wget/**](http://www.gnu.org/software/wget/)**) is installed.**

Continuous Delivery (CD)

**A). Archiving Artifacts:**

Artifact archiving is useful when you want to use an installable or distribution package in another project or build job.

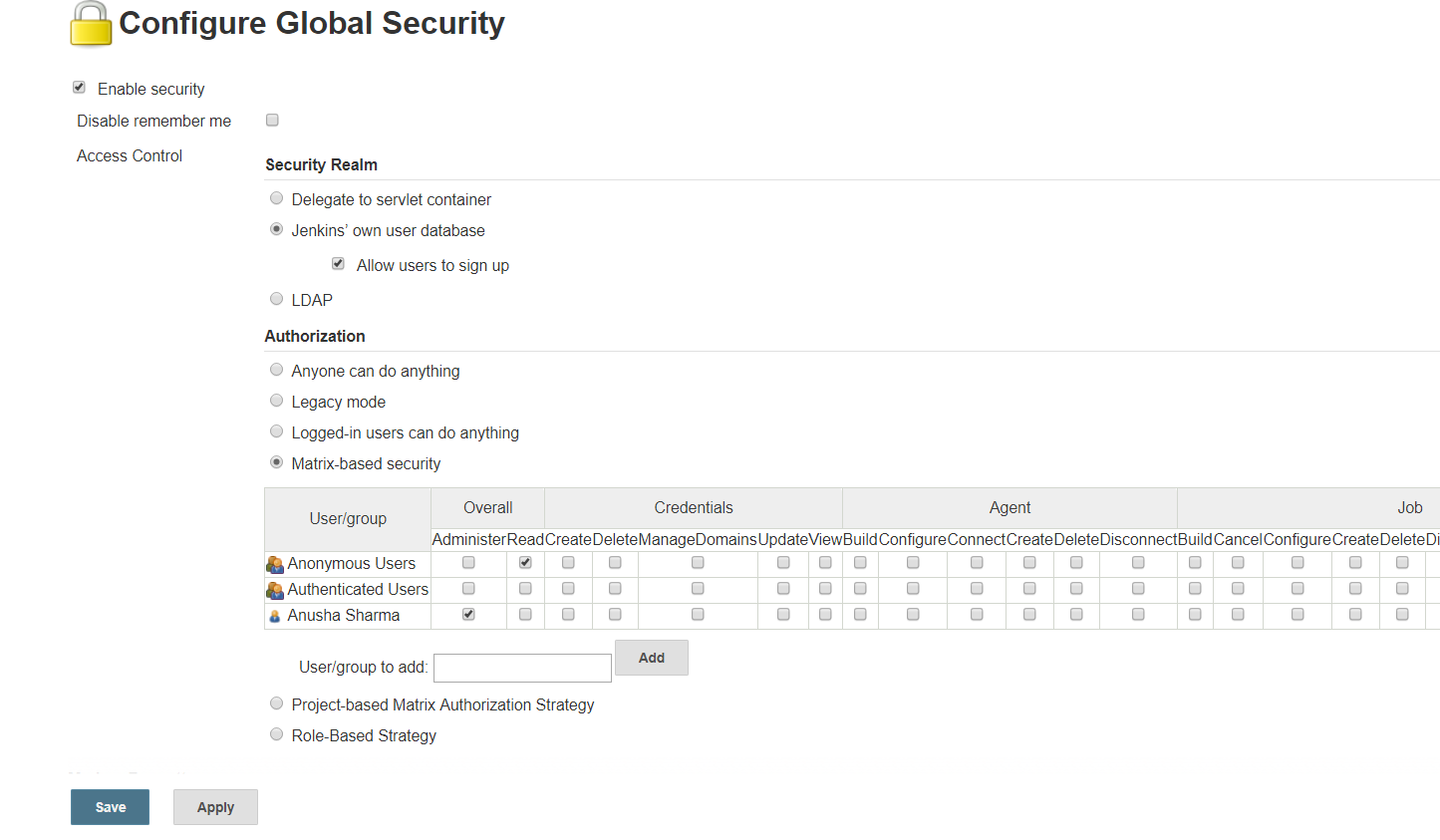
1. Create a project in Jenkins that creates a package file or distribution file, or any artifact that you want to utilize later from another project.
2. Go to Jenkins dashboard🡪 Jenkins project🡪 Post-build Actions🡪Add post-build action 🡪 Archive the artifacts
3. You can use wildcards, such as module/dist/\*\*/\*.zip.



B). **Copying an artifact from another build job:** useful for copying an archived artifact/file/package from another project into Jenkins or from one build to other. It helps in the deployment or copy operations. Even with Master/Agent architecture, we can copy artifacts effortlessly using the Copy Artifact Plugin.

1. You need to allow read access to anonymous users, as the Copyartifact plugin treats builds running as an anonymous. To allow read access to anonymous, do the following:

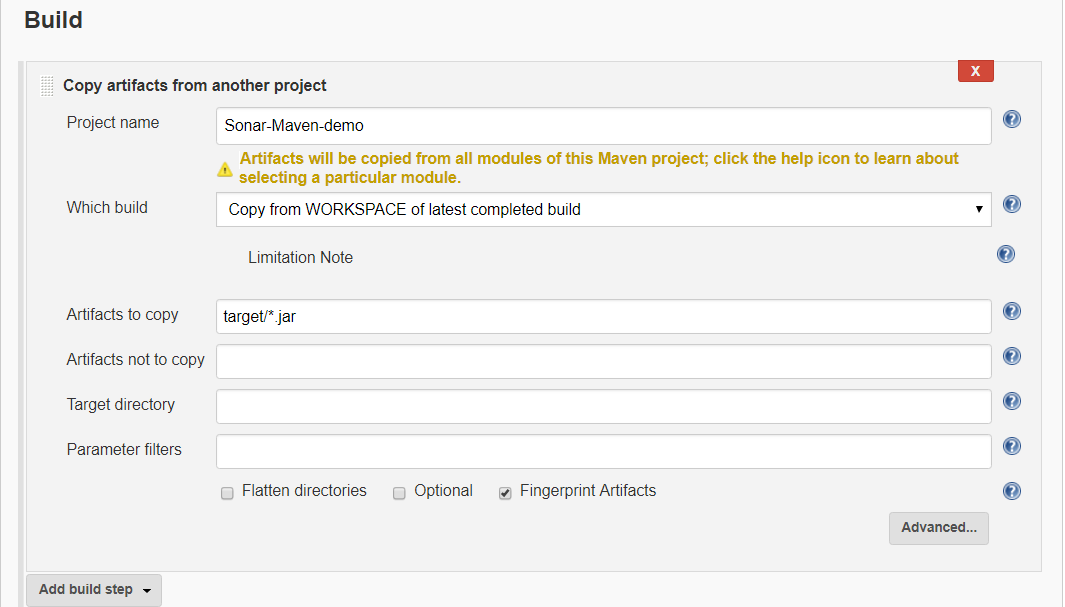
Go to Jenkins 🡪 manage Jenkins 🡪 Configure Global Security 🡪 Go to Security Realm 🡪 check “Allow users to sign up” 🡪 Go to Authentication 🡪 Choose Matrix-based security 🡪 Grant overall Read access to anonymous users.



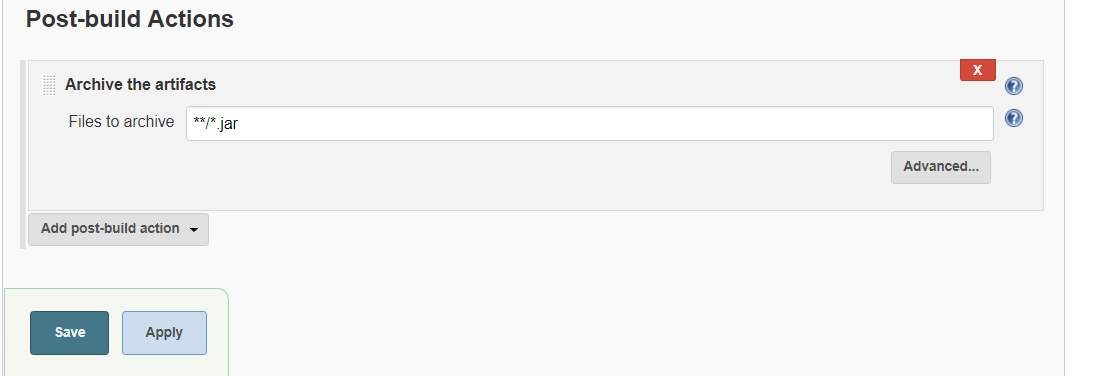
2. Go to Jenkins Project, in which you want to copy the artifact, or you can create a freestyle project.

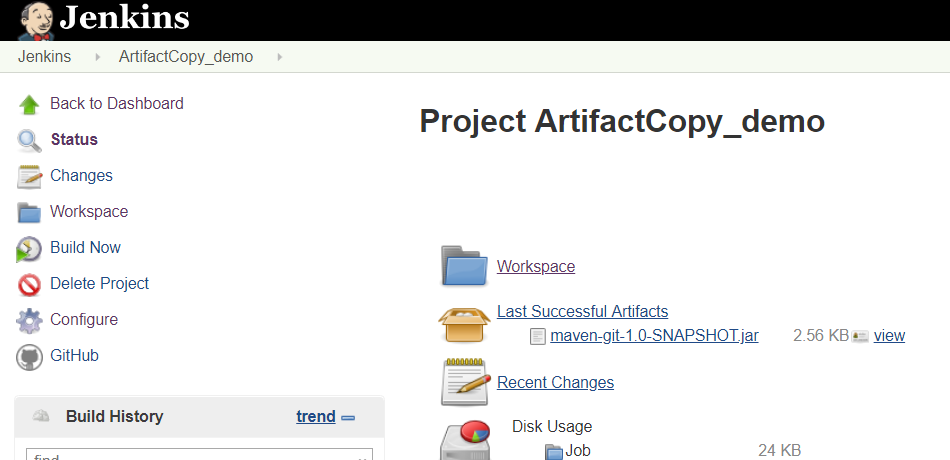
Go to Project Configure 🡪 Build section 🡪 Choose “Copy artifacts from another project”

* Give Name of the project/build in Jenkins from which you want to copy the artifact.
* Choose one option from the various in “Which Build” section (Click on Question mark to get help).
* You can provide the type of artifacts to copy else if left blank, it will copy every artifact.



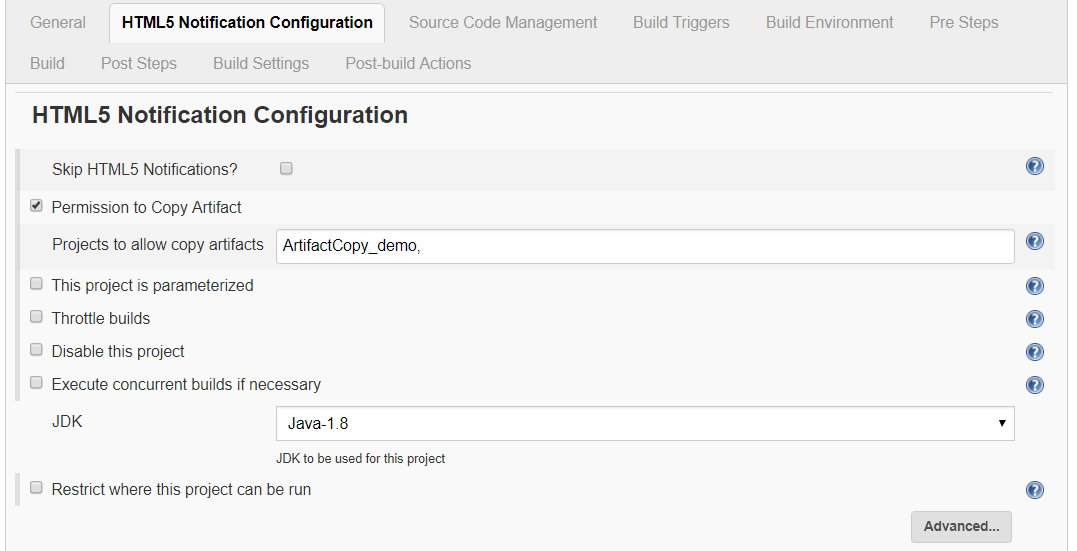
1. Go to Post-build Actions 🡪Choose Archive the artifacts (in order to see the artifacts on your project’s workspace) 🡪 Choose types of files to archive (use \*\* if you want to archive all files in the workspace).





1. Also, you can specify the projects in your build that can copy the artifacts.

* Go to Project 🡪Configure 🡪 HTML5 Notification Configuration 🡪 Permission to Copy Artifact 🡪 Provide the name of the projects to which you want to allow to copy the artifacts of your current project.

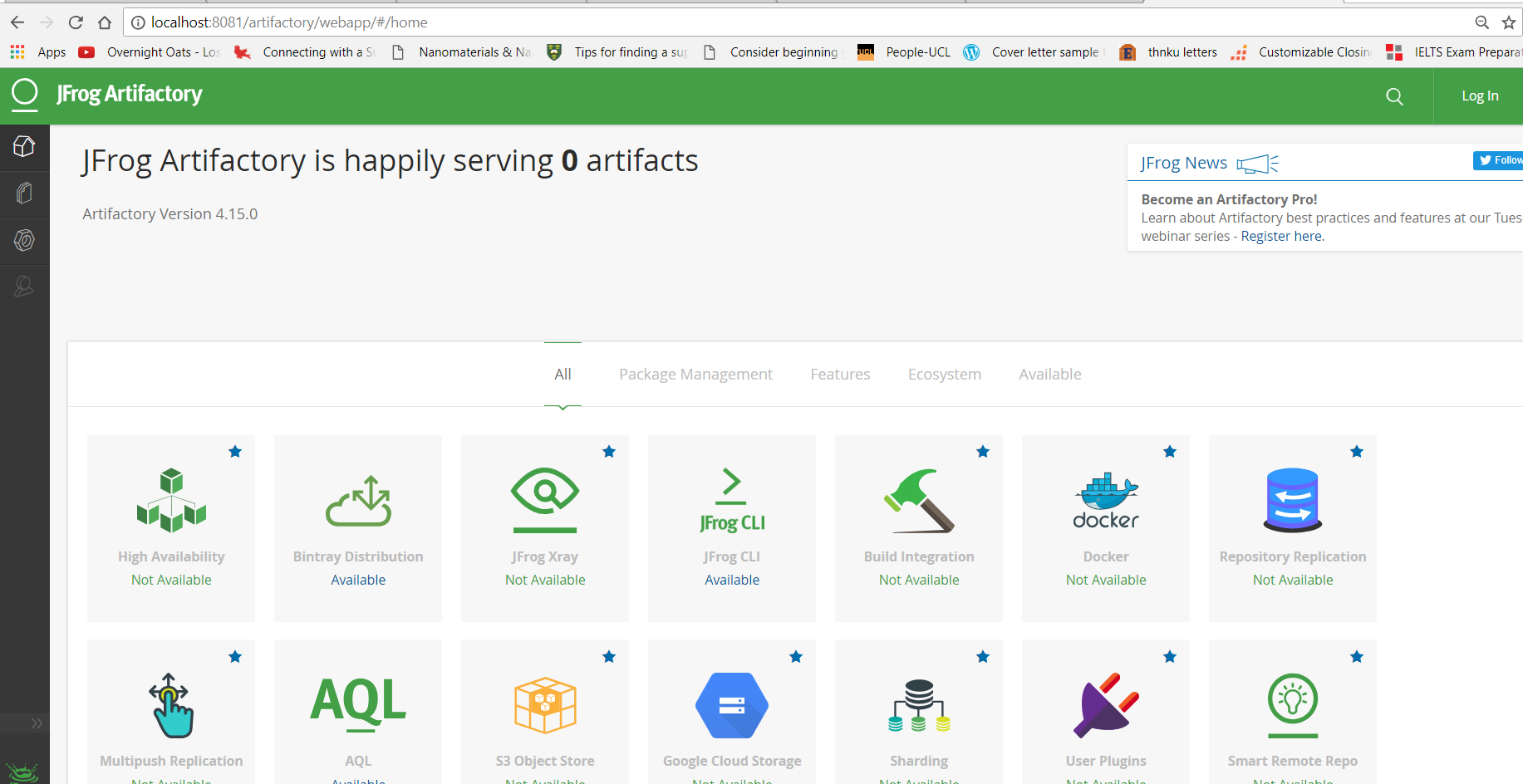


**C).** **Integrating Jenkins with Artifactory:**

1. Download the JFrog artifactory .zip folder from <https://bintray.com/jfrog/artifactory/jfrog-artifactory-oss-zip/4.15.0>

2. Extract the .zip folder in your system. Go to the Bin folder and execute *artifactory.bat*

3. Go to the browser and visit localhost:8081 in order to visit to the artifactory in browser.



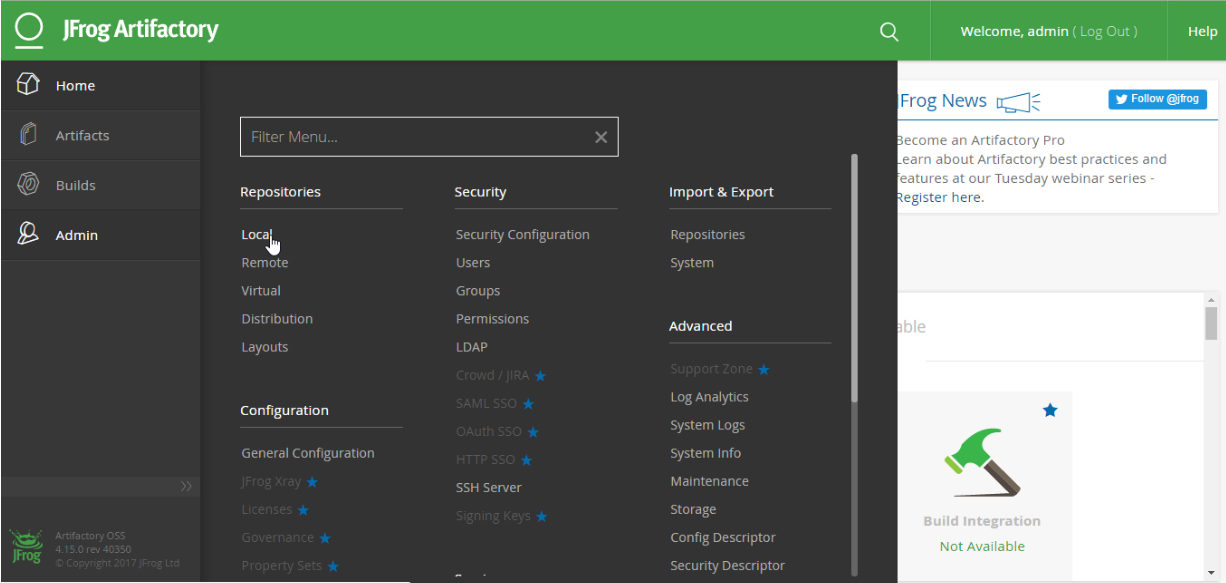
1. Log in as admin by providing the default credentials:

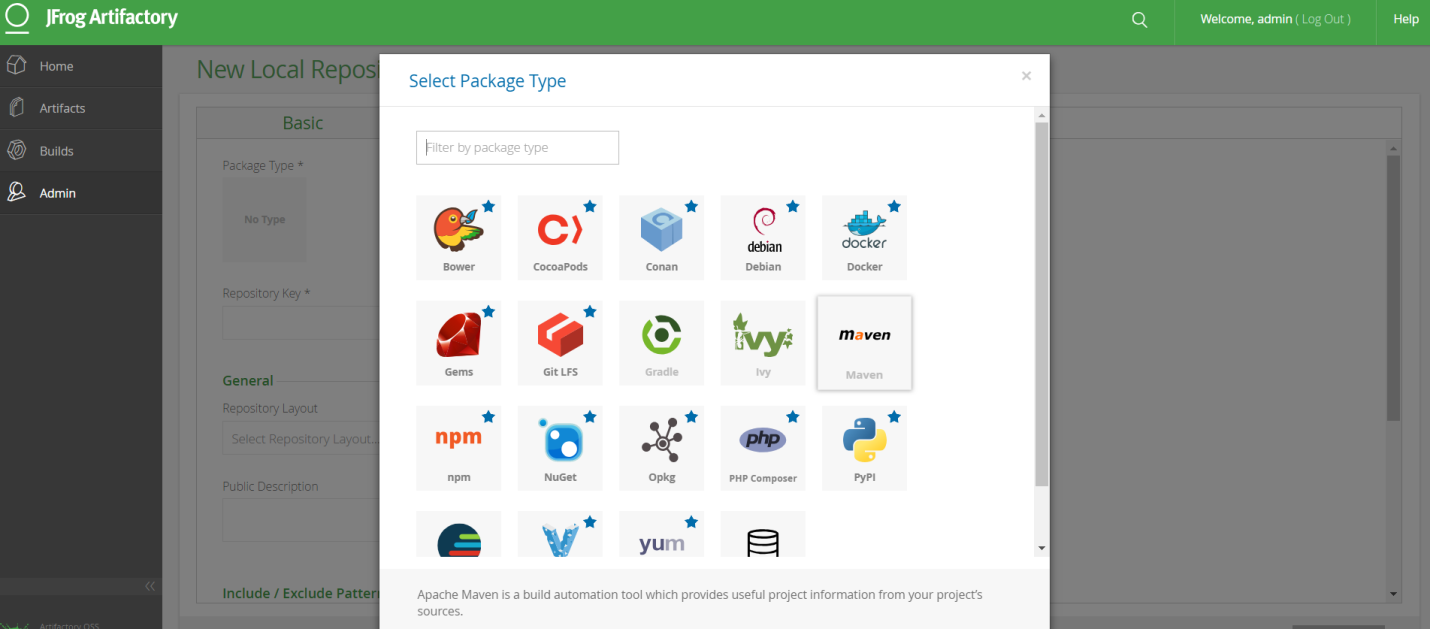
Username: admin

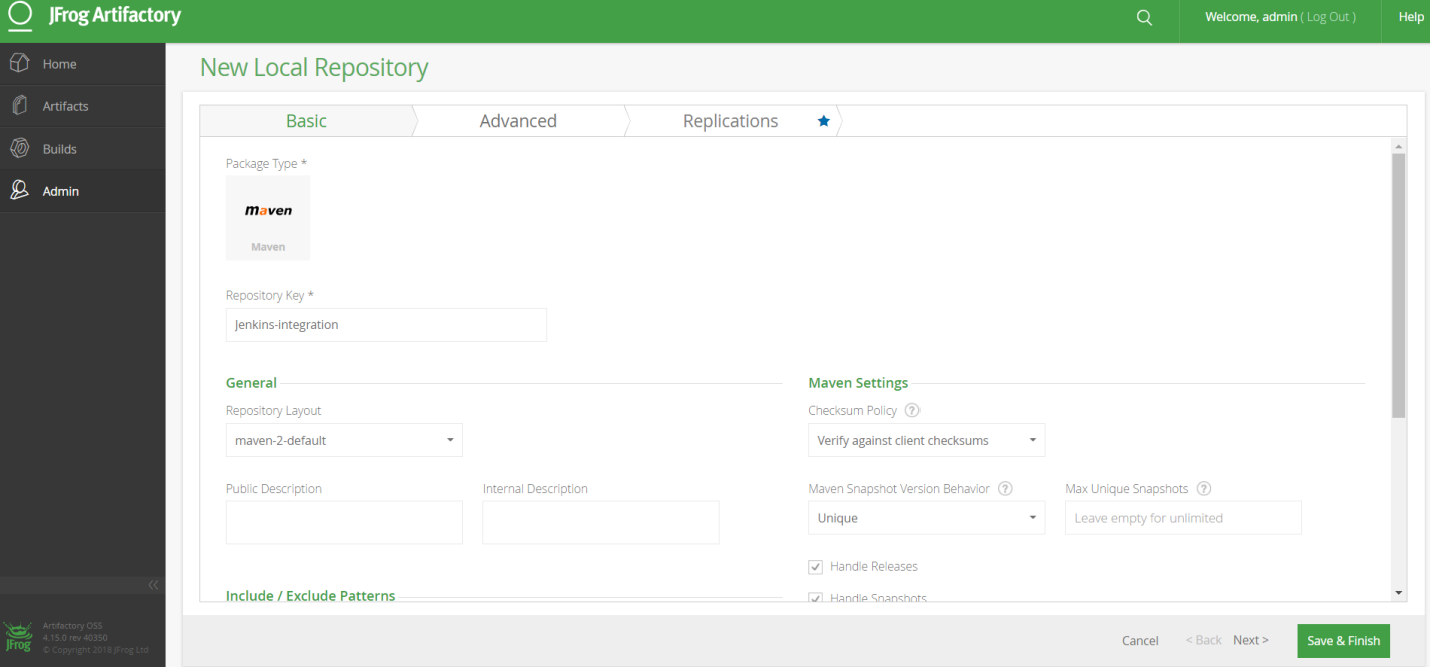
Password: password

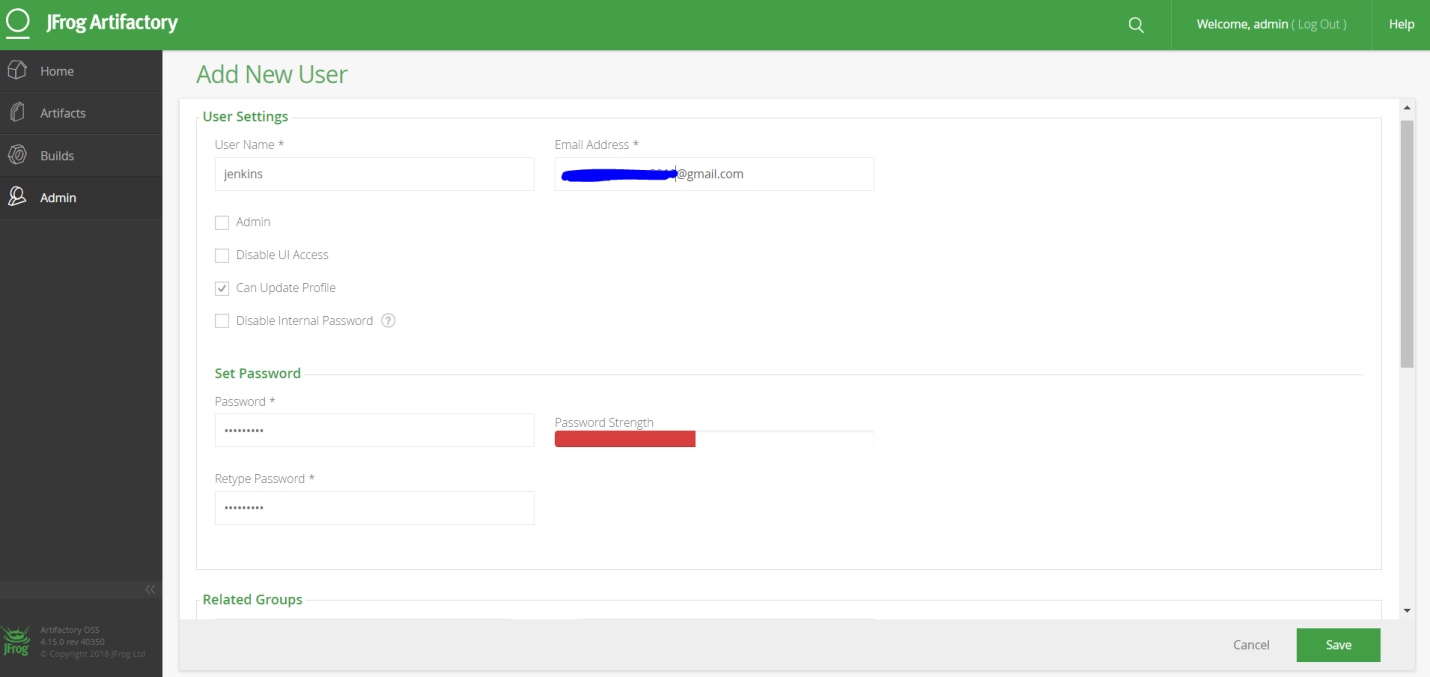
1. Create a Local repository to store package files created by the Jenkins project:

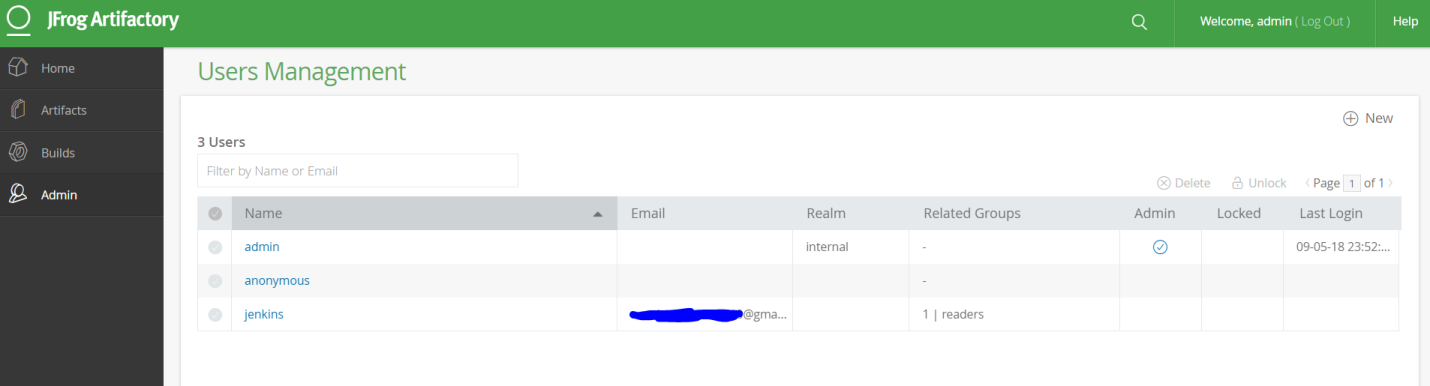
Go to Admin 🡪 Repositories 🡪 Local

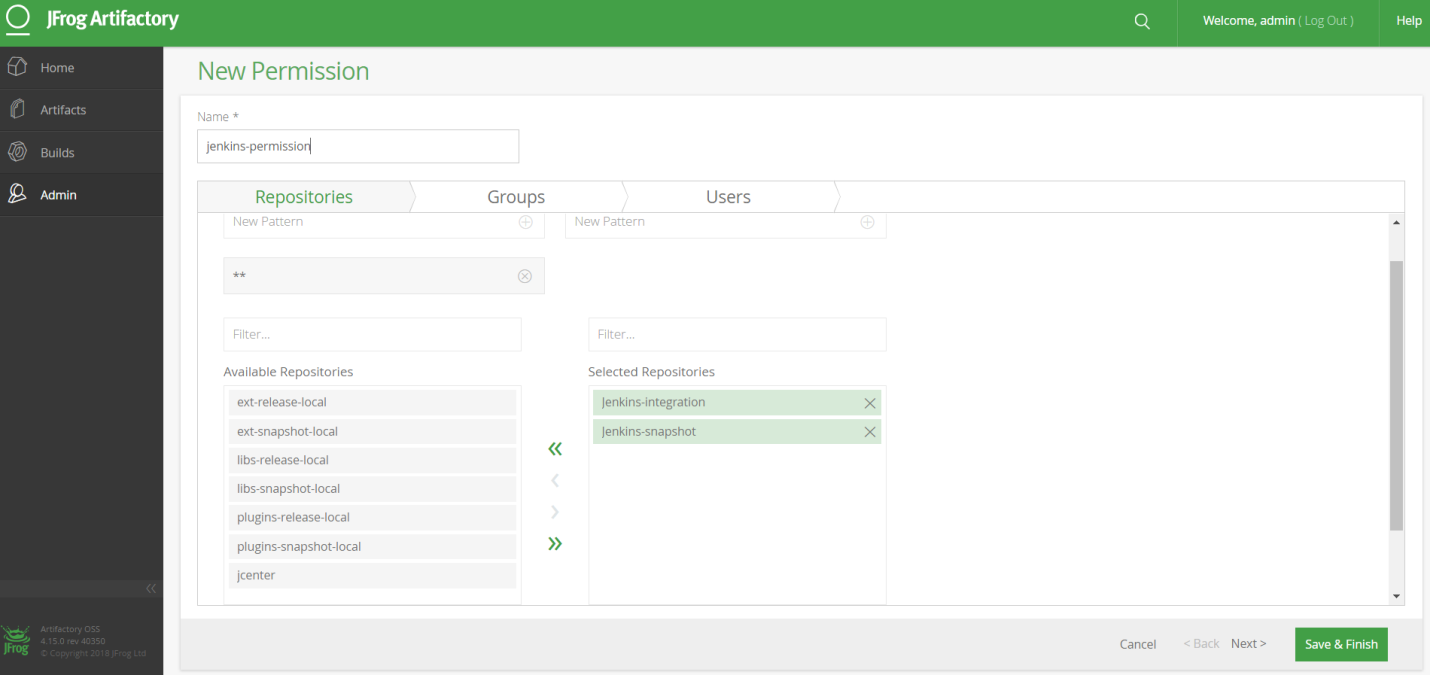


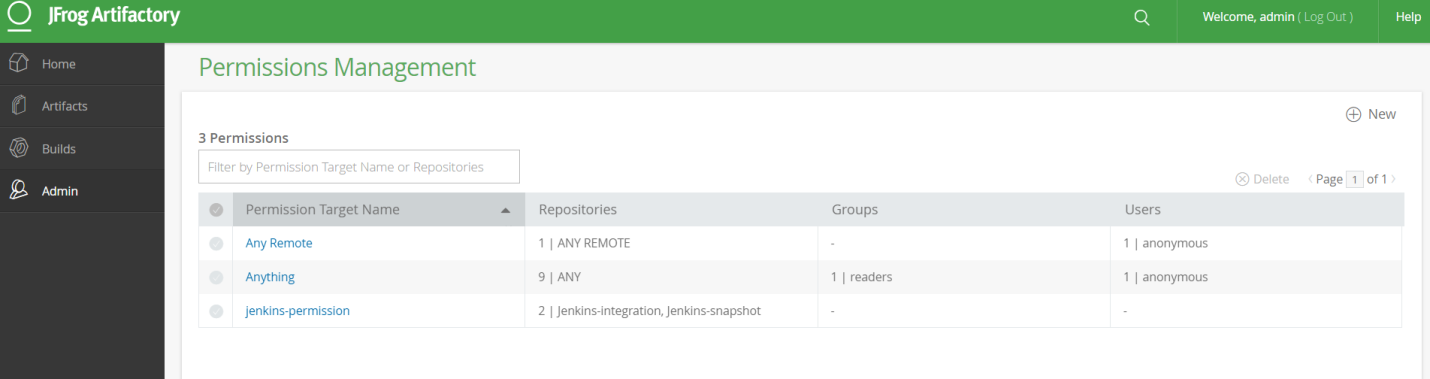


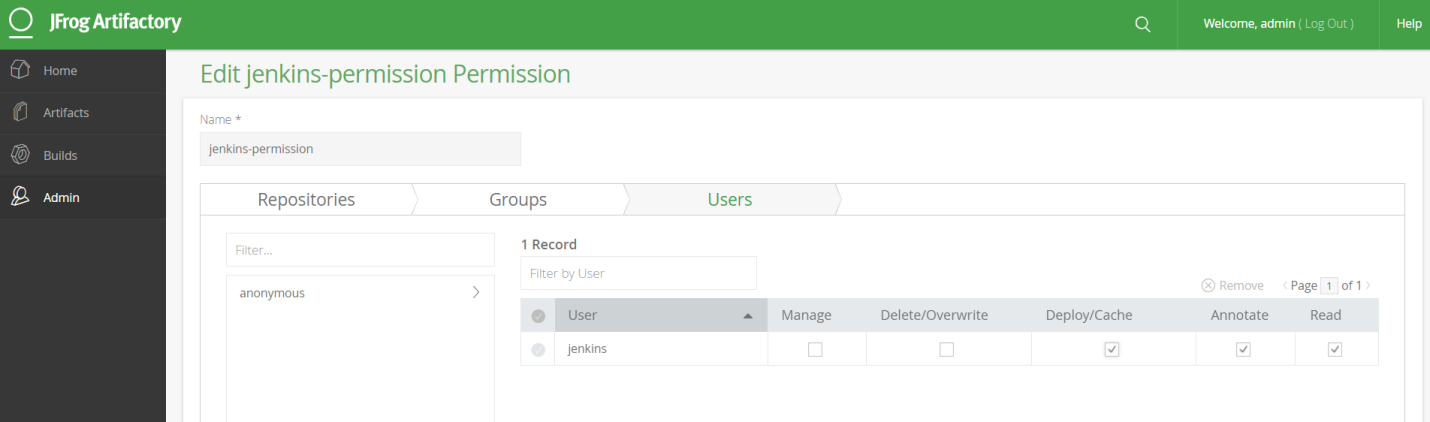


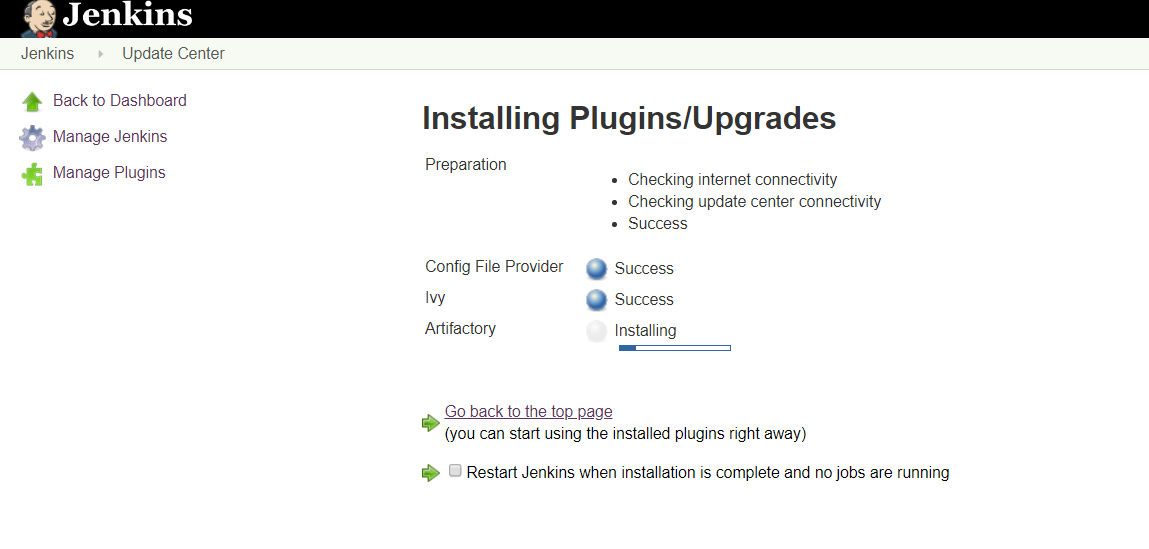


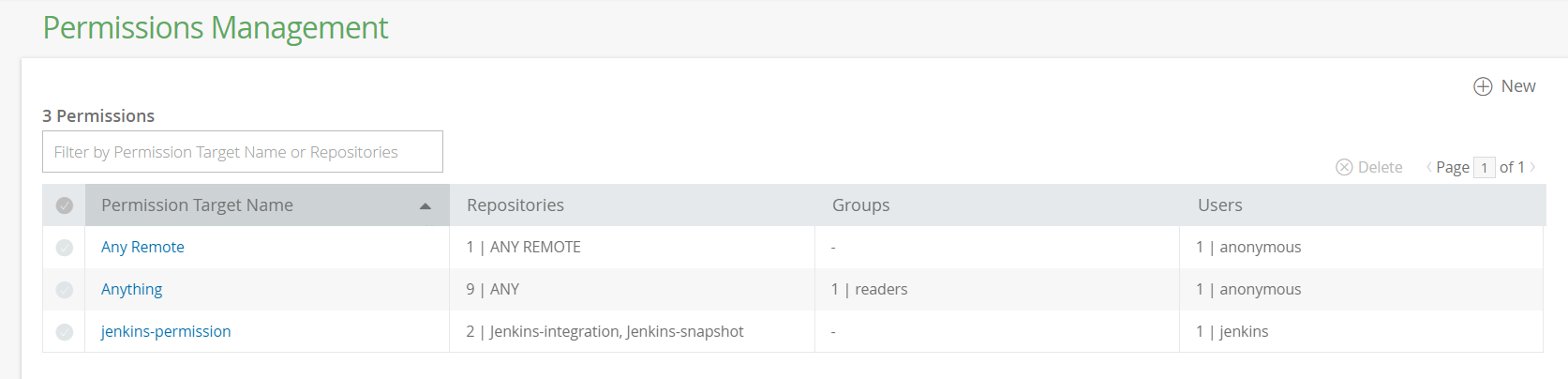












Now you are ready to integrate artifactory with Jenkins

Execute the sonarqube scanner for your maven project. Type the following commands in your folder.

mvn sonar:sonar \

-Dsonar.host.url=http://localhost:9000 \

-Dsonar.login= bfd22e40c2913e05d09377c288a51e2cfc521cf5

**d34f6f37664bb4e851a640bd8e37385b7d9df220**

mvn sonar:sonar \

-Dsonar.host.url=http://localhost:9000 \

-Dsonar.login=d34f6f37664bb4e851a640bd8e37385b7d9df220