**Semester: V Name of Student:**

**Academic Year: 2022-23 Student ID:**

**Class / Branch: TE IT**

**Subject: Advanced Devops Lab (ADL)**

**Name of Instructor:Prof. Manjusha Kashilkar**

**EXPERIMENT NO. 07**

**Aim: To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.**

**Steps:**

**1) Install and configure a Jenkins and SonarQube CICD environment using Docker containers.**

**2) Configure Jenkins with the SonarQube Scanner plugin for automated static code analysis.**

**1) Install and configure a Jenkins and SonarQube CICD environment using Docker containers.**

**Installation of Jenkins**

The version of Jenkins included with the default Ubuntu packages is often behind the latest available version from the project itself. To take advantage of the latest fixes and features, you can use the project-maintained packages to install Jenkins.

[**manjusha@apsit**](file:///C:\Users\manju\Downloads\manjusha@apsit)**:~$ wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -**

When the key is added, the system will return OK. Next, append the Debian package repository address to the server’s sources.list:

[**manjusha@apsit**](file:///C:\Users\manju\Downloads\manjusha@apsit)**:~$ sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'**

When both of these are in place, run update so that aptwill use the new repository:

[**manjusha@apsit**](about:blank)**:~$ sudo apt update**

Finally, install Jenkins and its dependencies:

[**manjusha@apsit**](about:blank)**:~$sudo apt install jenkins**

Let’s start Jenkins using systemctl:

[**manjusha@apsit**](about:blank)**:~$sudo systemctl start jenkins**

Since systemctl doesn’t display output, you can use its status command to verify that Jenkins started successfully:

[**manjusha@apsit**](about:blank)**:~$sudo systemctl status jenkins**

If everything went well, the beginning of the output should show that the service is active and configured to start at boot:

Now that Jenkins is running, let’s adjust our firewall rules so that we can reach it from a web browser to complete the initial setup.

**Opening the Firewall**

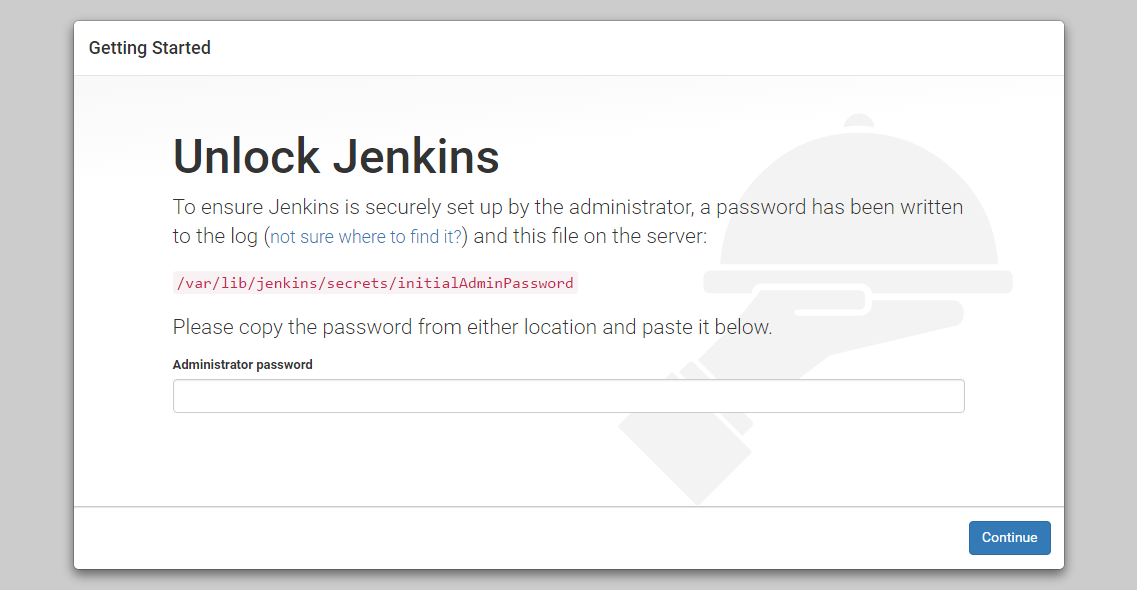
By default, Jenkins runs on port 8080, so let’s open that port using ufw:

[**manjusha@apsit**](about:blank)**:~$sudo ufw allow 8080**

**Setting Up Jenkins**

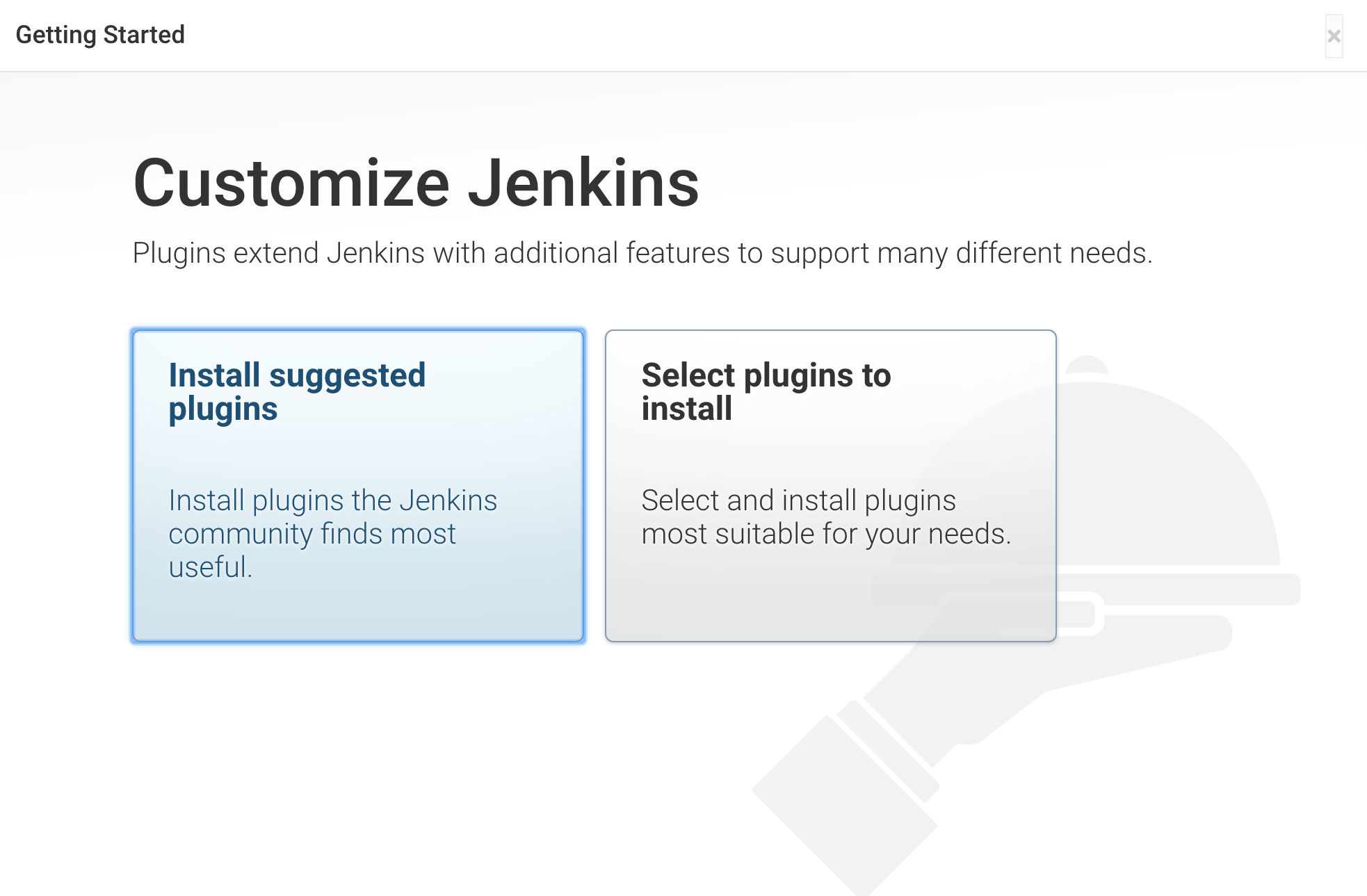
To set up your installation, visit Jenkins on its default port, 8080, using your server domain name or IP address: **http://your\_server\_ip\_or\_domain:8080**

You should see the Unlock Jenkins screen, which displays the location of the initial password:

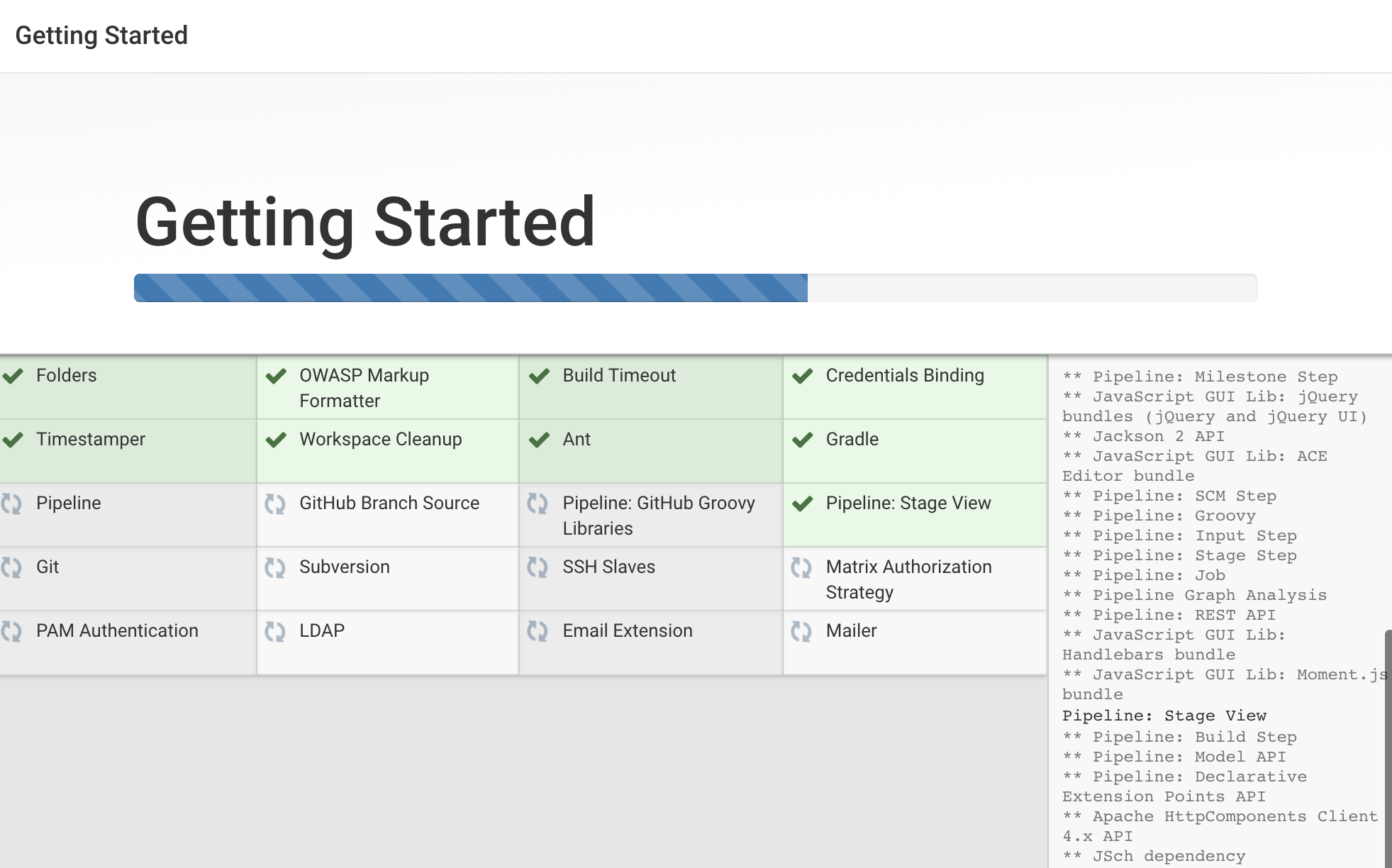
In the terminal window, use the cat command to display the password:

[**manjusha@apsit**](about:blank)**:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword**

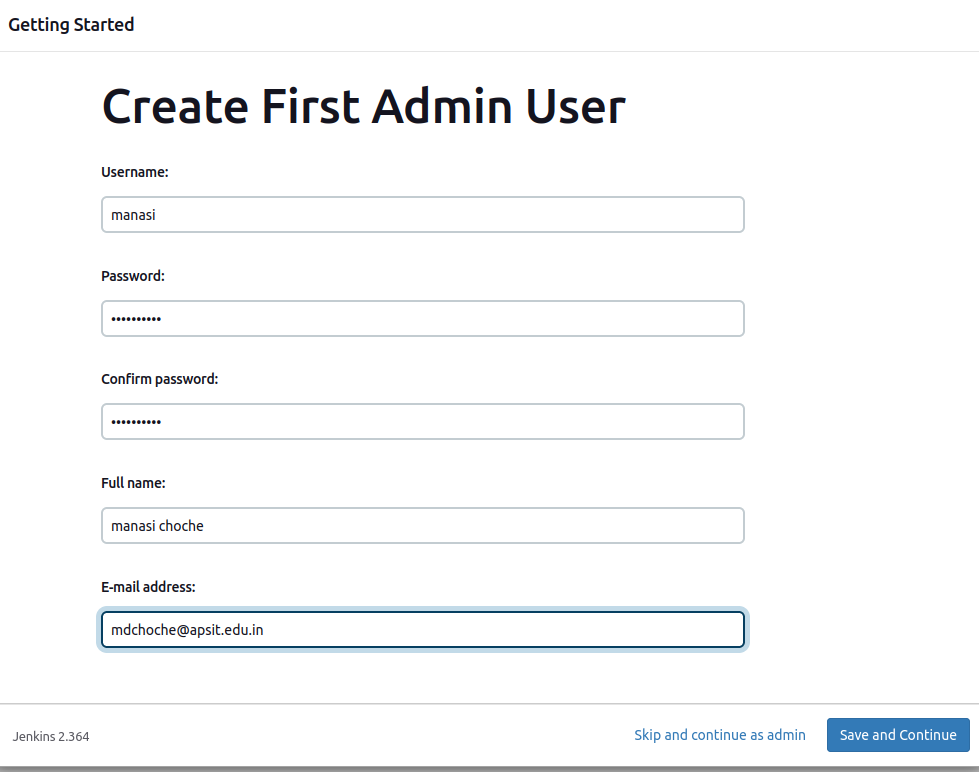
Copy the 32-character alphanumeric password from the terminal and paste it into the Administrator password field, then click Continue.

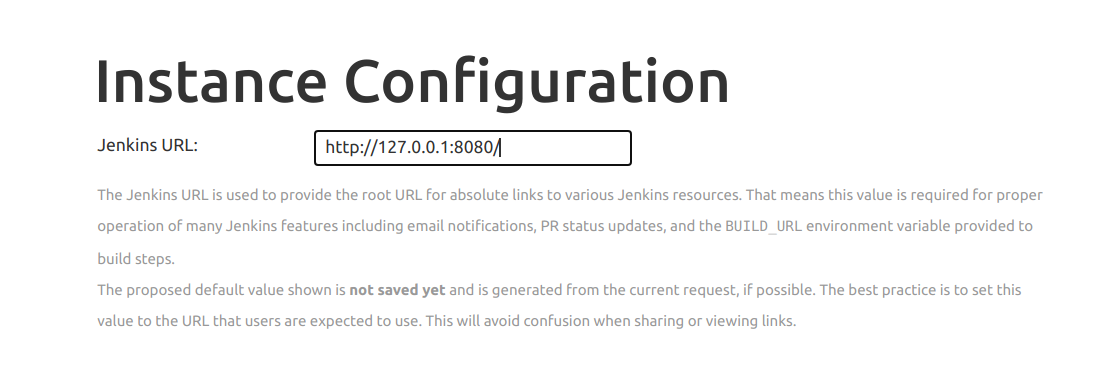
The next screen presents the option of installing suggested plugins or selecting specific plugins:

We’ll click the Install suggested plugins option, which will immediately begin the installation process:

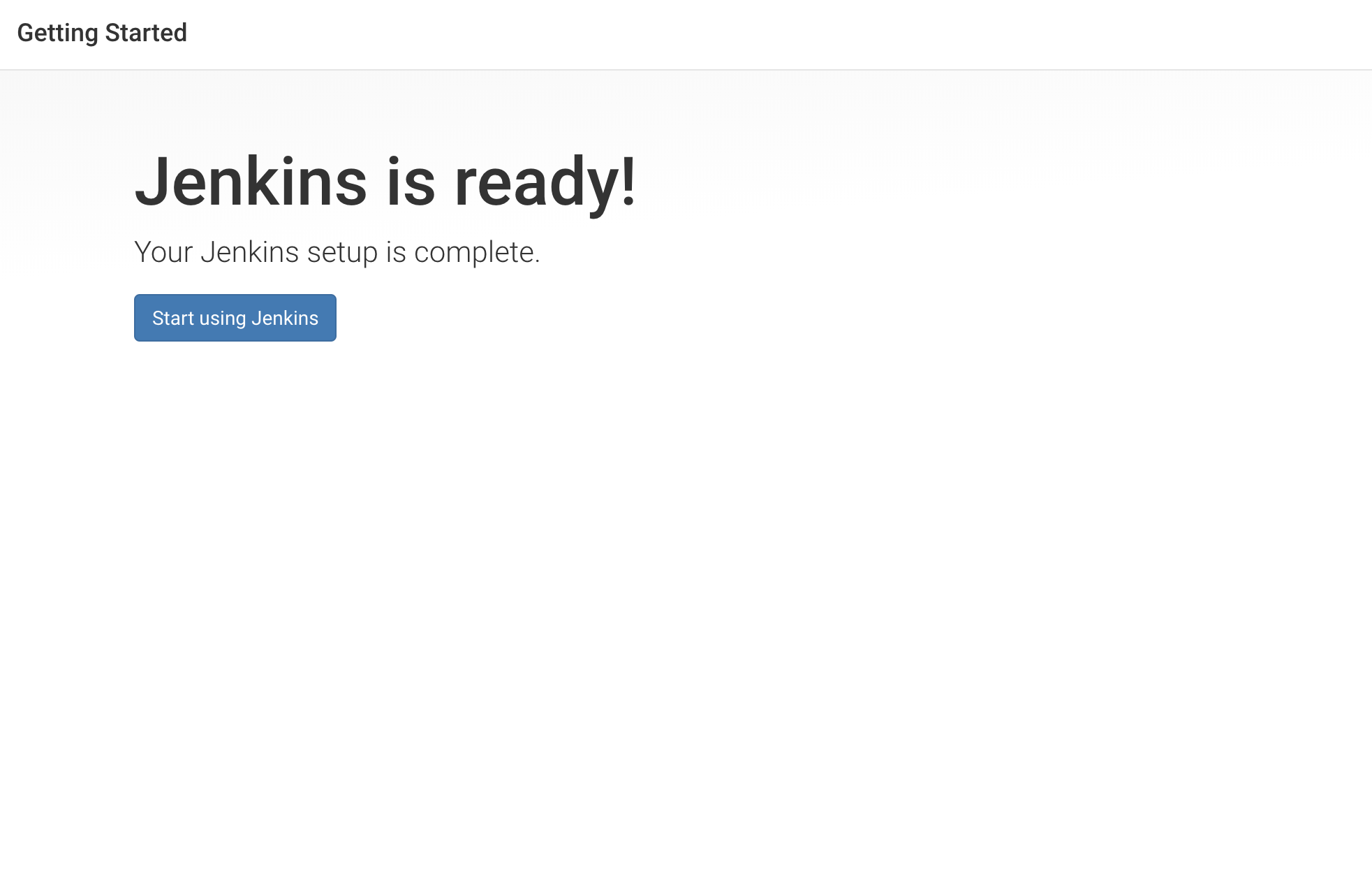


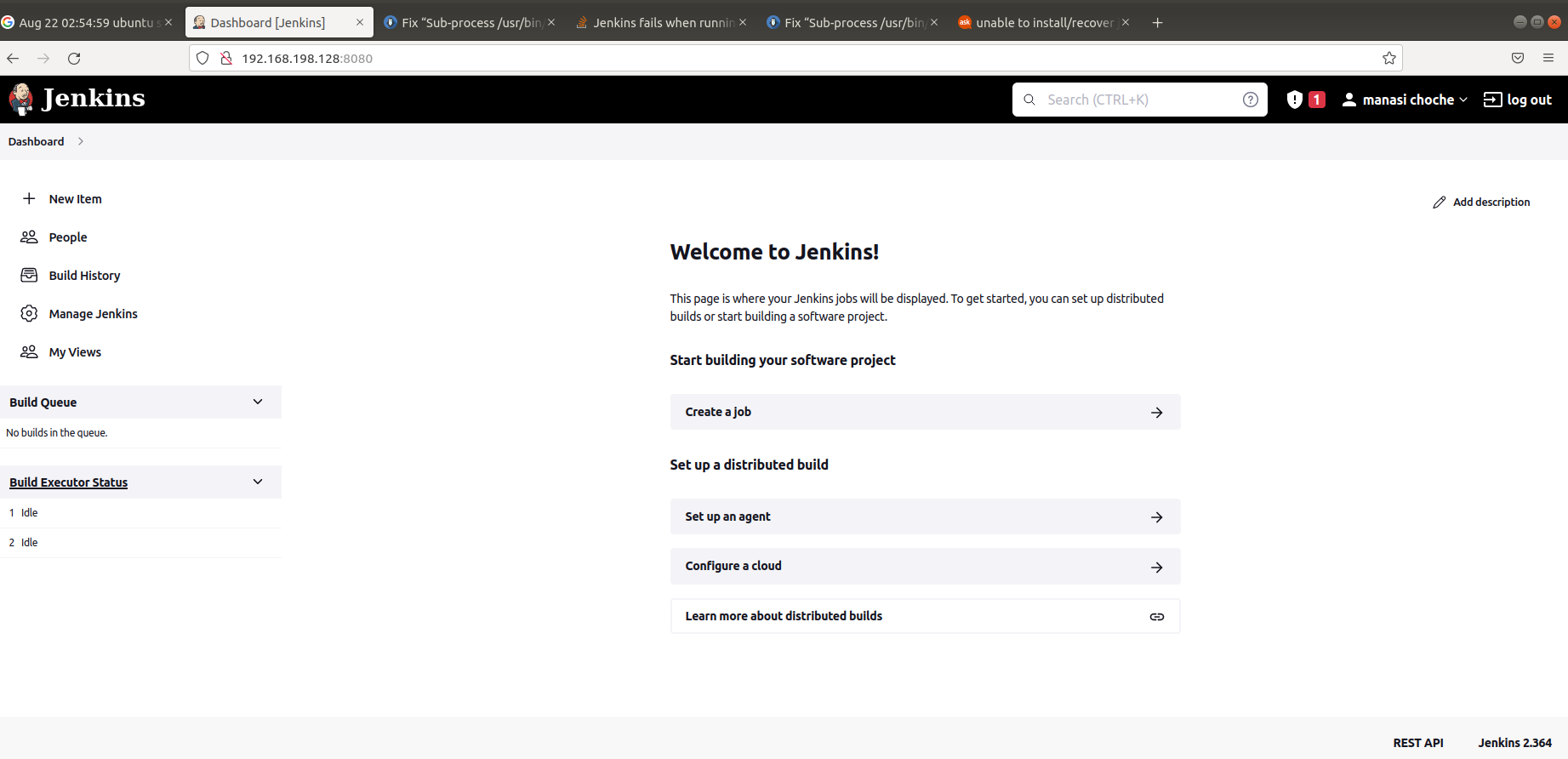
When the installation is complete, you will be prompted to set up the first administrative user. It’s possible to skip this step and continue as admin using the initial password we used above, but we’ll take a moment to create the user.





After confirming the appropriate information, click Save and Finish. You will see a confirmation page confirming that “Jenkins is Ready!”:

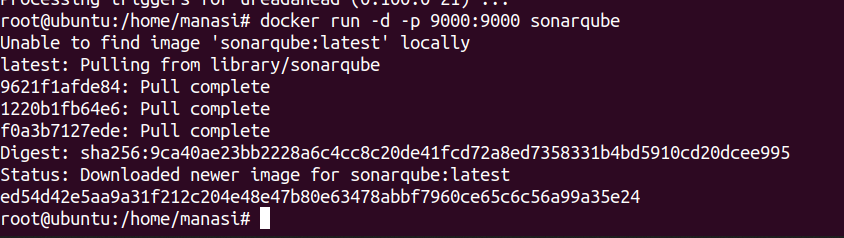
Click Start using Jenkins to visit the main Jenkins dashboard:



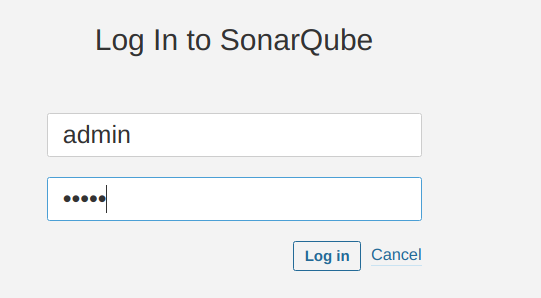
**SonarQube Setup**

Before proceeding with the integration, we will setup SonarQube Instance. we are using SonarQube Docker Container.

[**manjusha@apsit**](about:blank)**:~$docker run -d -p 9000:9000 sonarqube**



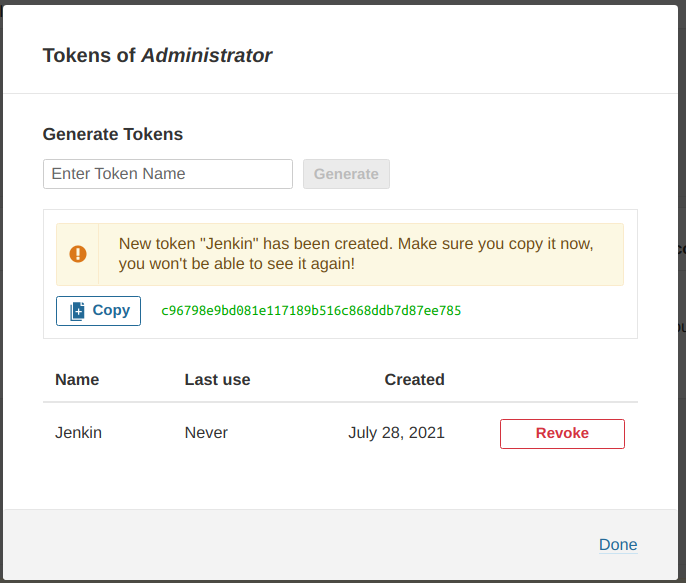
In the above command, we are forwarding port 9000 of the container to the port 9000 of the host machine as SonarQube is will run on port 9000. Then, from the browser, enter [http://localhost:9000](http://localhost:9000/). After That, you will see the SonarQube is running. Then, login using default credentials (admin:admin).



### Generate User Token

Now, we need to get the SonarQube user token to make connection between Jenkins and SonarQube. For the same, go to **Administration> User > My Account > Security** and then, from the bottom of the page you can create new tokens by clicking the Generate Button. Copy the Token and keep it safe.

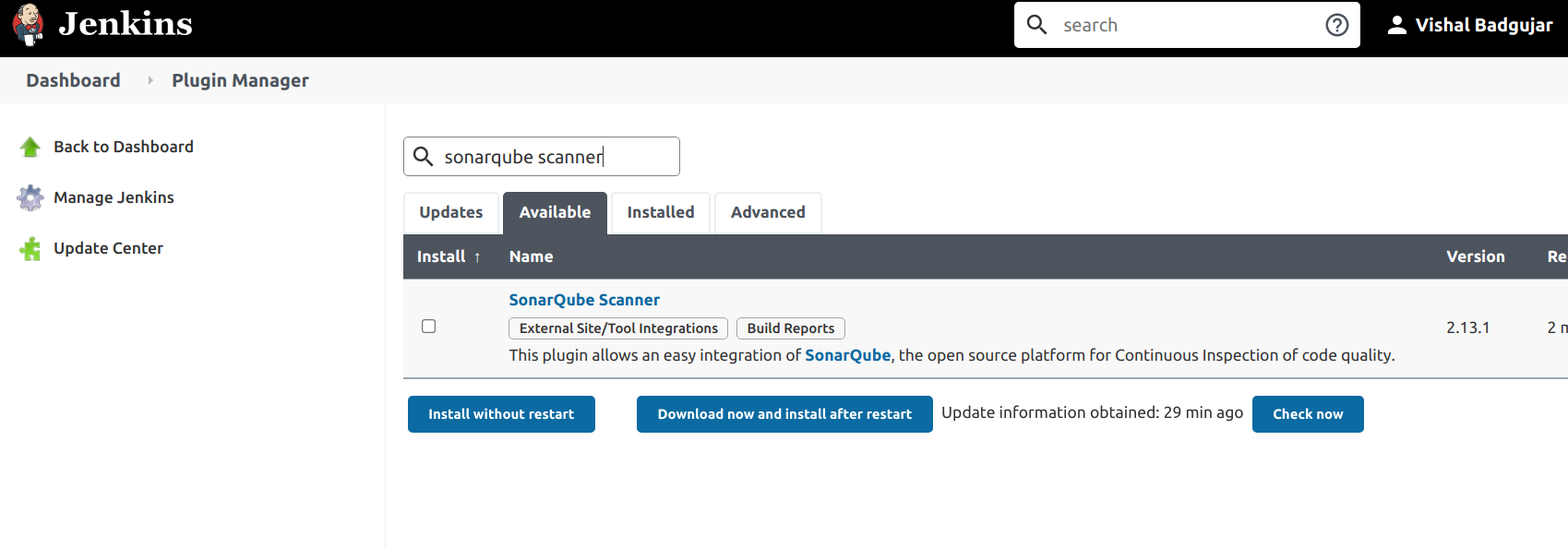
**C96798e9bd081e117189b516c868ddb7d87ee785 SonarQube**



**2) Configure Jenkins with the SonarQube Scanner plugin for automated static code analysis.**

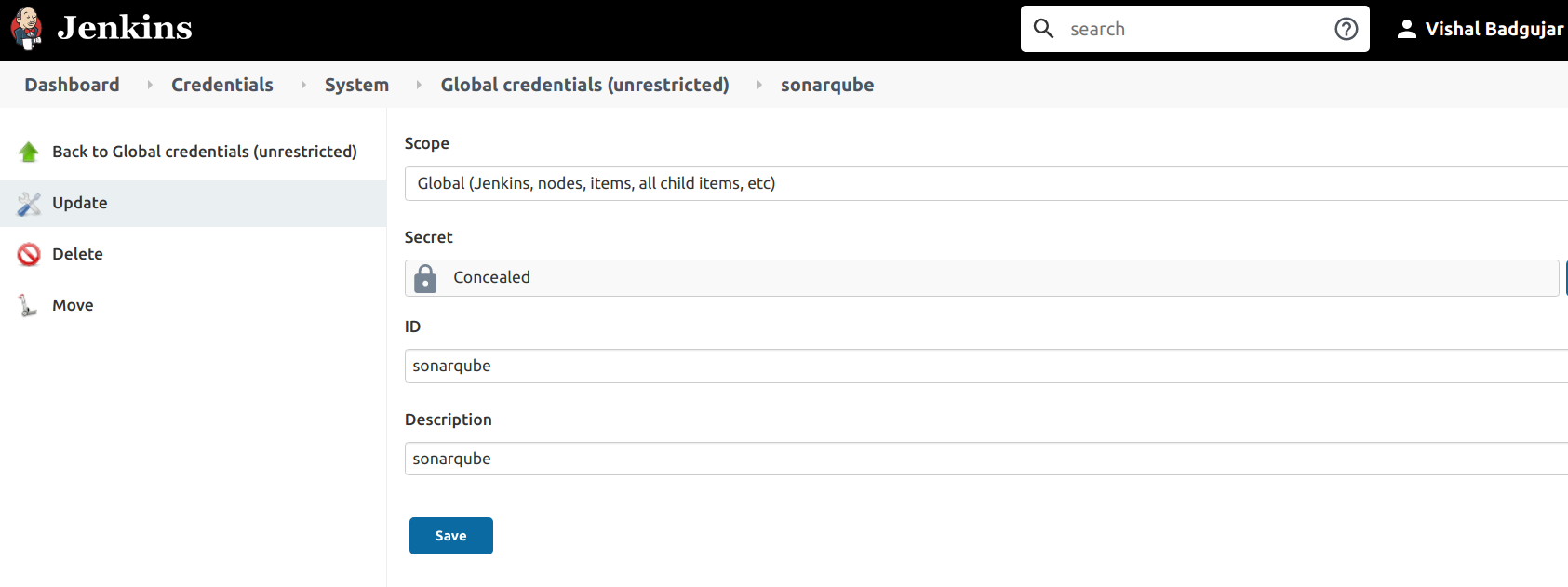
## Jenkins Setup for SonarQube

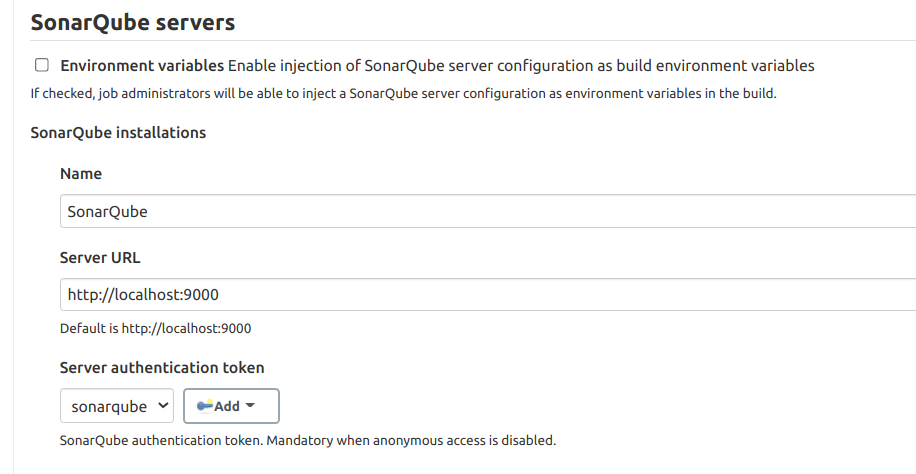
Before all, we need to install the SonarQube Scanner plugin in Jenkins. For the same, go to **Manage Jenkins > Plugin Manager > Available.** From here, type SonarQube Scanner then select and install.



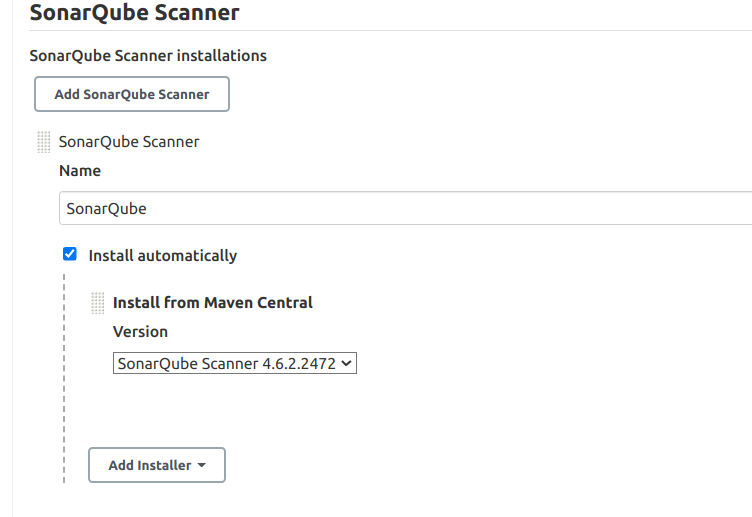
### Tool Configuration SonarQube Scanner

Now, we need to configure the Jenkins plugin for SonarQube Scanner to make a connection with the SonarQube Instance. For that, got to **Manage Jenkins > Configure System > SonarQube Server.** Then, Add SonarQube. In this, give the Installation Name, Server URL then Add the Authentication token in the Jenkins Credential Manager and select the same in the configuration.



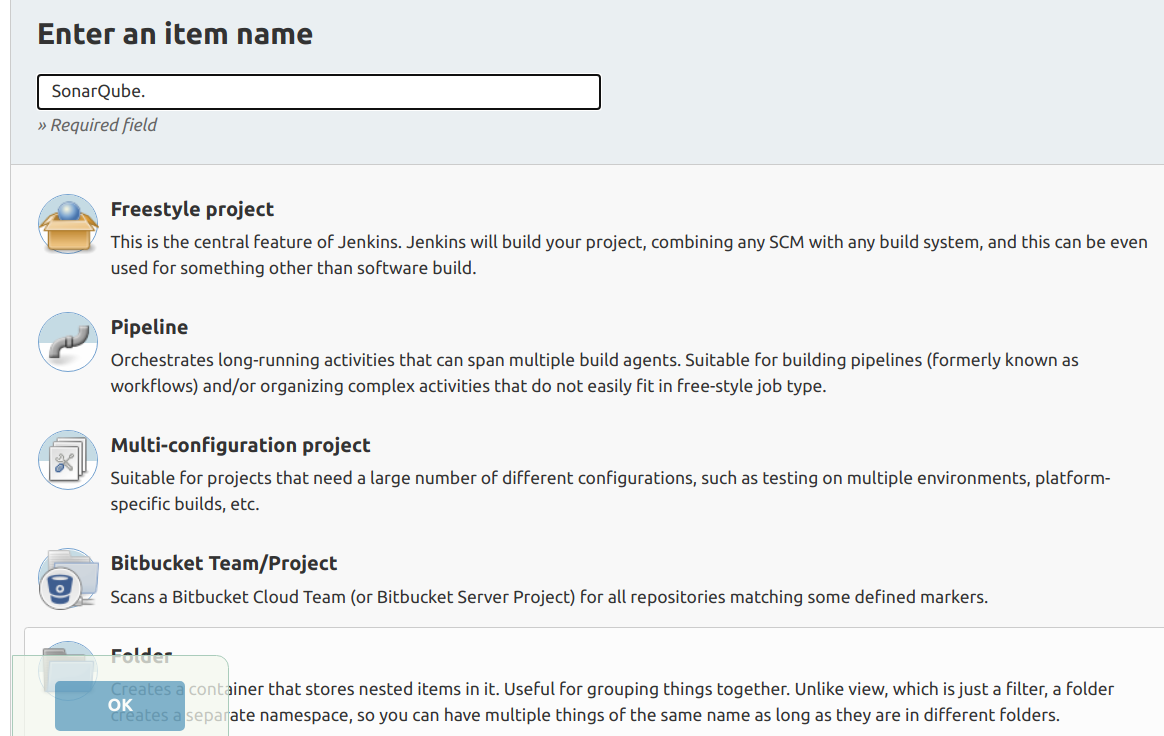


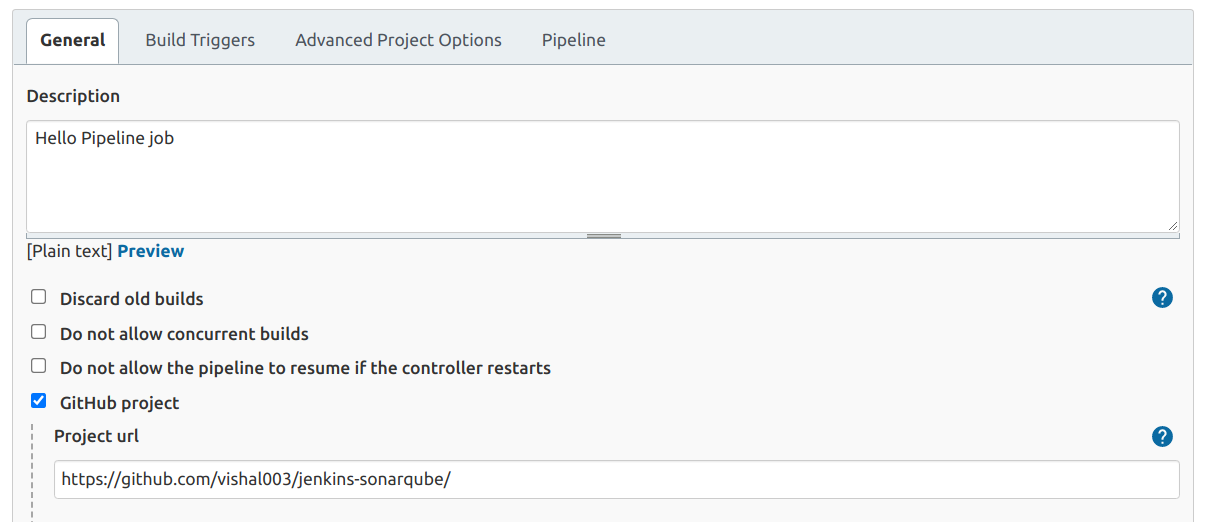
Then, we need to set-up the SonarQube Scanner to scan the source code in the various stage. For the same, go to **Manage Jenkins > Global Tool Configuration > SonarQube Scanner**. Then, Click **Add SonarQube Scanner Button**. From there, give some name of the scanner type and **Add Installer** of your choice. In this case, I have selected SonarQube Scanner from Maven Central.



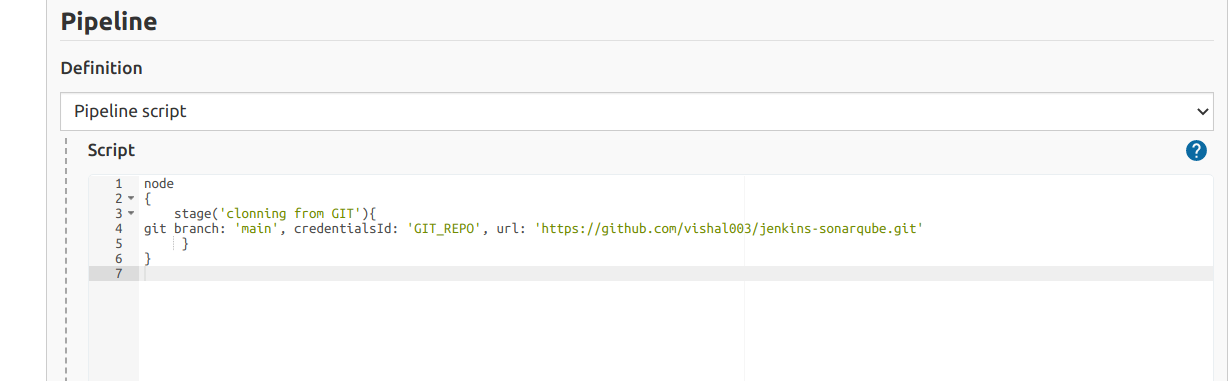
### SonarQube Scanner in Jenkins Pipeline

Now, It’s time to integrate the SonarQube Scanner in the Jenkins Pipeline. For the same, we are going to add one more stage in the Jenkinsfile called SonarQube and inside that, I am adding the following settings and code.

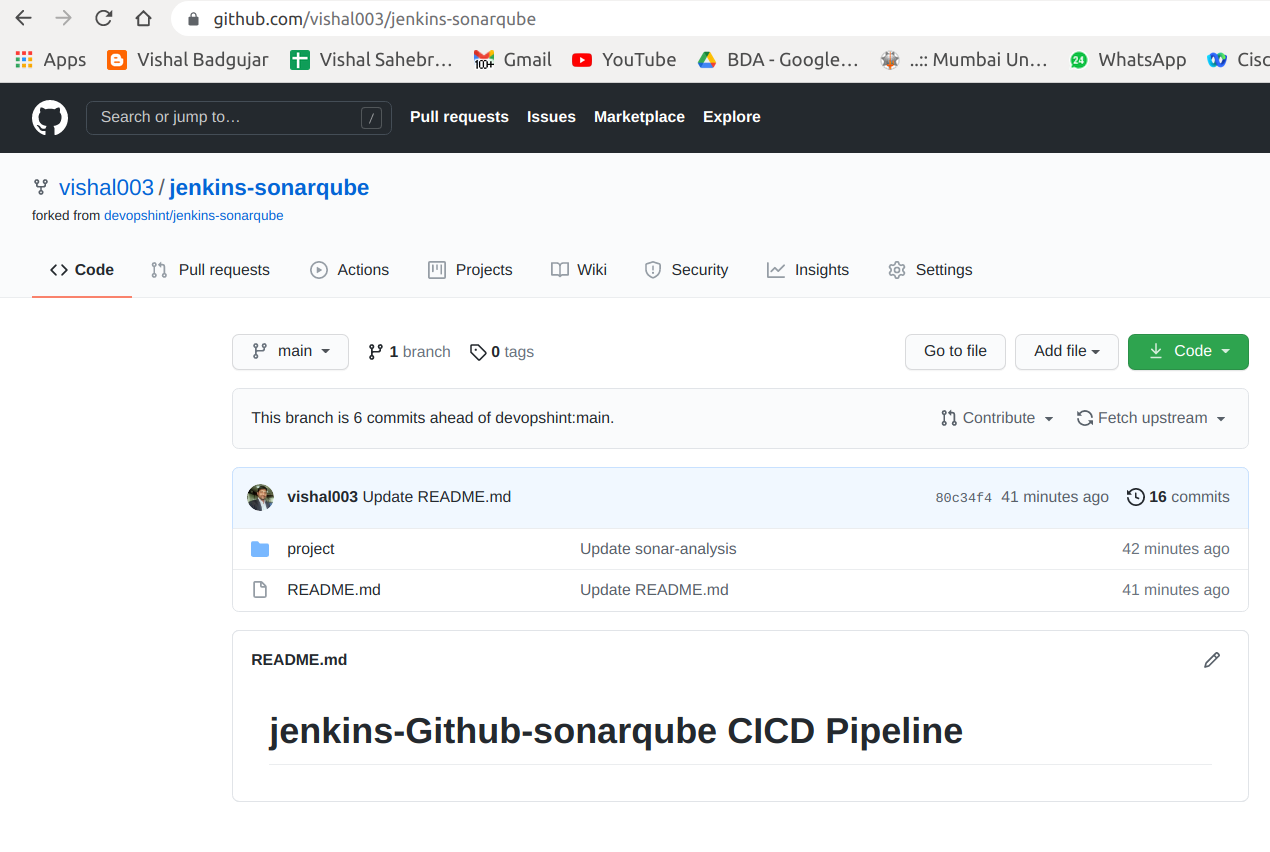




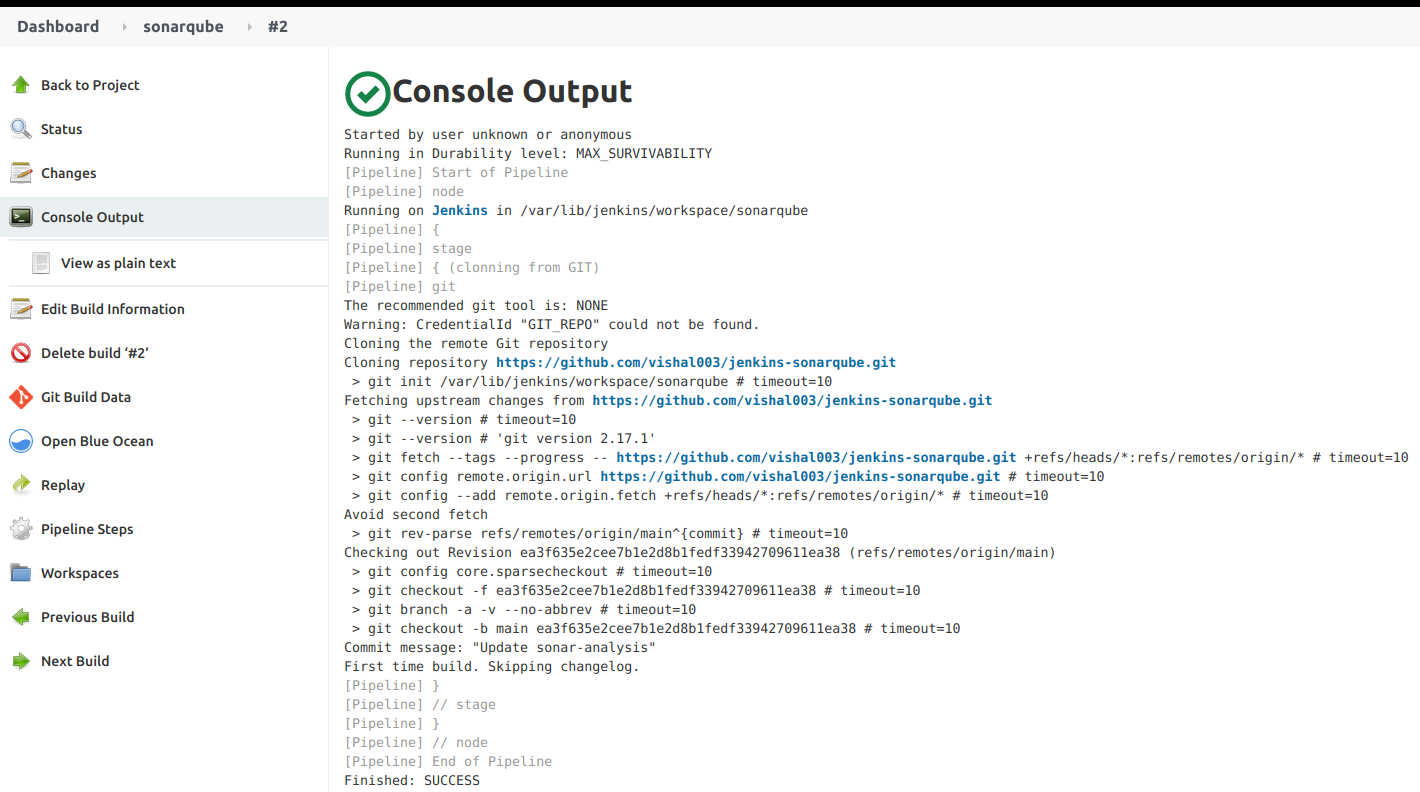
**Github Configuration in Jenkins Pipeline**



**Git Clonning into Jenkins**



**Github Repository Contents**

Successfully Build Github Repository in Jenkins

**Pre-requiste required for Integration settings of Jenkins SAST with SonarQube we have done here successfully, now in order to Integrate of Jenkins CICD with SonarQube with the help of sample JAVA program we will implement in next experiment.**

**Conclusion: Write your own findings.**