**How to Install Jenkins**

**Pre-requisite**:-

Java should be installed . If not installed please follow below steps.

sudo yum install java-1.8.0-openjdk-devel

**Jenkins Installation**

1. Enable Jenkins repo using below command

curl --silent --location http://pkg.jenkins-ci.org/redhat-stable/jenkins.repo | sudo tee /etc/yum.repos.d/jenkins.repo

1. Add the repo to system

sudo rpm --import <https://jenkins-ci.org/redhat/jenkins-ci.org.key>

1. Once repo is is added to system install Jenkins using below command

sudo yum install Jenkins

1. Start Jenkins using below command

sudo systemctl start Jenkins

1. systemctl status Jenkins :--- run this command to check status of Jenkins
2. Jenkins is up and running now and you can hit it on port 8080.
3. It will prompt for a alphanumeric key which can be found at :-

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

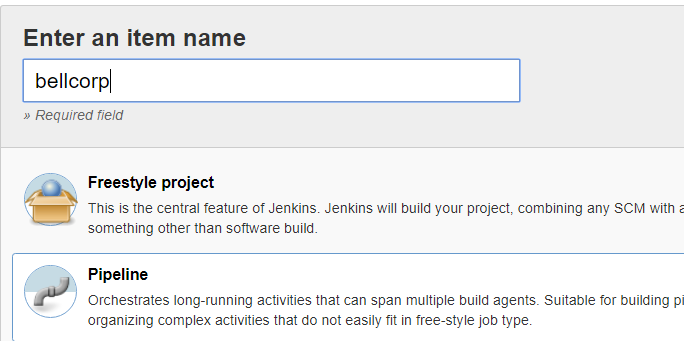
1. Create a user for yourself and install default plugins to start

**Steps to Create Pipeline Project:-**

1. Click on New Item



1. Select Pipeline project and give a name for project, then click ok



1. Add the Jenkins file and run the job.

**Jenkins Integration with Slack and Sonarqube:**  
**Integration with Slack Channel:**

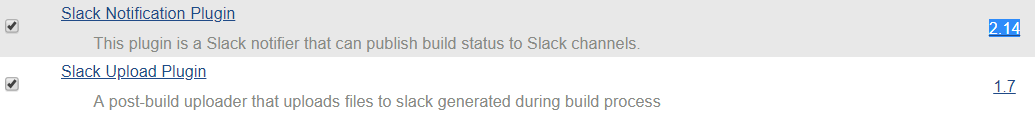
**Installing Slack Plugin in Jenkins ver. 2.138:**

➟ Install [**Slack Notification Plugin**](http://wiki.jenkins-ci.org/display/JENKINS/Slack+Plugin)

➟ Go to Manage Jenkins → Manage Plugins → Available Tab → Search for “[Slack Notification Plugin](http://wiki.jenkins-ci.org/display/JENKINS/Slack+Plugin)” → Install

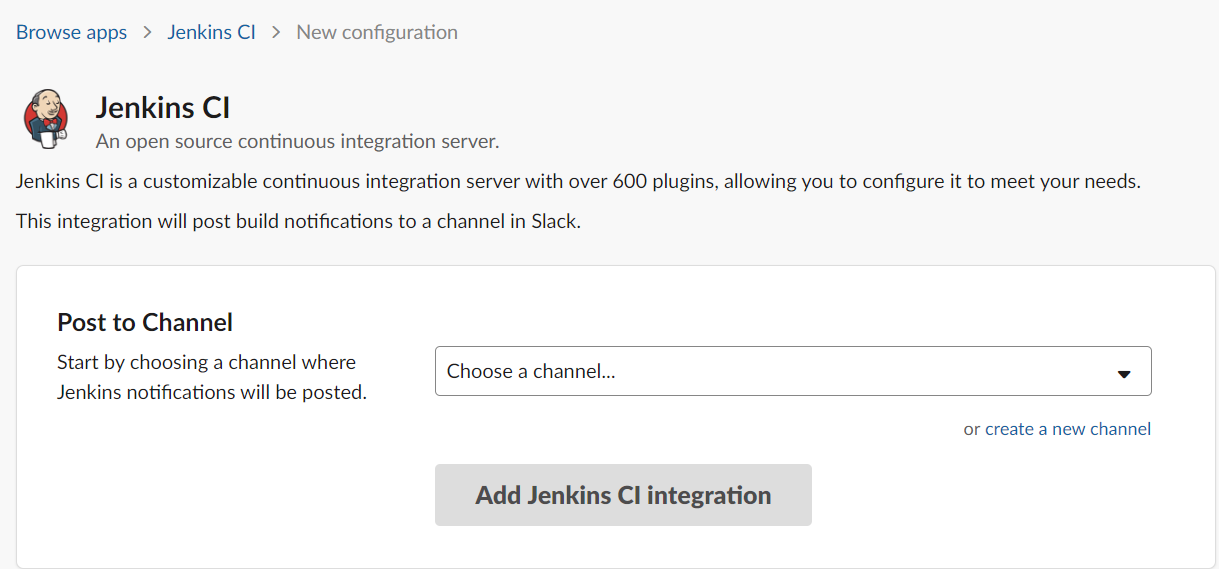
➟ Current installed version of Slack Notification Plugin is 2.14

➟ Incase we want to upload build related files to Slack channel we can install “Slack Upload Plugin”



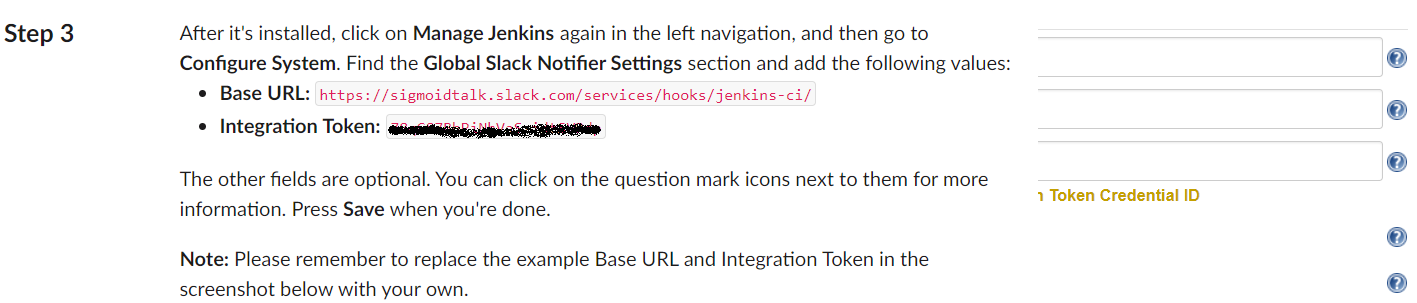
**Creating Slack channel and generating Integration Token for Jenkins:**  
➟ Create a Slack account  
➟ Go to http://<your-slack-space>.slack.com**/services/new/jenkins-ci**

➟ This link should take you to the below attached page for configuring Jenkins CI to Slack



➟ Under “Post to Channel” option, from the drop down menu select the Slack Channel you want to publish build reports to.  
➟ Now click on “Add Jenkins CI integration” which shall redirect you to a detailed webpage guide on how to Install Slack Notification Plugin and provides “Base URL” and “Integration Token” details which further needs to be feeded in to Jenkins at Configure System → Global Slack Notifier Settings

➟ Details of Base URL and “Integration Token” are provided in “Step 3” section of the webpage loaded above after clicking on “Add Jenkins CI integration”.

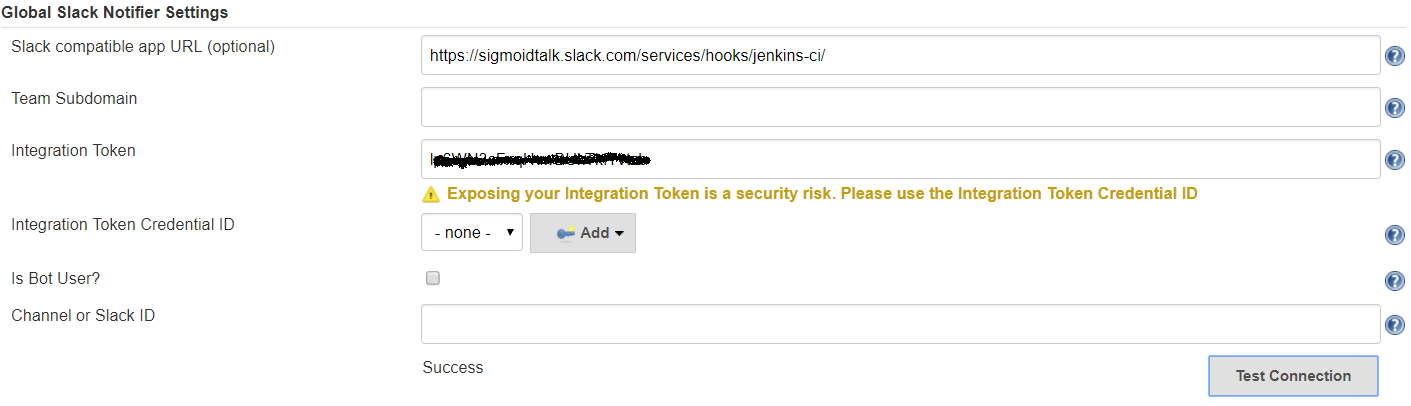


**Adding Slack Notifier settings to Jenkins ver. 2.138:**

➟ Go to Manage Jenkins → Configure System → Global Slack Notifier Settings

➟ Enter Slack app url as mentioned below along with Integration Token generated from Slack

➟ Once the details are entered we can check the configuration by clicking on “Test Connection” which should return a success output as depicted below.



**Integration with SonarQube:**

**Installing and configuring MySQL as a Database for SonarQube on Linux (CentOS/RHEL):**

➟ Install and configure MySQL  
# wget <http://repo.mysql.com/mysql-community-release-el7-5.noarch.rpm>

# rpm -ivh mysql-community-release-el7-5.noarch.rpm (Install mysql from rpm file)

# yum update -y (Update installed packages on the OS)

# yum install mysql-server (Install MySQL server)

# systemctl start mysqld (Start mysql service on the system)  
# mysql\_secure\_installation (Setup password for your MySQL instance)  
# mysql -u root -p (Login to the MySQL instance)

➟ Check default storage engine of MySQL being used which should be innoDB  
mysql> SHOW GLOBAL VARIABLES LIKE 'storage\_engine';

+----------------+--------+

| Variable\_name | Value |

+----------------+--------+

| storage\_engine | InnoDB |

+----------------+--------+

1 row in set (0.01 sec)

➟ Create a Database using MySQL CLI and a sonarqube user.  
mysql> CREATE DATABASE sonarqube;

mysql> CREATE USER 'sonarqube'@'localhost' IDENTIFIED BY 'password';

➟ Grant all privileges on sonarqube database to the sonarqube user

mysql> GRANT ALL PRIVILEGES ON sonarqube.\* TO 'sonarqube'@'localhost';

➟ Open /etc/my.cnf file and under [mysqld] section add the query cache parameter. The minimum size should be 15 MB. You can increase the size based on your server type.

# vi /etc/my.cnf

query\_cache\_size = 15M

**Installing and configuring SonarQube server on Linux:**  
➟ Download the SonarQube installation file to the location you prefer

➟ Make sure Java is installed on the system if not install Java using the below command  
# yum install -y java-1.8.0-openjdk.x86\_64  
# cd /tmp  
# wget <https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-6.7.6.zip>

# unzip sonarqube-6.7.6.zip (unzip sonarqube zip file downloaded)

# mv sonarqube-6.7.6 sonarqube (rename the sonarqube folder for ease of use)

➟ Open sonar.properties file and configure details related to mysql and sonar webpage  
# cd /tmp/sonarqube/conf  
# vi sonar.properties  
sonar.jdbc.username=sonarqube ## Specify mysql user created while configuring MySQL

sonar.jdbc.password=password ## Specify the password setup for the MySQL user

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

## Setup java memory options to avoid HeapDump OOM error

sonar.web.javaOpts=-Xmx1024m -Xms1024m -XX:+HeapDumpOnOutOfMemoryError  
sonar.web.host=0.0.0.0 ## You can specify 0.0.0.0 for the web GUI or your own public IP

## By default sonarQube runs on port 9000 If you want on port 80 or any other port,change the following parameters for accessing the web console on that specific port.

sonar.web.port=9000

sonar.web.context=/sonar ## To access sonarQube on some path like http://url:/sonar

**Start SonarQube Service:**  
➟ To be able to start the sonar service we need to run the sonar.sh script at the bin directory

# cd /tmp/sonarqube/bin/linux-x86-64  
# ./sonar.sh start  
# ./sonar.sh

Usage: ./sonar.sh { console | start | stop | restart | status | dump }

# ./sonar.sh status  
SonarQube is running (1900).

**Create a SonarQube service:**  
➟ Create a systemd service for sonar to be able to manage the sonar service efficiently and easily  
# cd /etc/systemd/system  
# vi sonar.service  
[Unit]

Description=SonarQube service

After=syslog.target network.target

[Service]

Type=forking

ExecStart=/tmp/sonarqube/bin/linux-x86-64/sonar.sh start

ExecStop=/tmp/sonarqube/bin/linux-x86-64/sonar.sh stop

User=sonarqube

Group=sonar

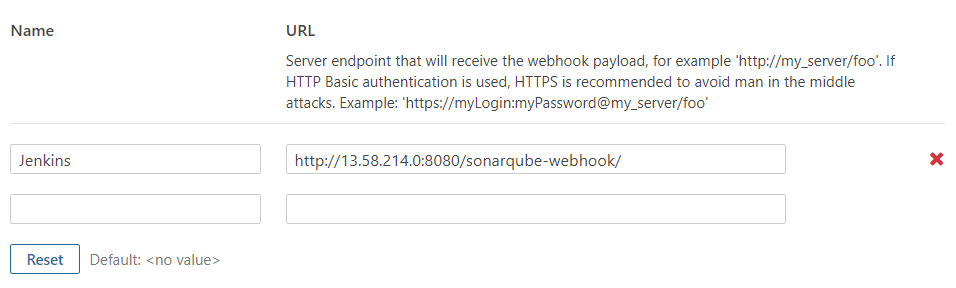
Restart=always

[Install]

WantedBy=multi-user.target

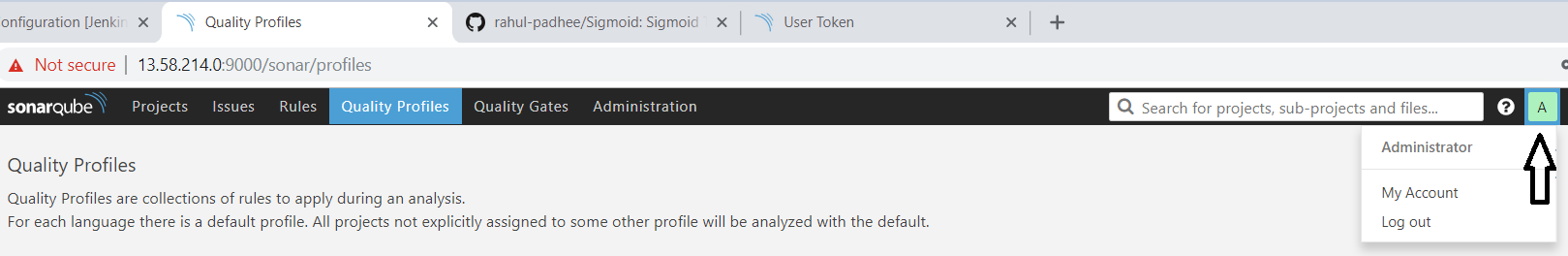
➟ Start the SonarQube application and enable it across boot  
# systemctl start sonar  
# systemctl enable sonar  
# systemctl status sonar

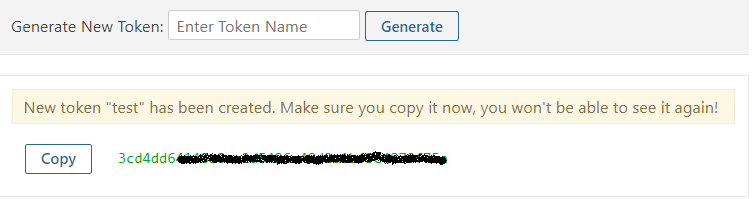
SonarQube service creation documentation:  
<https://docs.sonarqube.org/latest/setup/operate-server/>

**SonarQube UI configuration for Jenkins:**  
➟ Login to SonarQube url @ [http://IP:9000/sonar](http://ip:9000/sonar)  
➟ Default login username:password for SonarQube is admin:admin  
➟ Configure webhook for Jenkins in SonarQube server. Webhooks are used to notify external services (ex: Jenkins) when a project analysis is done.  
➟ Go to SonarQube homepage → Administration → Webhooks  


http://<JenkinsURL>:/8080/sonarqube-webhook/

**Generating Server Authentication Token from SonarQube UI for Jenkins:**  
➟ Click on your login account on the top right corner of the webpage

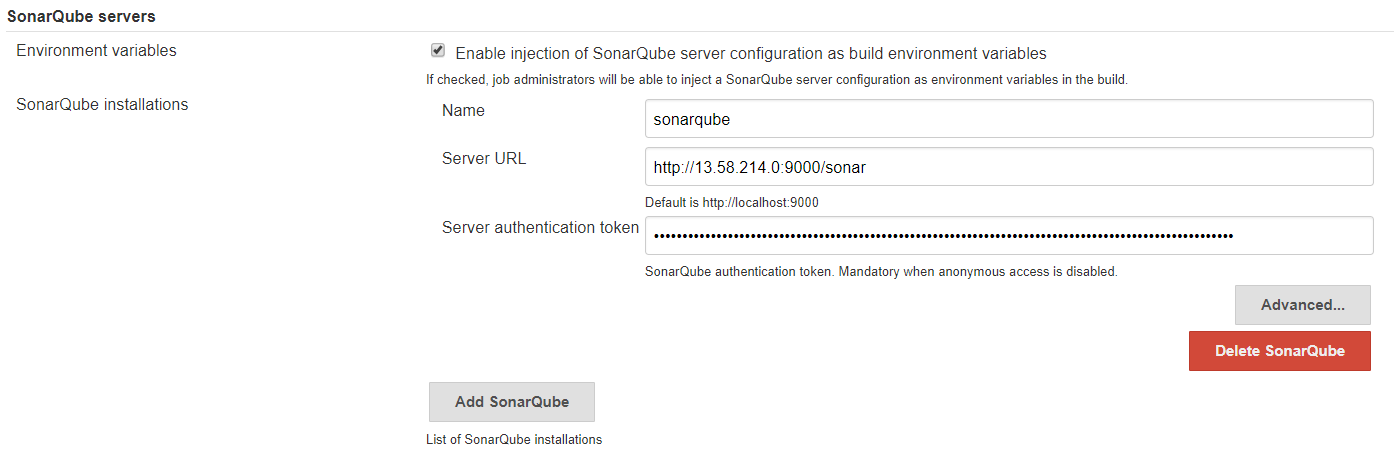


➟ Then click on “My Account” → Now do click on the “Security” Tab → Under Tokens Section enter a name for the token and click on Generate  


➟ Make sure to copy the token and keep it handy as it would not be available the next time.

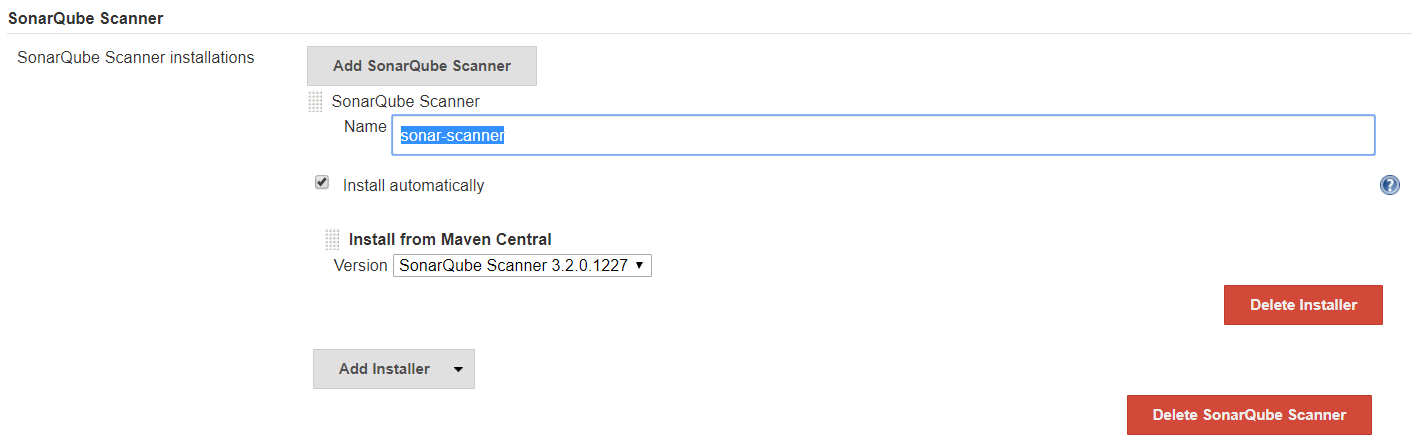
**Adding SonarQube server details to Jenkins:**

➟ Go to Manage Jenkins → Configure System → SonarQube Servers  
➟ Provide the name of the SonarQube server which is further specified in the Jenkinsfile as  
withSonarQubeEnv('sonarqube')  
➟ Furnish details of SonarQube Server URL which was configured on the host OS  
➟ Enter the Server authentication token generated from Sonarqube UI.



**Configuring SonarQube Scanner in Jenkins:**➟ Go to Manage Jenkins → Global Tool Configuration → SonarQube Scanner  
➟ Under SonarQube Scanner click on “Add SonarQube Scanner”  
➟ Provide a name to the SonarQube Scanner which further needs to be specified as an environment variable to the SonarQube analysis stage, in the Jenkinsfile.

environment { scannerHome = tool 'sonar-scanner' }



**Some considerations when trying to execute SonarQube service on the system:**➟ **SonarQube cannot be run as the user root and hence we need to create an user, say “sonarqube” user which can be used to run the sonarqube server**  
# groupadd sonar (create a group named sonar for the sonarqube user)  
# useradd -c "Sonar System User" -d /tmp/sonarqube -g sonar -s /bin/bash sonar

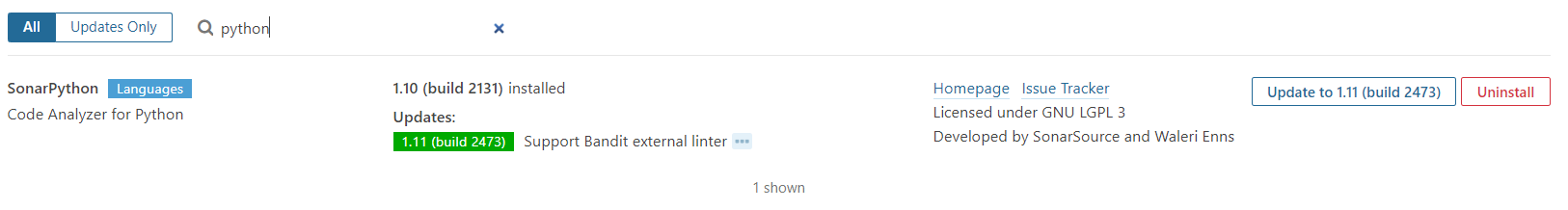
# chown -R sonarqube:sonar /tmp/sonarqube/

# cd /tmp/sonarqube/bin/linux-x86-64/

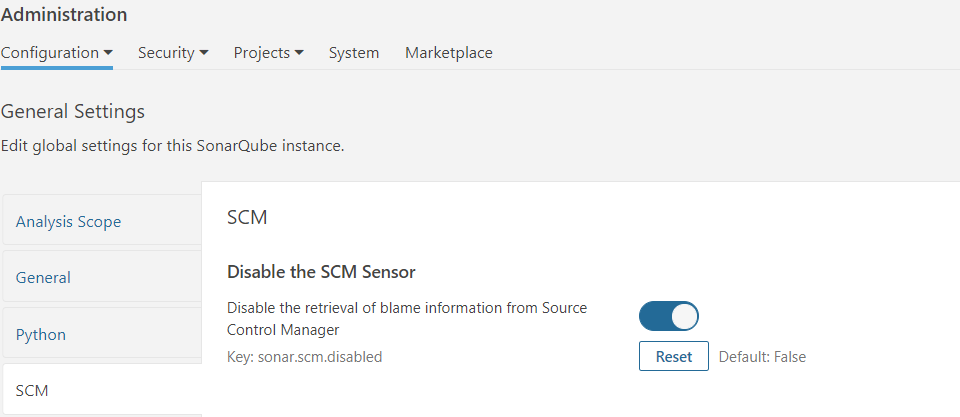
# vi sonar.sh (edit sonar start script and specify sonarqube user as RUN\_AS\_USER)

RUN\_AS\_USER=sonarqube

➟ **When running the Jenkins CI pipeline, the sonar scanner analysis stage might fail with the below errors:**  
ERROR: Error during SonarQube Scanner execution  
ERROR: No quality profiles have been found, you probably don't have any language plugin installed.  
ERROR:   
ERROR: Re-run SonarQube Scanner using the -X switch to enable full debug logging.

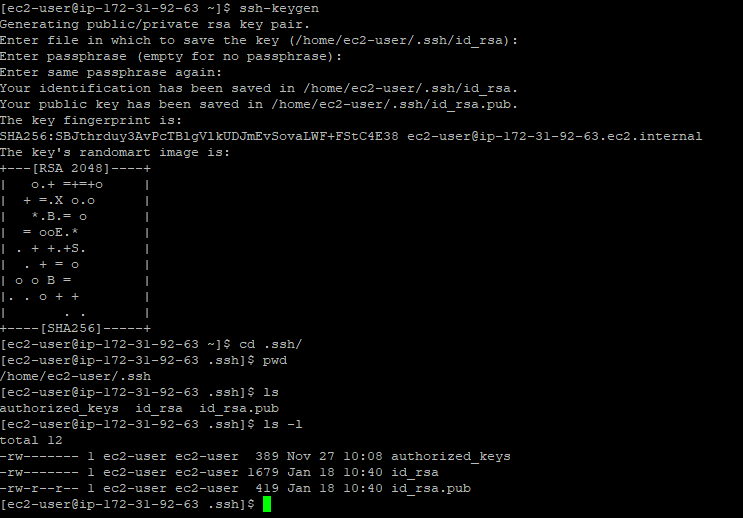
**CAUSE:** SonarPython i.e. Code Analyzer for Python is not installed  
**RESOLUTION:**   
Go to SonarQube Homepage → Administration → Marketplace → Search for Python in the search box as depicted below and click on Install  


➟ **During SonarQube Scanner execution stage, the CI process might fail because of the below errors:**  
ERROR: Error during SonarQube Scanner execution  
ERROR: SCM provider was set to "git" but no SCM provider found for this key. No SCM provider installed

**RESOLUTION:** Goto SonarQube homepage → Administration → Click on Configuration drop down and click on General Settings → Go to SCM and disable the SCM Sensor  
  


**Private GIT repository integration with Jenkins:**

➟ Generate a public-private key using puttygen or ppk.



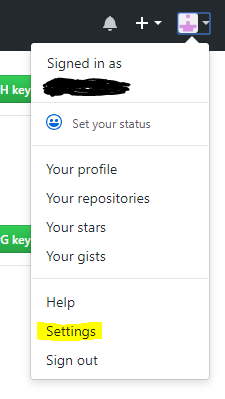
The keys generated are placed in /home/<respective-user>/.ssh.

This location can be changed.

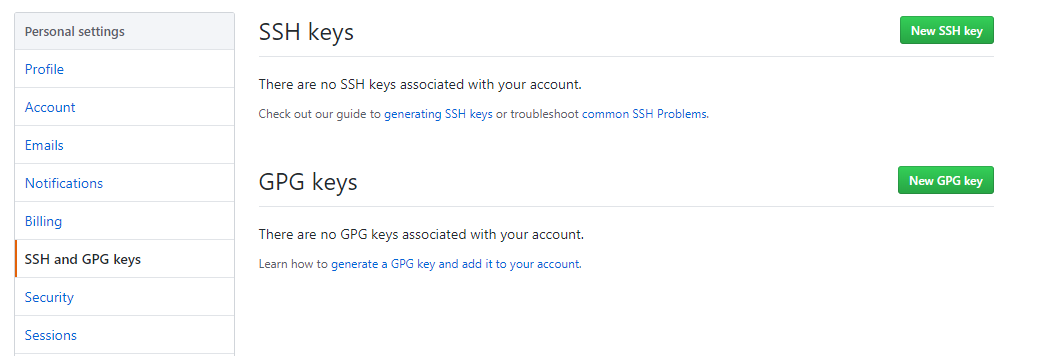
PEM:- id\_rsa

Public key:- id\_rsa.pub

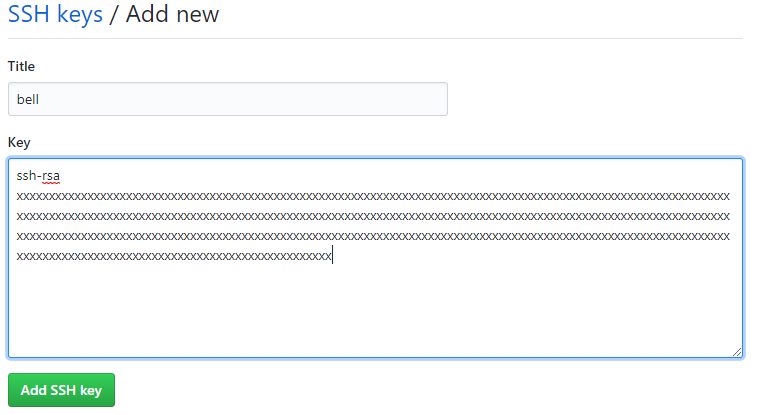
1. Go to settings in Git repo



1. Go to SSH and GPG keys setting



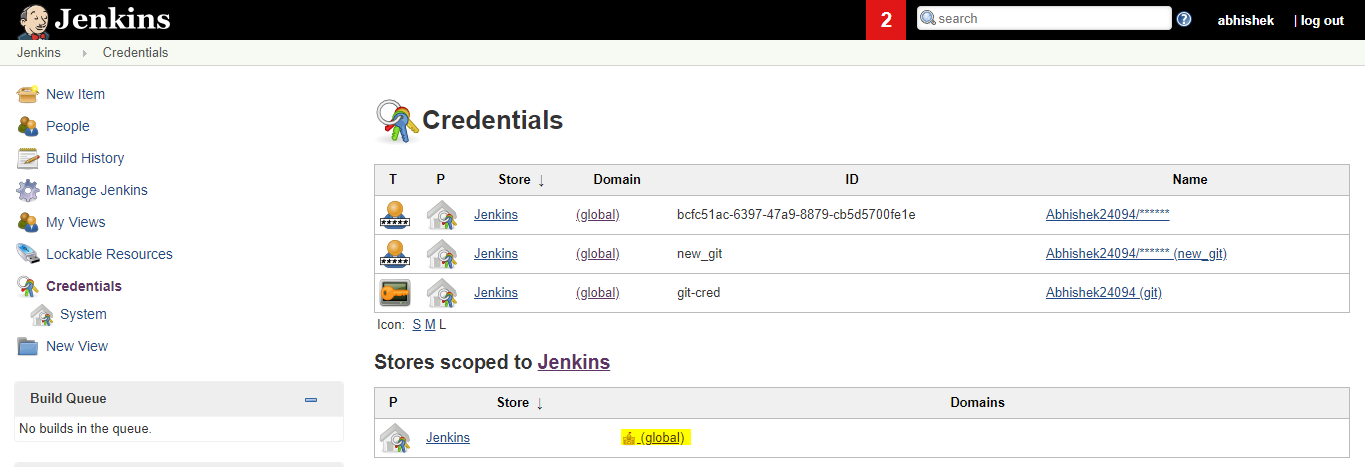
1. Click on New SSH key and add the public key generated above(id\_rsa.pub) and then click on Add SSH key

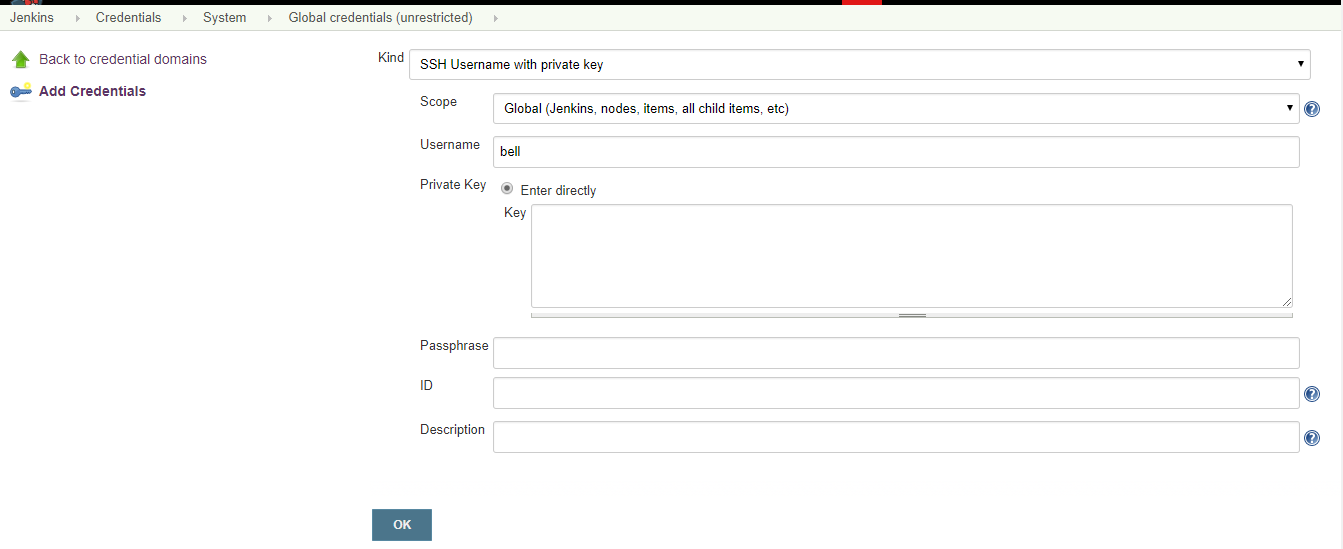


1. Login to Jenkins and go to credentials in left panel

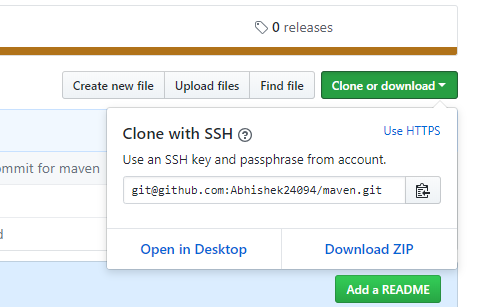


1. Click on global and click on add credentials in left panel

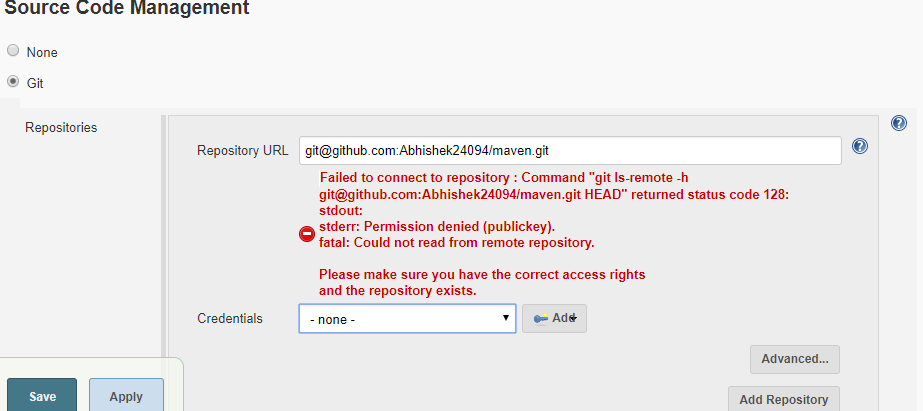




1. Choose SSH Username option from the Kind drop down and pass the private key that we created earlier
2. Click Ok
3. Check you Git repo connectivity by passing the repo **SSH** not **HTTPS** url as we are using ssh for connectivity. You can get SSH url from clone download option in git repo



1. We might get below kind of error if no credentials or wrong credentials are provided



1. Upon passing proper credentials errors will be gone and private repo will be integrated with Jenkins.

