**IMPLEMENTATION OF** **CI/CD PIPELINE PROJECT WITH HELP OF AWS, GIT, JENKINS, MAVEN, SONARQUBE, DOCKER AND NEXUS**

**ABSTRACT**

Few years back when agile methodology was playing a major role in the industry, software was deployed in monthly, quarterly or annual basis which was time consuming. But now it’s DevOps era! Where software can be deployed multiple times a day. In current era, delivering creative ideas in a rapid and steady manner is eminently significant for all organizations. In addition to that, organizations need to react to vigorous market requirements, faster time to market, decrease in failure rate and increase in customer interaction. This could be achieved with the help of DevOps methodology. DevOps methodology extends the agile to quickly produce software and automatically deploy them across various platforms/environment in order to gain high performance and quality assurance products. Continuous integration/Continuous deployment (CI/CD) is the backbone of DevOps environment. By automating the build, testing and deployment of software, CI/CD bridges the gap between development and operation teams. In this project, the source code(java application) will be deployed using AWS cloud service, Git, Maven, Sonarqube, Jenkins, Docker, Nexus in order to automate the entire environment.

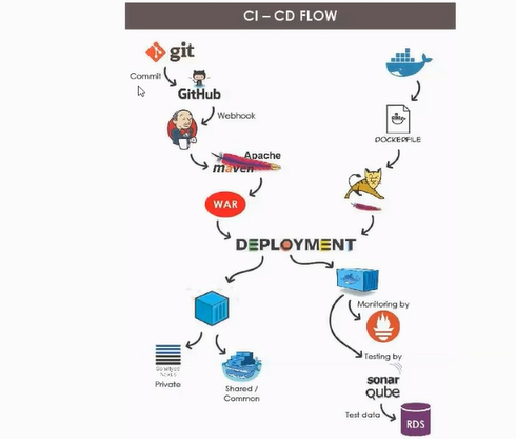
**INTRODUCTION**

Due to increasing competition in software industry, organizations play a major in assigning required resources to develop and deliver trustworthy and high quality products to consumers. Consumers expect to have continuous interaction with DevOps team so that they can provide their continuous feedback. DevOps is blending of two terms development and operations which aims to provide conjoin approach to industry‟s software development and operation team job in software development lifecycle. It provides a good communication between these two teams. DevOps describes the conformation of automation and programmable software development and infrastructure deployment and maintenance. Continuous integration, continuous deployment and continuous delivery are the important factors in software industry that helps organizations to constantly release new attributes and products that are trustworthy. Continuous integration focuses on integrating each developers work multiple times per day so that debugging of error is easy. Continuous delivery focuses on demoting discordance in deployment or release process and automating the build step so that code can be released securely at any time. CI/CD pipeline provides following benefits in software delivery lifecycle: obtaining rapid feedback from customers, rapid and steady release leads to have customer satisfaction and quality assured product, CD helps to automate tasks which was carried out manually.

**PROPOSED METHODOLOGY:**

In this project, we copy the source code from local machine to ec2 server using winscp. Using git, we commit and push the source code to the public repository called github. Jenkins automatically triggers the source code from git hub by integrating Jenkins and git hub. Maven triggers the source code from Jenkins and converts it as a war file by integrating maven and jenkins. Sonarqube automatically triggers the war file of source code for quality testing by integrating jenkins and Sonarqube. Tomcat install in docker to deploy the war file on top of it. war file converted as docker image and push to the docker hub and also private repository called nexus.

**PROJECT ARCHITECTURE:**



**TOOLS USED IN THIS PROJECT:**

**Git:**

Git is a version control tool used to push the code into remote repository i.e., Github.com during software development lifecycle. It is also used to monitor changes in file sets. Developers push their code to repository created in Github.com using git commands. Initially install git into the server using sudo yum install git -y command.

**Maven:**

Maven is project management and comprehension tool which provides complete build lifecycle framework for developers. Maven is based on Project Object Tool (POM) file. POM is used for project builds, dependency and documentation. POM is a XML file that is present in the base directory of project as pom.xml. POM file contains all the necessary information and configuration details of the project.

**SonarQube:**

SonarQube is an open source platform developed by sonar source for continuous inspection of code quality to perform automatic reviews with static analysis of code to detect bugs. SonarQube offers reports on duplicated code, coding standards, unit tests, code coverage, code complexity, comments, bugs and security recommendations. SonarQube provides fully automated analysis and can be integrated with building tool like maven and continuous integration tool like Jenkins.

**Jenkins:**

Continuous integration (CI) process is carried out using Jenkins tool. Jenkins is an open source automation server helps to automate manual work of software development related to building, testing, and deploying, facilitating continuous integration and continuous delivery. It is a server-based system that runs in serverlet containers such as Apache Tomcat.

**Docker:**

Docker is a containerization platform that is used to create a package containing an application and all its dependencies altogether in the form of a docker container to make sure that the application works perfectly in all environments. Docker container is a standardized unit which is created on the fly to deploy a specific application or environment. Consider a scenario where code running in one machine is not running in another machine. This is due to environmental change. To overcome this problem, Docker is used. Docker image is created.

**Nexus:**

Nexus Repository is an open source repository that supports many artifact formats, including Docker, Java™, and npm. With the Nexus tool integration, pipelines in your toolchain can publish and retrieve versioned apps and their dependencies by using central repositories that are accessible from other environments.

**AWS EC2 Sever:**

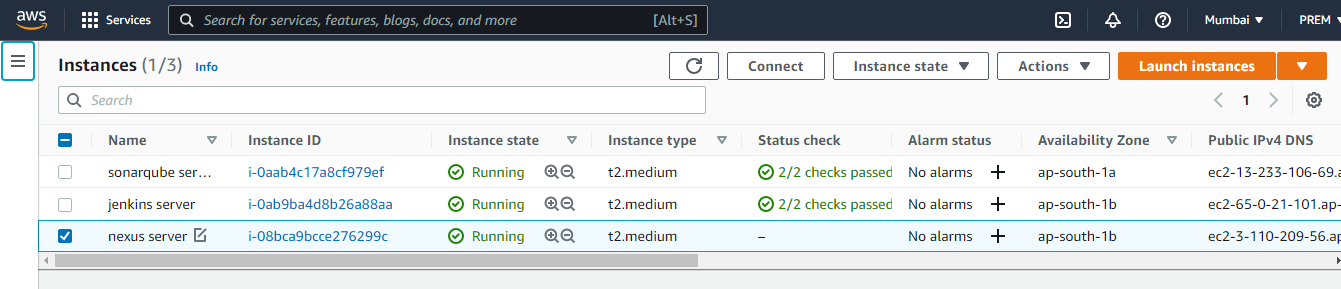
Elastic Compute Cloud is a virtual machine that represents a physical server for you to deploy your application. Ec2 allows users to build apps to automate scaling according to changing needs and peak periods, and makes it simple to deploy virtual servers and manage storage, lessening the need to invest in hardware and helping streamline development processes.

**RDS:**

Amazon Relational Database Service (RDS) is a collection of managed services that makes it simple to set up, operate, and scale databases in the cloud. Amazon RDS supports an array of database engines to store and organize data. It also helps with relational database management tasks, such as data migration, backup, recovery and patching.

**WORKING PROCEDURE:**

Create and launch three ec2 servers on any region (Mumbai) and go as root user. Install and start Git, Jenkins, maven, Docker on to the Jenkins server. Install and start SonarQube on Sonarqube server and install and start nexus on nexus server.

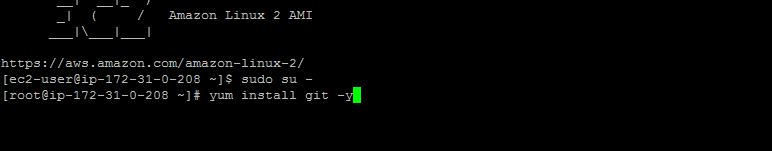


**GIT INSTALLATION:**

# Switch to root user

# Git installs on the jenkins server by following command.

* yum install git –y



**INSTALLATION OF JENKINS:**

**Prerequisites of Installation Jenkins:**

# 256 MB of RAM is required.

# 1 GB of drive space is required(although 10 GB is a recommended minimum if running Jenkins as a Docker container**.**

# JDK should install before installing the jenkins.

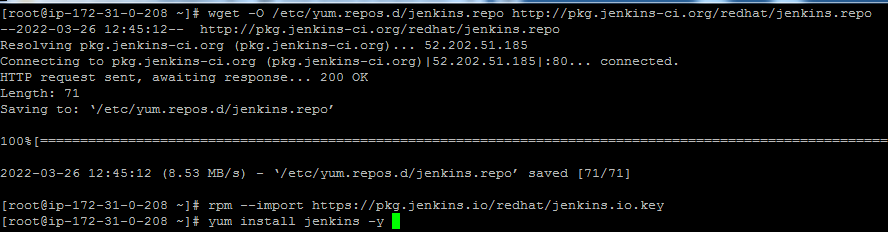
# Jenkins installs on the jenkins ec2 server by following commands.

# Switch to root user.

* amazon-linux-extras install epel –y
* yum update –y
* wget -O /etc/yum.repos.d/jenkins.repo \https://pkg.jenkins.io/redhat-stable/Jenkins.

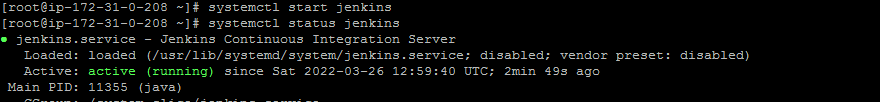
repo

* rpm --import <https://pkg.jenkins.io/redhat-stable/jenkins.io.key>
* amazon-linux-extras install java-openjdk11
* yum install jenkins -y

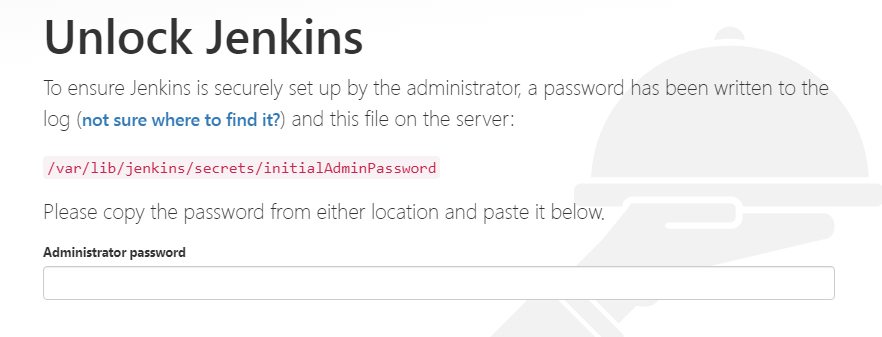


# Jenkins Start and know the status of jenkins by following commands.

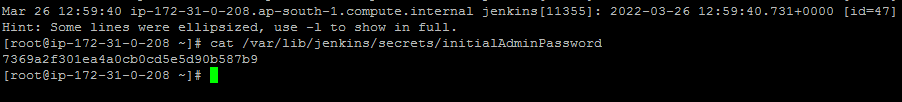
* systemctl start jenkins
* systemctl status jenkins

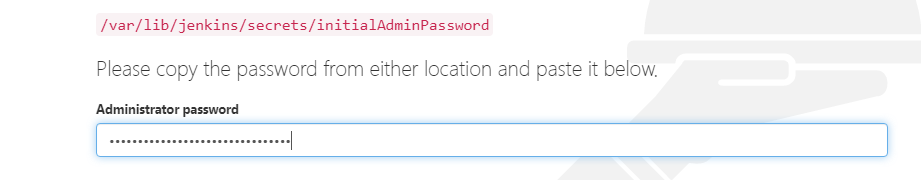


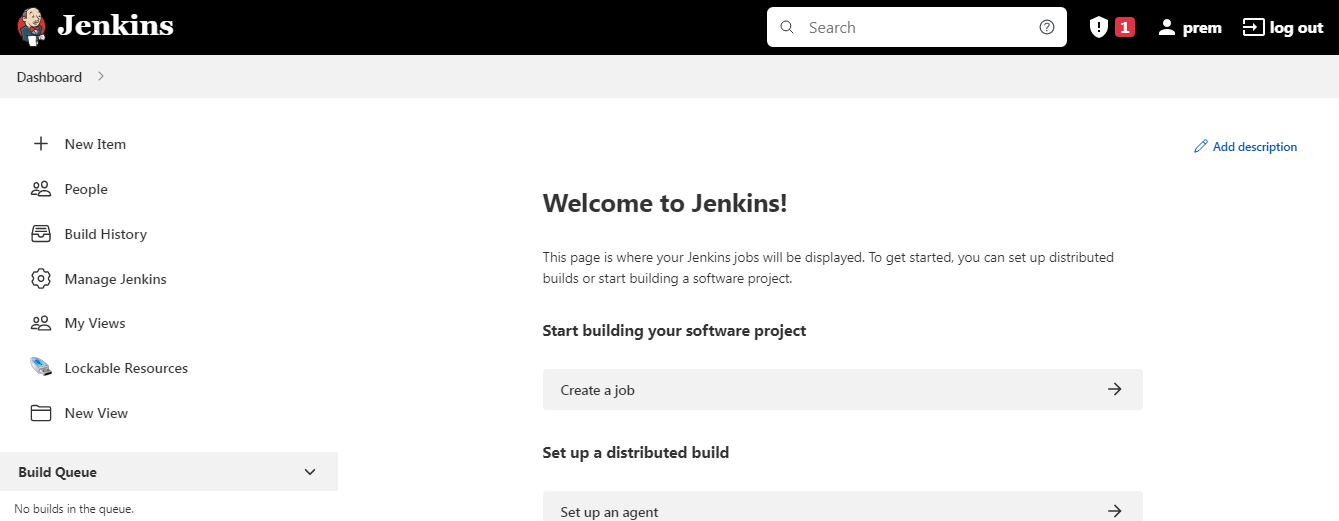
# Hit the browser by public ip address of jenkins server with 8080(jenkins port number) to open jenkins console.



# Get the Administrator password from jenkins servers /var/lib/jenkins/secrets/initialAdminPassword path and paste it onto jenkins consoles Administrator password place to unlock the jenkins console.

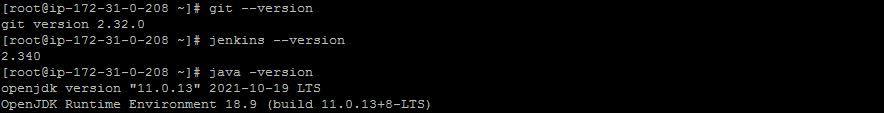






# Check the version of git, jenkins and java by the following commands.

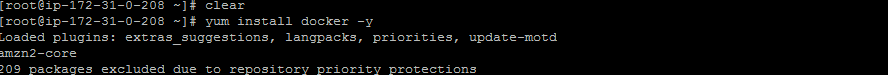
* Git –version
* Jenkins –version
* Java --version



**DOCKER INSTALLATION:**

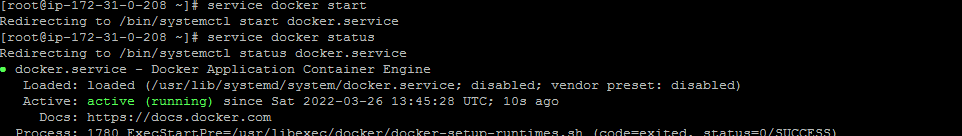
# Docker installs on server by following command.

* yum install docker –y



# docker start and know the status of docker by the following command.

* service docker start.
* Service docker status



**MAVEN INSTALLATION:**

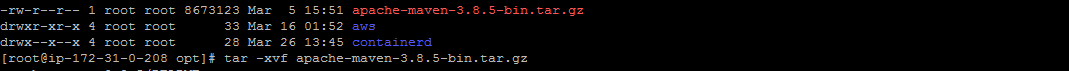
# Maven installs on server’s opt directory.

# cd /opt/

# Maven installs on the server by following command.

* wget https://mirrors.estointernet.in/apache/maven/maven3/3.8.5/binaries/apache-maven-3.8.5-bin.tar.gz
* tar –xvf apache-maven-3.8.5



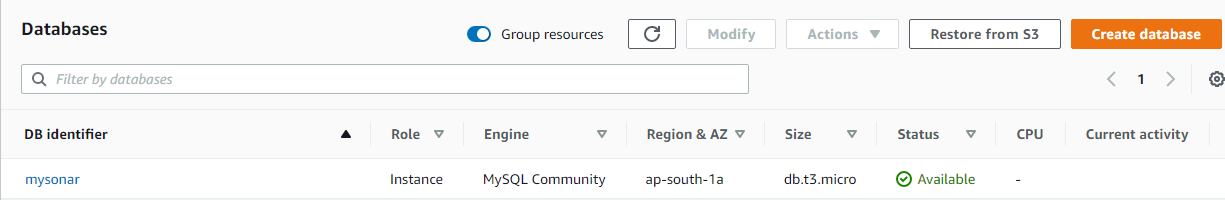


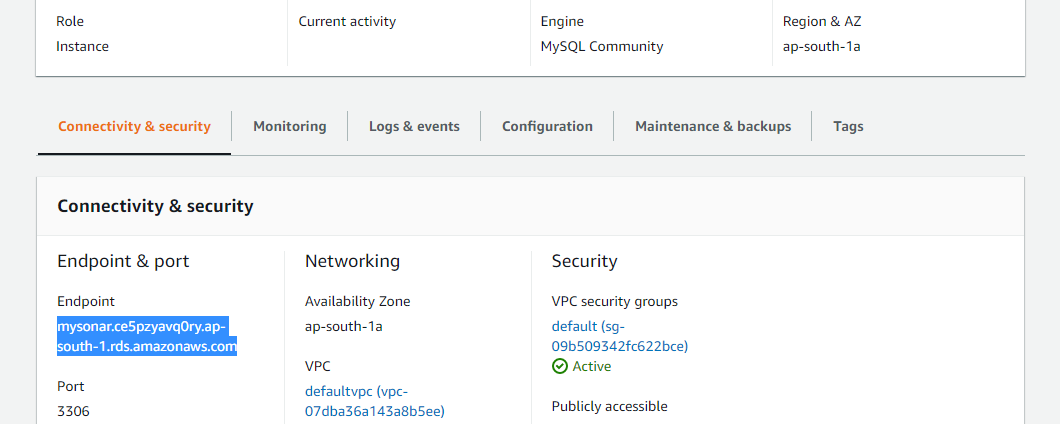
**INSTALLATION OF SONARQUBE:**

**Prerequisite of Sonarqube:**

* 3gb ram machine is required.
* Java open-jdk is necessary.
* RDS Data base server (mysql) is required
* Creation of local user and remote user and permission access in data base server.
* Sonarqube should not start with root user

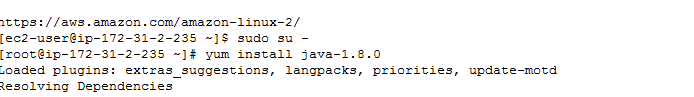
# Creation of RDS database server





# Java installation on the sonarqube server using the following command.

* yum install java-1.8.0

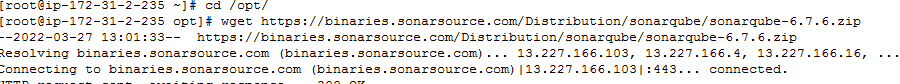


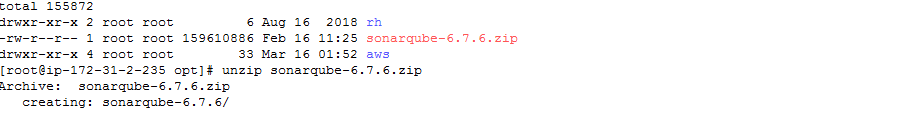
# Installation of sonarqube by the following command.

# SonarQube installs on server’s opt directory.

# cd /opt

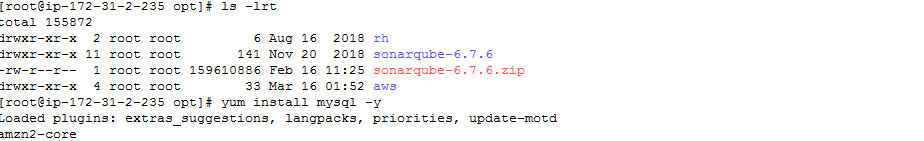
* wget <https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-6.7.6.zip>
* unzip sonarqube-6.7.6.zip





# Install mysql on Sonarqube server by following command.

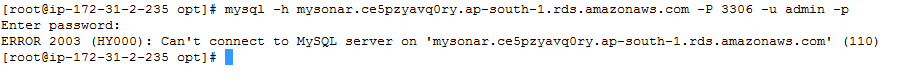
* yum install mysql -y



# Connect RDS endpoint to the mysql database by following command.

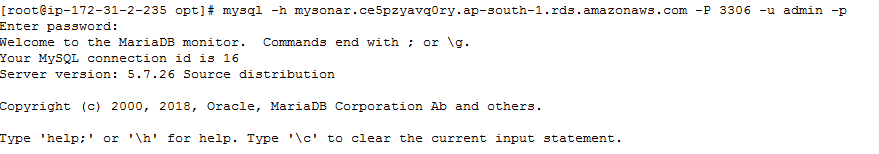
* Mysql –h endpoint mysonar.ce5pzyavq0ry.ap-south-1.rds.amazonaws.com –P 3306 –u admin –p

**ERROR 1:**



**TROBLESHOOT:**

1. Go to EC2 Dashboard
2. Go to Security Groups tab
3. Select and only select the RDS database security group. You'll see the security group detail at the bottom
4. Click Inbound tab
5. Click Edit button
6. Add Type:MYSQL/Aurora;Protocol:TCP;Range:3306;Source:0.0.0.0/0



**#** Create local and remote user and grant mysql database access to the user..

# cd /opt/

# Create Database by following command.

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

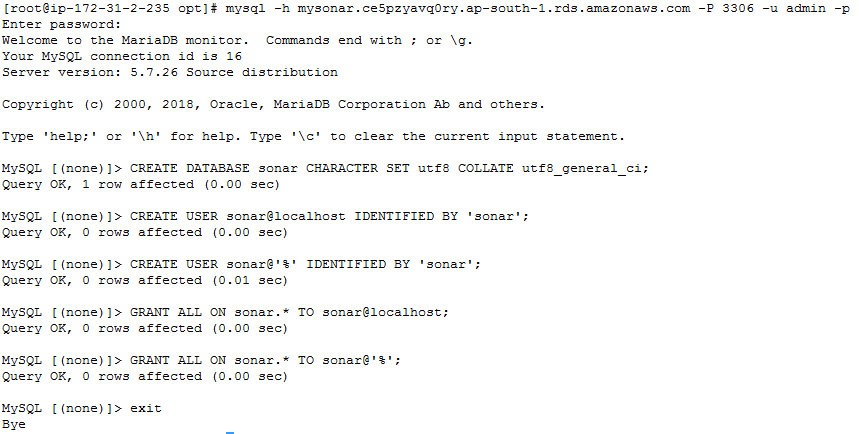
# Create a local and a remote user by following command.

* CREATE USER sonar@localhost IDENTIFIED BY 'sonar';
* CREATE USER sonar@'%' IDENTIFIED BY 'sonar';

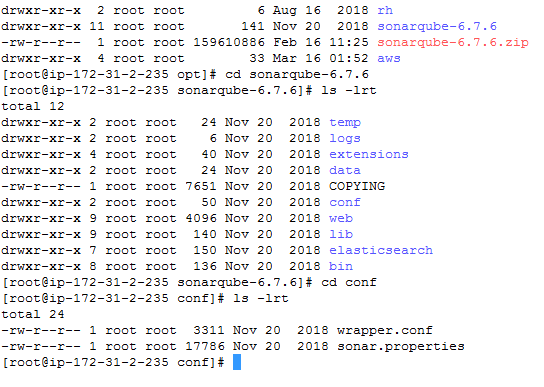
# Grant database access permissions to users by following command

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

# Exit.



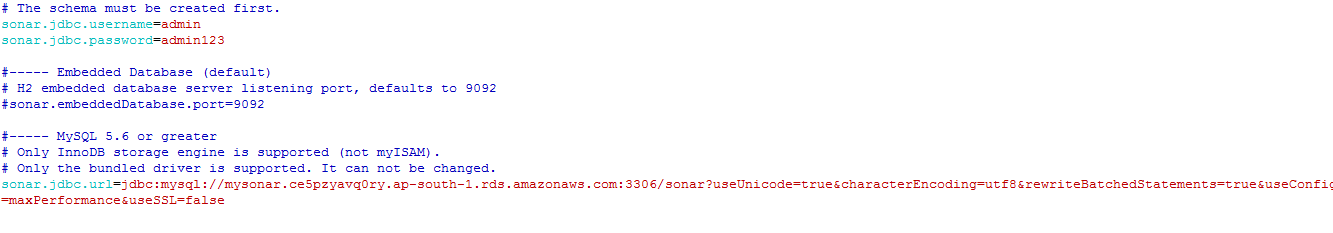
# cd /opt/Sonarqube-6..7.6/conf

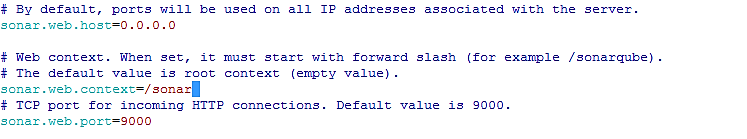
****

# 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
* sonar.jdbc.url=jdbc:mysql mysonar.ce5pzyavq0ry.ap-south-1.rds.amazonaws.com :3306/sonar?useUnicode=true&characterEncoding=utf8&rewriteBatchedStatements=true&useConfigs=maxPerformance&useSSL=false
* sonar.web.host=0.0.0.0
* sonar.web.context=/sonar

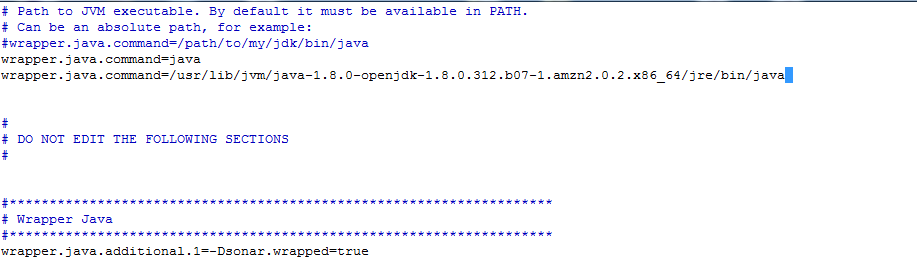
****

****

# Do changes in wrapper.conf file

# Filename: /opt/sonar/conf/wrapper.conf

Wrapper.java.command=/usr/lib/java-1.8.0-openjdk-1.8.0.312.b07-1.amzn2.0.2.x86\_64/jre/bin/java

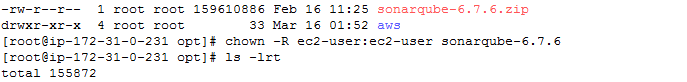
****

# Sonarqube should not start with root user.

# cd /opt/

# following command is used to convert the sonarqube-6.7.6 directory as a ec2-user from root user.

* chown –R ec2-user:ec2-user sonarqube-6.7.6

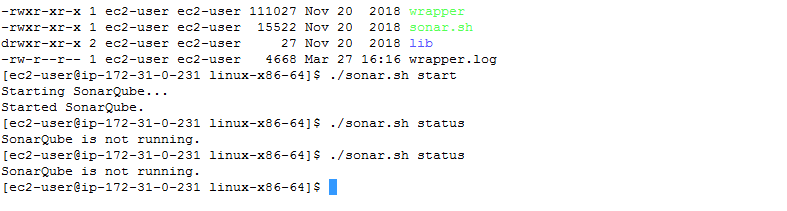
****

# cd /sonarqube-6.7.6/bin/linux-x86-64

# SonarQube Start and know the status of the Sonarqube by following command.

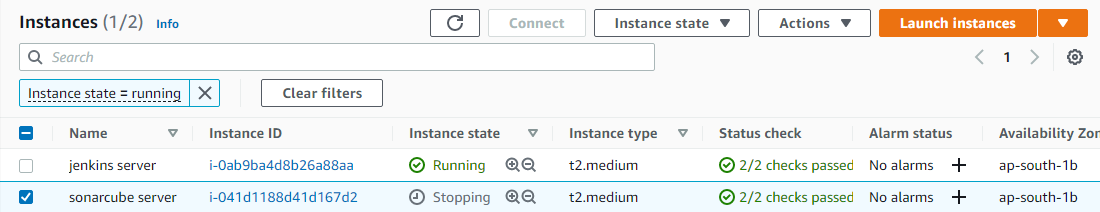
* ./sonar.sh start
* ./sonar.sh.status

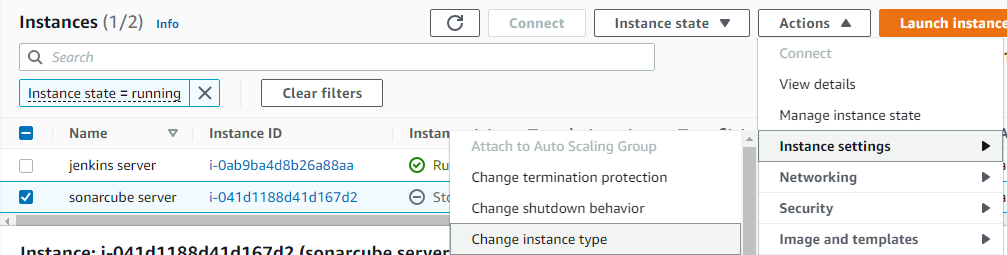
**ERROR 2:**

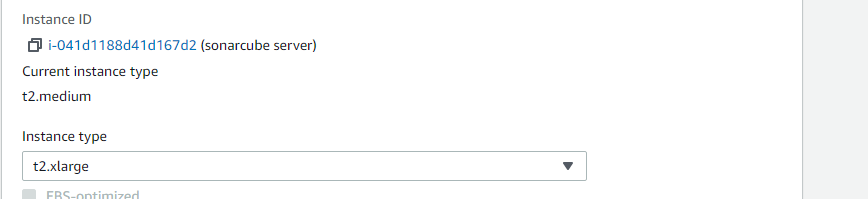
****

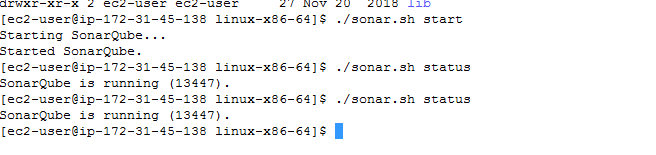
**TROUBLESHOOT:**

# Change the instance type from t2.mediam to t2.large and restart the Sonarqube.

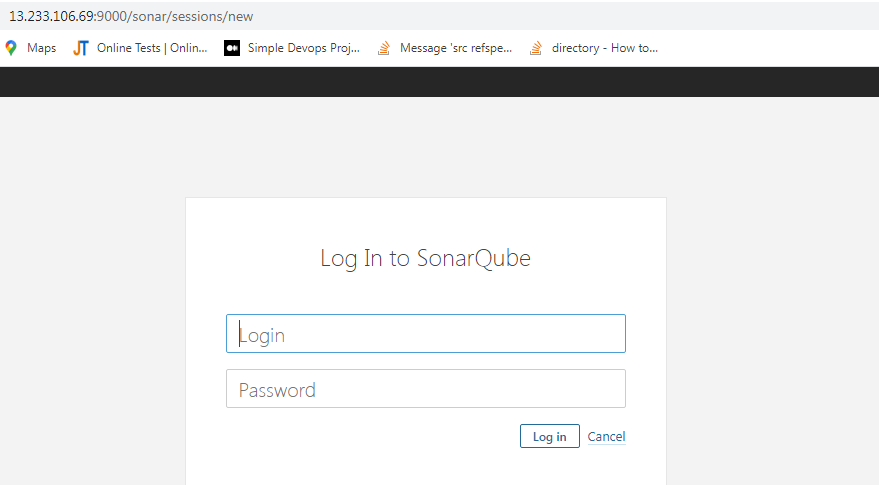
****

****

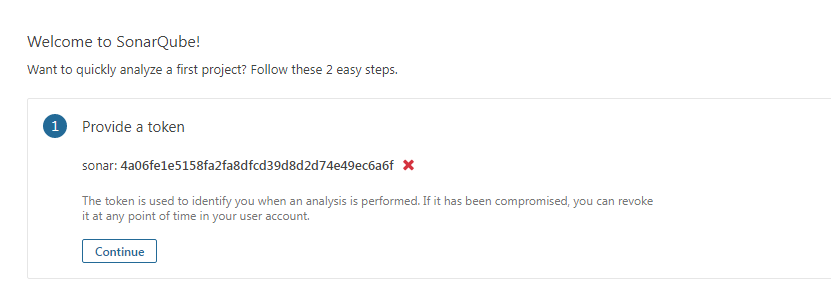
****

****

# Access the sonarqube console with 9000 port and create a token

****

# This token is to communicate jenkins and Sonarqube.

****

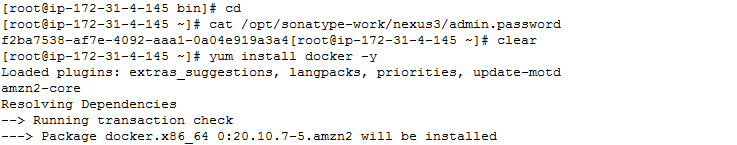
**INSTALLATION OF NEXUS:**

**Prerequisite of Nexus:**

* Minimum 3bb ram machine is required.
* Java open-jdk is also required

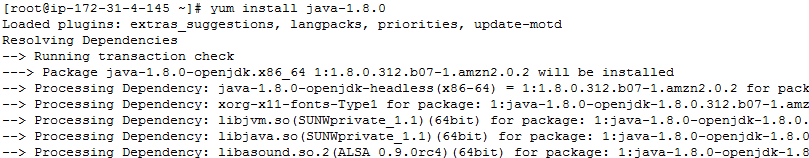
# Docker installs on the nexus server by following command.

* yum install docker -y



# Java installs on the nexus server by the following command.

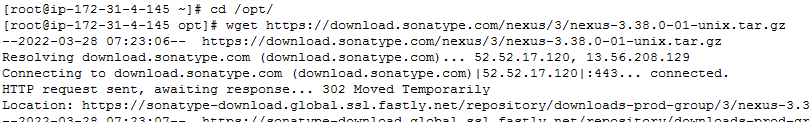
* yum install java-1.8.0



# Install nexus on opt diecrtory

# cd /opt

* wget <https://download.sonatype.com/nexus/3/nexus-3.38.0-01-unix.tar.gz>



# untar the nexus-3.38.0-01-unix.tar.gz using following command

* tar –xvf nexus-3.38.0-01-unix.tar.gz

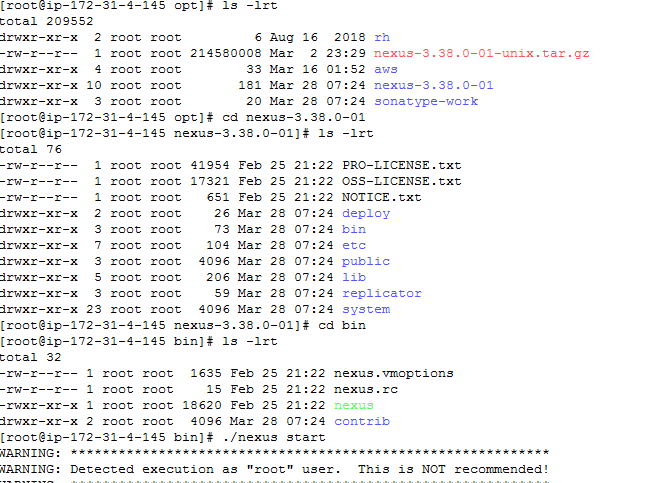
# Start the nexus from bin using the following command.

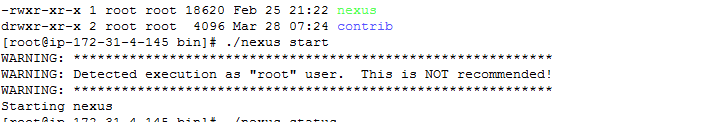
# cd /opt/ nexus-3.38.0-01/bin

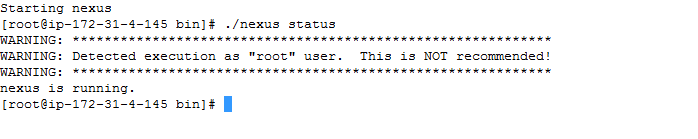
* **./**nexus start

# Know the status of nexus using following command.

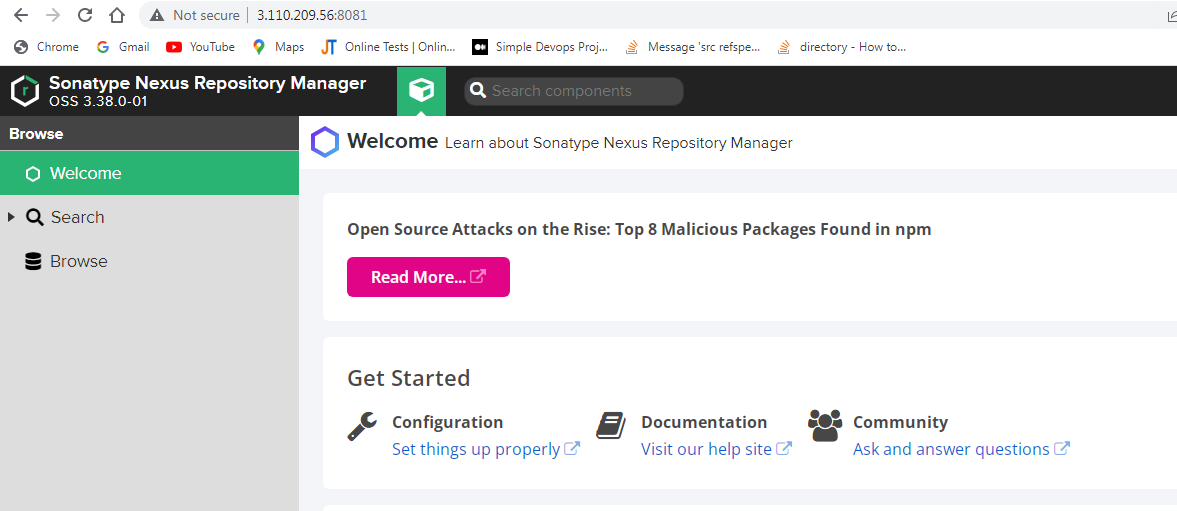
* **./** nexus status

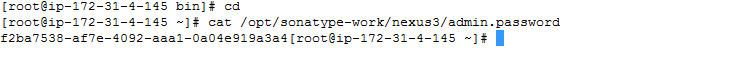




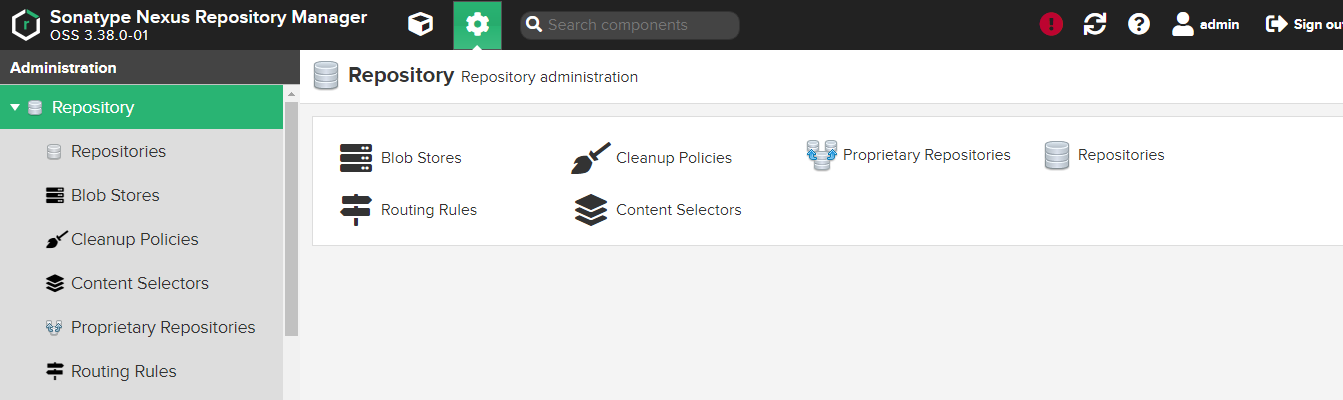


# Access the nexus console with port 8081 and get the password from cat /opt/sonatype-work/nexus3/admin.password/

****

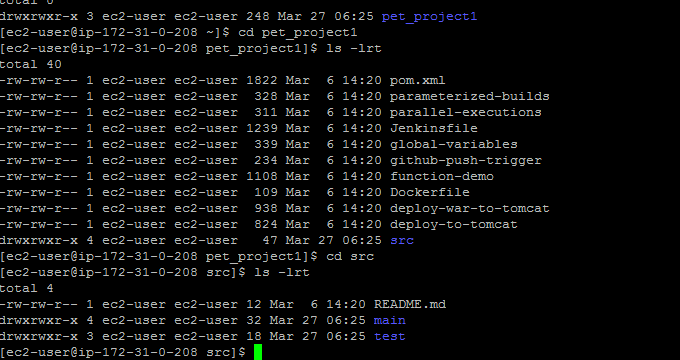
****

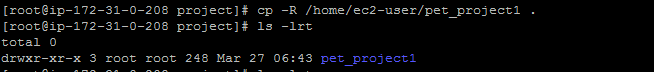
# Copy above password and paste it to nexus console to open the sonatype nexus repository manager.



**PUSH THE SOURCE CODE TO GITHUB:**

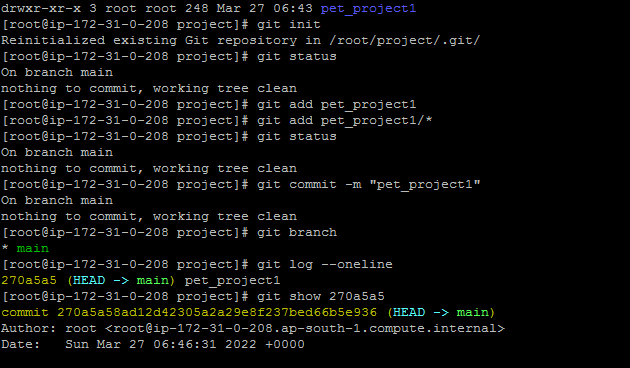
**#** Copy the source code from local system to ec2 server using winscp.

****

****

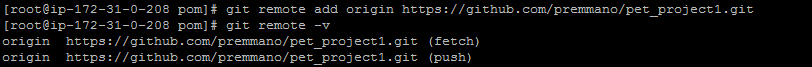
# Commit the source code from working directory to local server by the following commands.

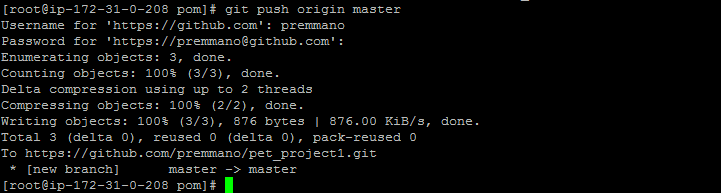
* git init
* git add pet\_project1/\*
* git commit –m “pet\_project”

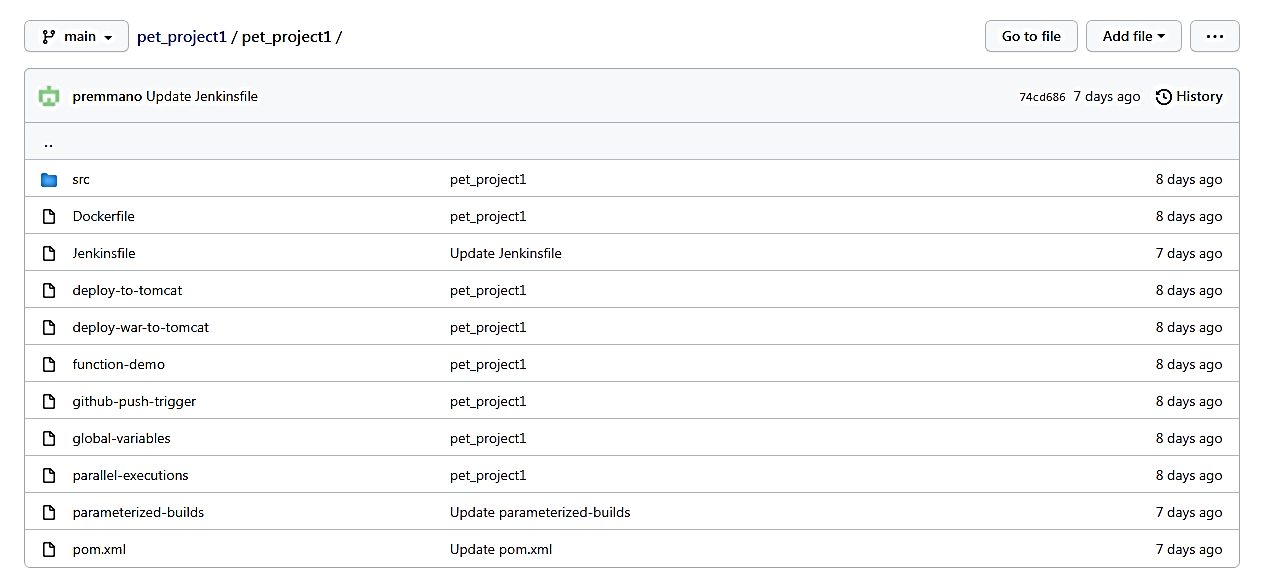
****

# Committed source code push to remote repository from local repository by the following commands.

* git remote add origin <https://github.com/premmano/pet_project1.git>
* git push origin master

****

****

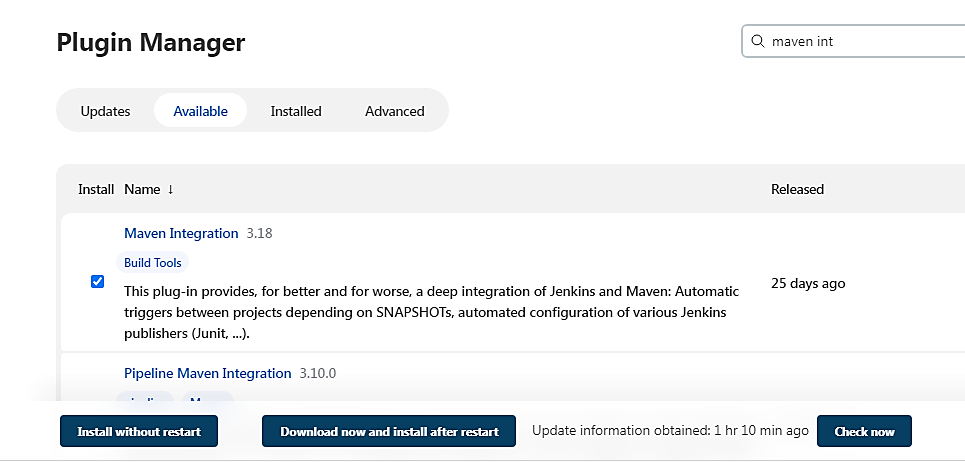
****

**INTEGRATING AND CONFIGURING TOOLS WITH JENKINS:**

**INTEGRATION OF JENKINS AND MAVEN:**

# Integrate the jenkins and maven using **mavan integration** plugin.

# Go to Manage jenkins – manage plugin – available – maven integration – install without restart.

****

# Put the jenkins file script to pipeline.

# Go to New – projectCICD – pipeline – pipeline script

node{

stage('SCM Checkout'){

git 'https://github.com/damodaranj/my-app.git'

}

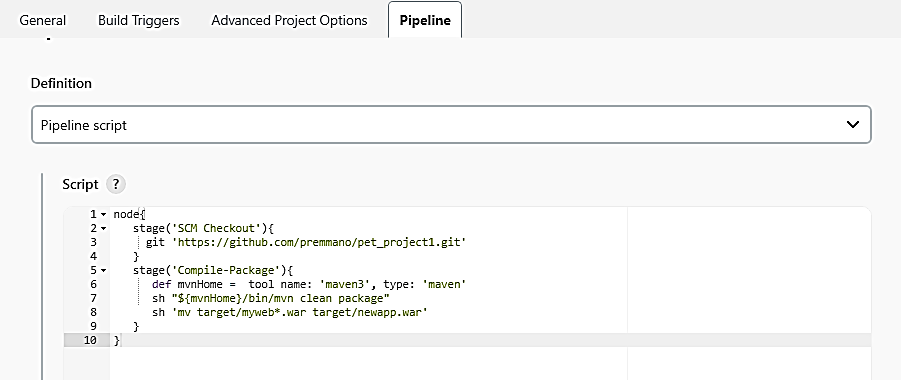
stage('Compile-Package'){

def mvnHome = tool name: 'maven3', type: 'maven'

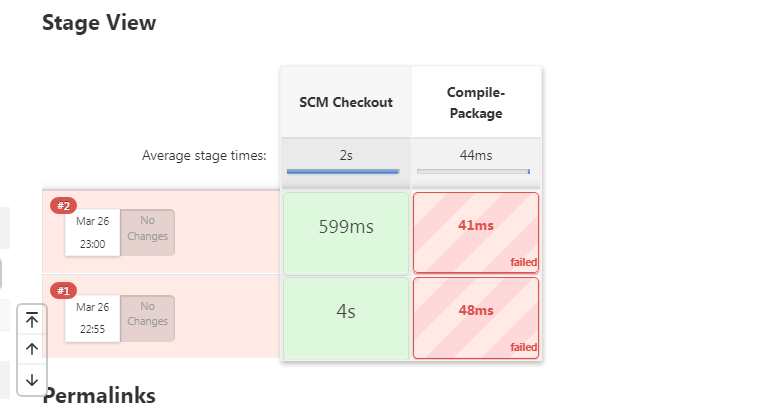
sh "${mvnHome}/bin/mvn clean package"

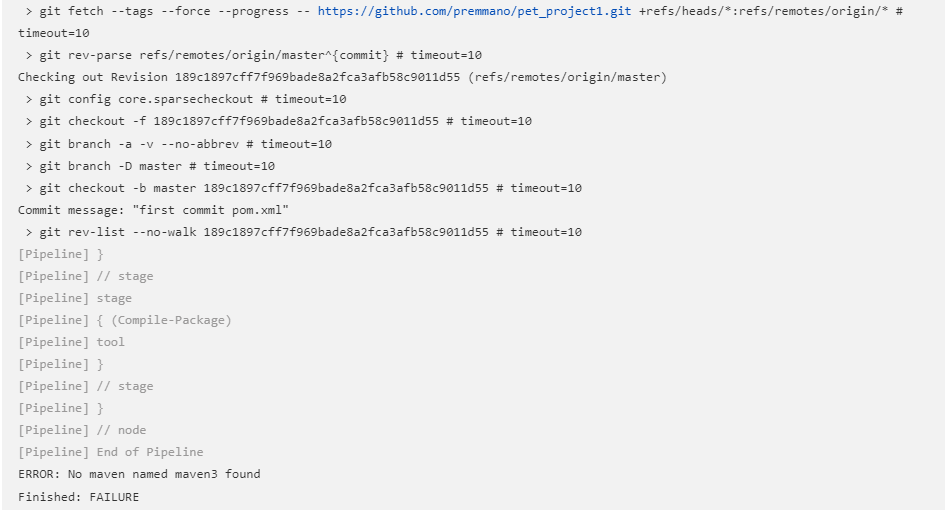
sh 'mv target/myweb\*.war target/newapp.war'

}

****

**ERROR 3:**

****

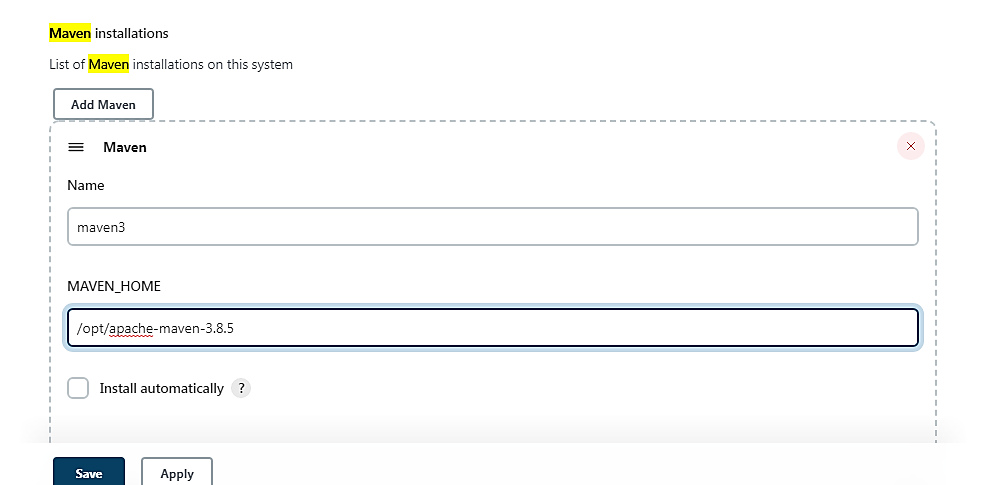
****

**TROUBLESHOOT:**

# It is required to specify the maven path on the jenkins console to fix the error.

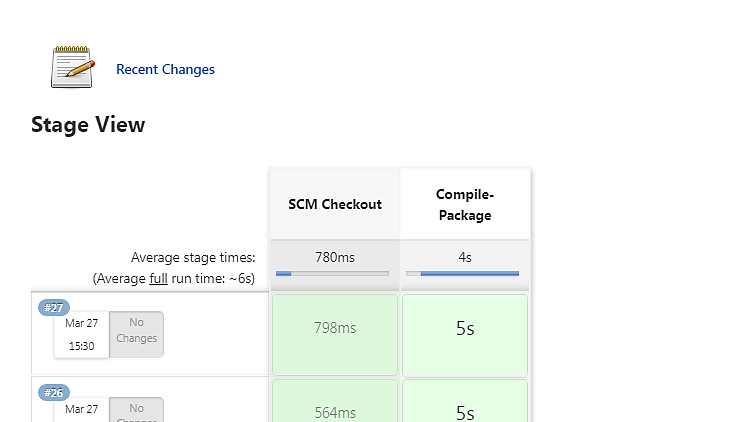
# Go to jenkins console - Manage jenkins – global tool configuration - maven

**Maven path specify:**

****

**Run the script:**

# Go to New – projectCICD – pipeline – pipeline script – add the script - save

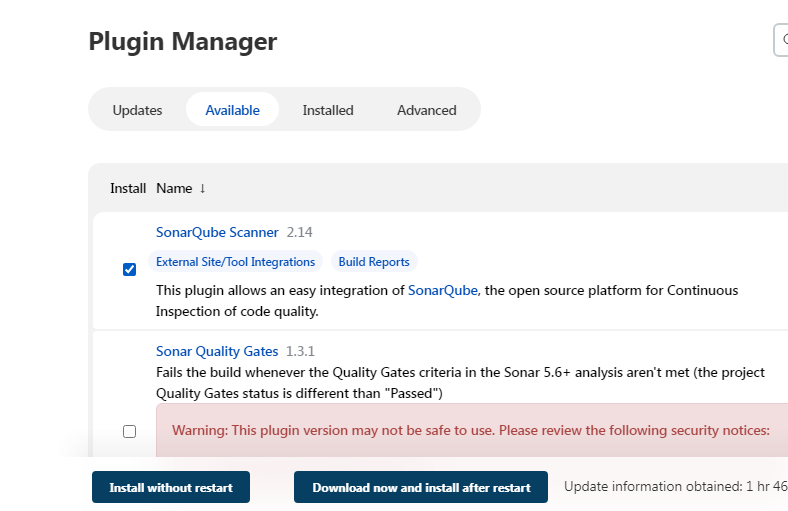
****

**INTEGRATION OF JENKINS AND SONARQUBE**:

# Open jenkins console.

# Integrate jenkins and sonarqube by **SonarQube Scanner plugin.**

# Go to Jenkins console - Manage jenkins – manage plugins – available – SonarQube sscanner – install without restart.

****

# SonarQube credentials should add on jenkins server.

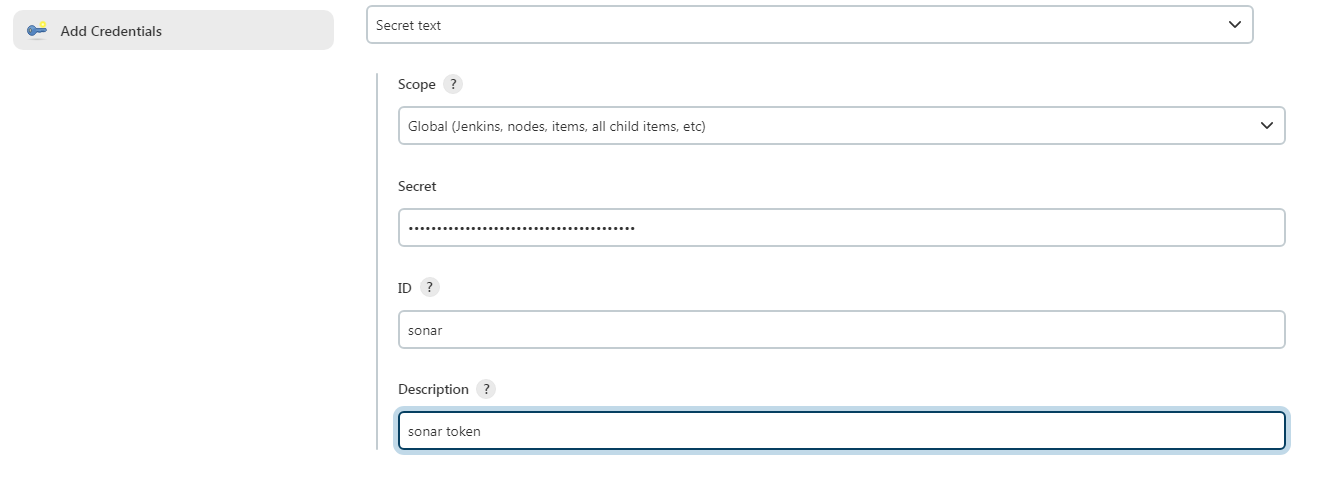
# Go to jenkins console - Manage jenkins – manage credentials – jenkins – Global credentials – add credentials

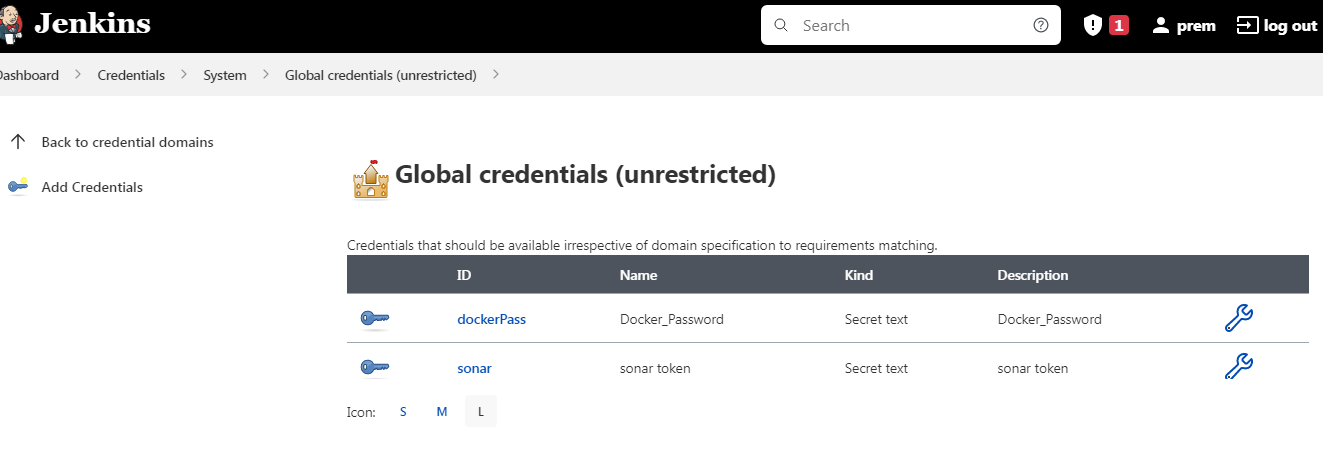
# Scope – Global

# Secret – Paste sonarqube token

# ID – sonar

# Description – sonar token



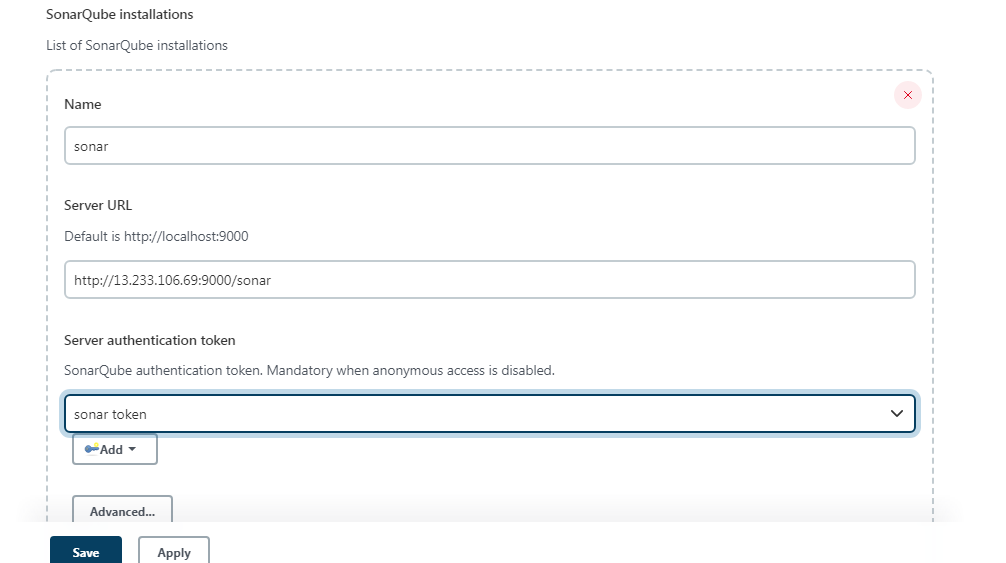
****

# Go to Jenkins console - Manage jenkins – configure system – Sonarqube servers – sonar installations.

# Name – sonar

# Server url – <http://sonarqube> server ip:9000/sonar (<http://13.233.106.69:9000/sonar>)

# Server authentication token – sonar token

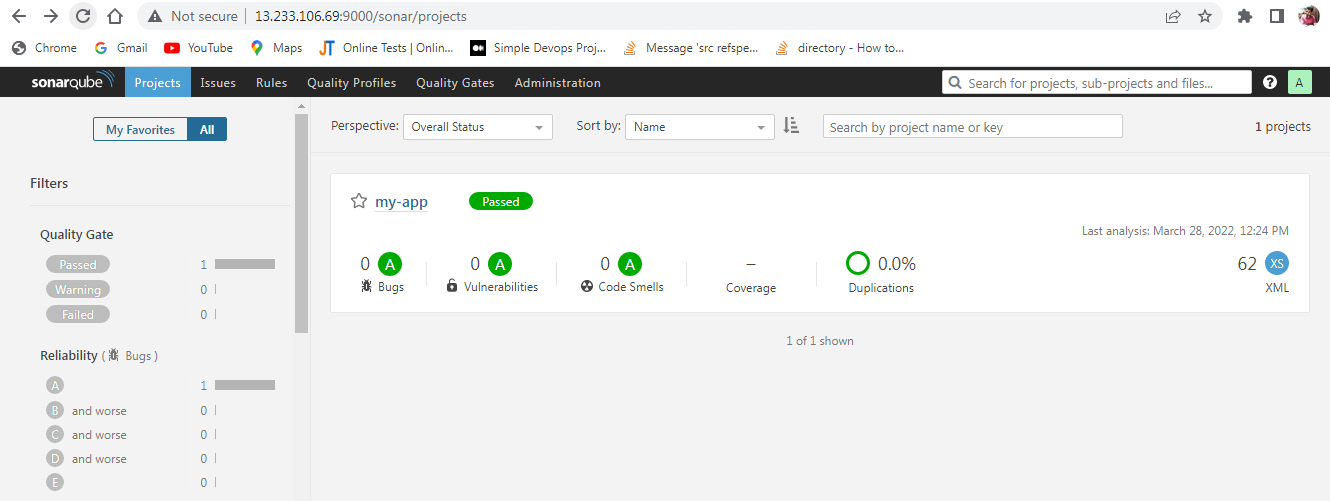
****

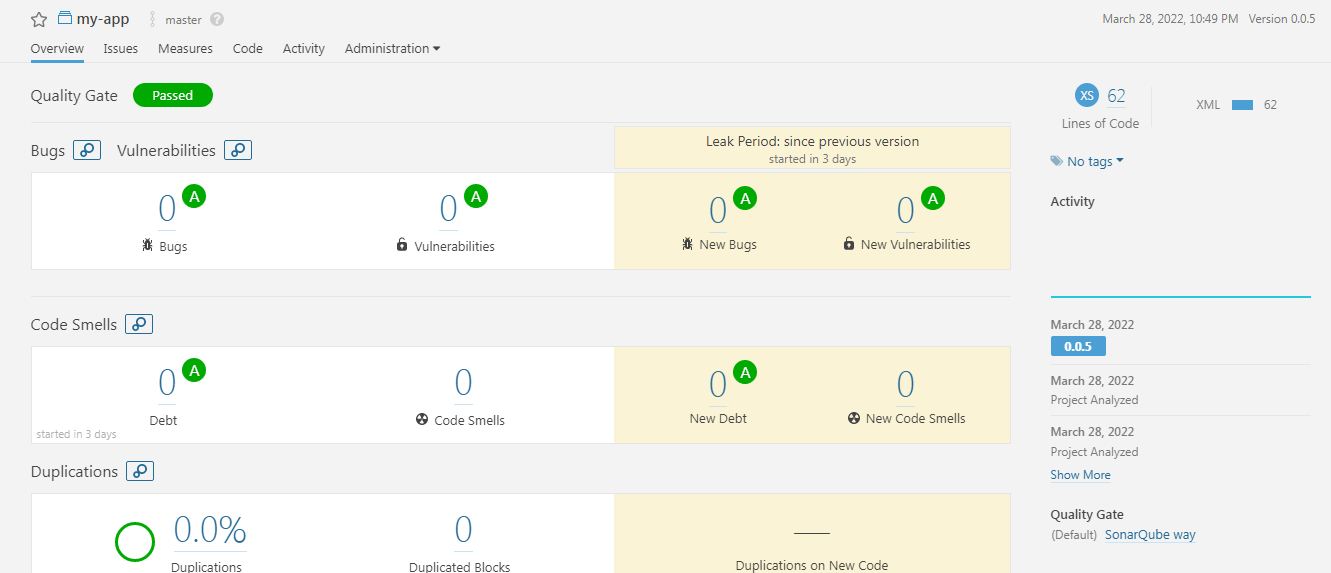
# run the script

# Go to New – projectCICD – pipeline – pipeline script – add the script – save.



# Go to sonarqube console with port no 9000.

****

****

**DOCKER BUIlD:**

**Docker File:**

FROM tomcat:8

# Take the war and copy to webapps of tomcat

COPY target/newapp.war /usr/local/tomcat/webapps/

# Run the script.

# Go to New – projectCICD – pipeline – pipeline script – add the script - save

stage('Build Docker Imager'){

sh 'docker build -t premmano/myweb:0.0.2 .'

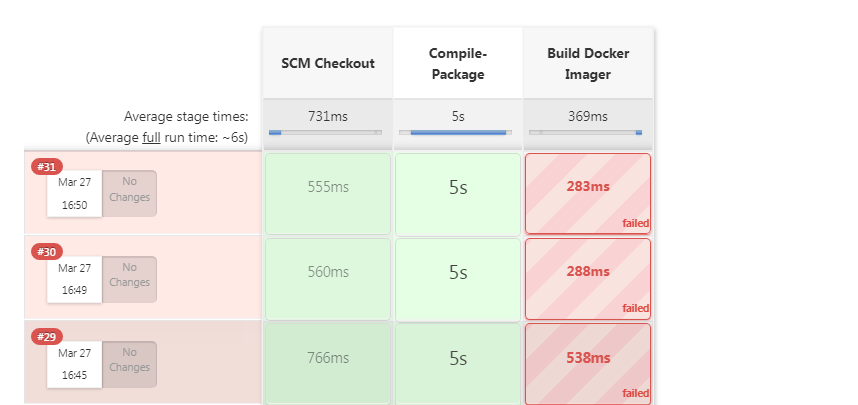
}

# This script converts the docker file into a docker image called as premmano/myweb.

****

**ERROR 4:**

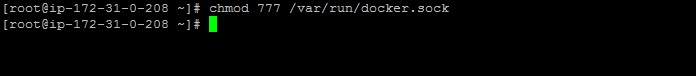
**#** Can not connect to the docker daemon at unix:///var/run/docker.sock.

****

****

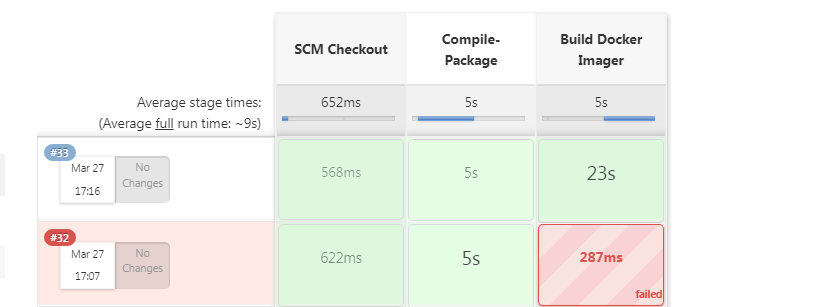
**TROUBLE SHOOT:**

Give full permission to fix the error.

****

# Run the jenkins file script.

# Go to New – project\_CICD – pipeline – pipeline script – add the script - save

****

**DOCKER PUSH:**

# This script is to push the docker image to dockerhub. The Docker hubs user name and password are essential to push the image to the docker hub.

stage('Docker Image Push'){

withCredentials([string(credentialsId: 'dockerPass', variable: 'dockerPassword')]) {

sh "docker login -u premmano -p ${dockerPassword}"

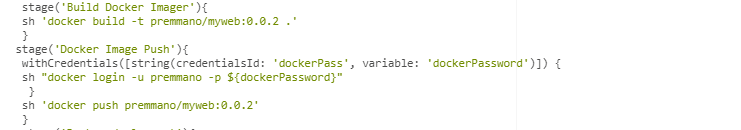
}

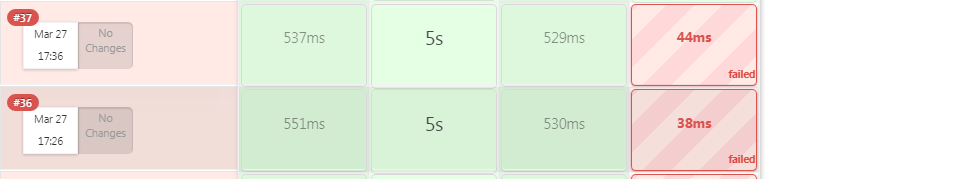
sh 'docker push premmano/myweb:0.0.2'

}

# Run the script.

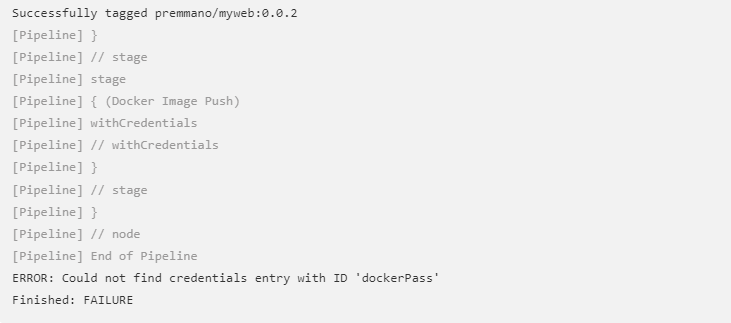
# Go to new – projectCICD – pipeline – pipeline script – add the script - save

****

****

**ERROR 5:**

# Could not find any credentials entry with ID “dockerPass” on jenkins server.

****

**TROUBLE SHOOT:**

# Set the docker credential id on jenkins server to integrate jenkins and docker.

# Go to configuration – manage jenkins – manage credentials – jenkins – global credentials – add credentials.

# kind - secret text

# Scope – Global

# Secret – Dockerhub password

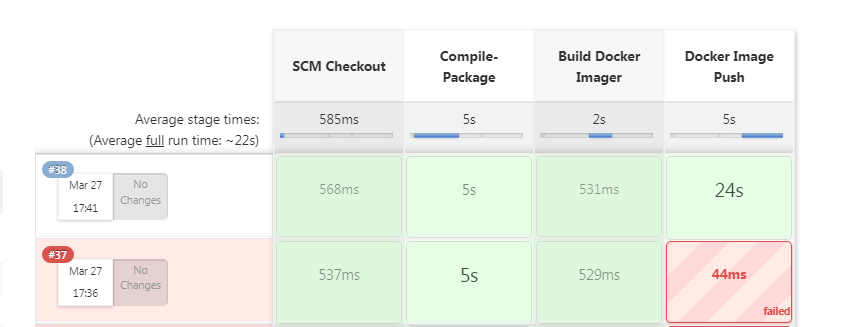
# ID – dockerpass

# Description – dockerhub\_password

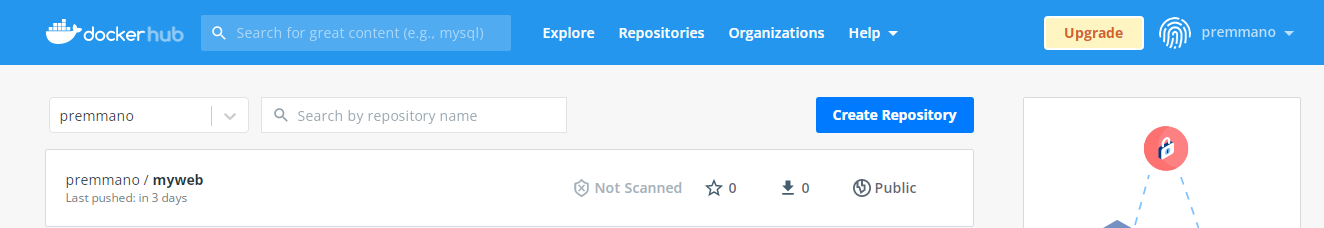
****

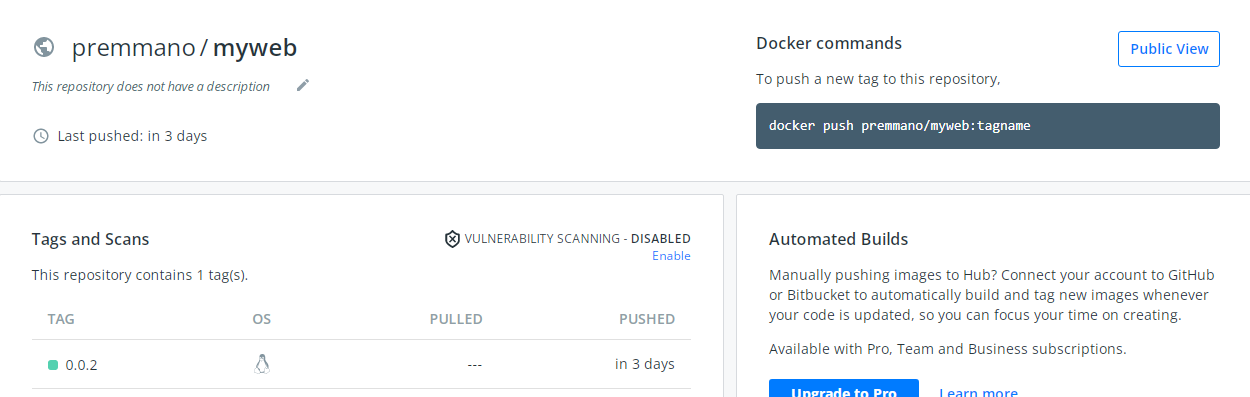
# Run the script.

# Go to New – projectCICD – pipeline – pipeline script – add the script - save

****

# Go to docker hub and check the image.

****

****

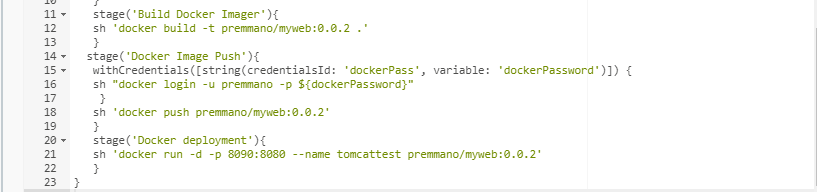
**DOCKER DEPOYMENT:**

stage('Docker deployment'){

sh 'docker run -d -p 8090:8080 --name tomcattest premmano/myweb:0.0.2'

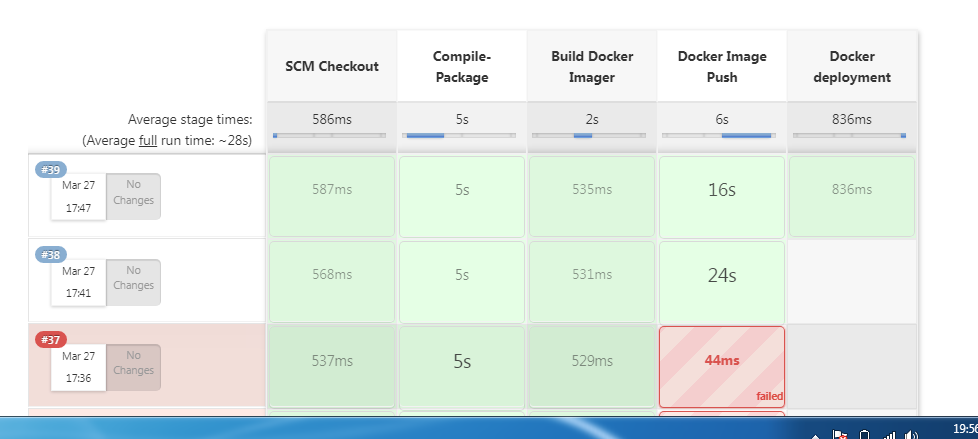
}

# This script is to launch the container called tamcattest.

****

# Run the script.

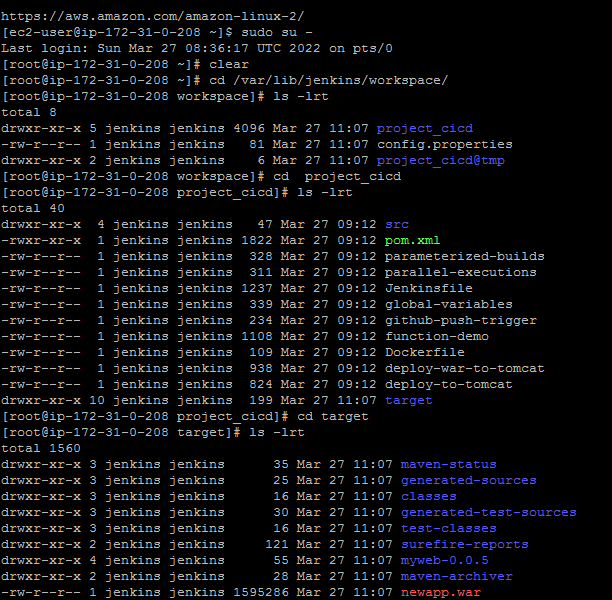
# New – projectCICD – pipeline – pipeline script – add the script - save

****

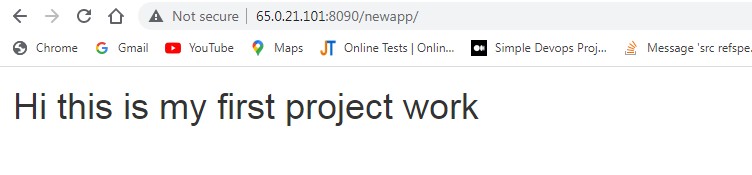
****

# Go to filename name: newapp.war

# cd /var/lib/jenkins/workspace/cicd\_project/target/newapp.war

****

# Check: jenkins server ip:8090/newapp/

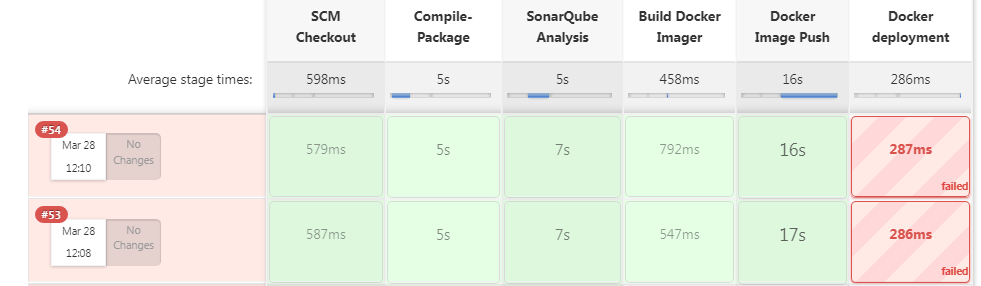
****

# Go to Github – Src – main – webapp - index.html

# In the index.html file mention “Hi this is my first project work”.

****

# run the script again.

****

**ERROR 6:**

# The container name “tomcattest” is already used by a existing container. So that container must be removed (or renamed) in order to reuse that name.

****

**TROUBLESHOOT:**

**#** The following script is used to remove ( or rename) that container for reuse that name.

stage('Remove Previous Container'){

try{

sh 'docker rm -f tomcattest'

}catch(error){

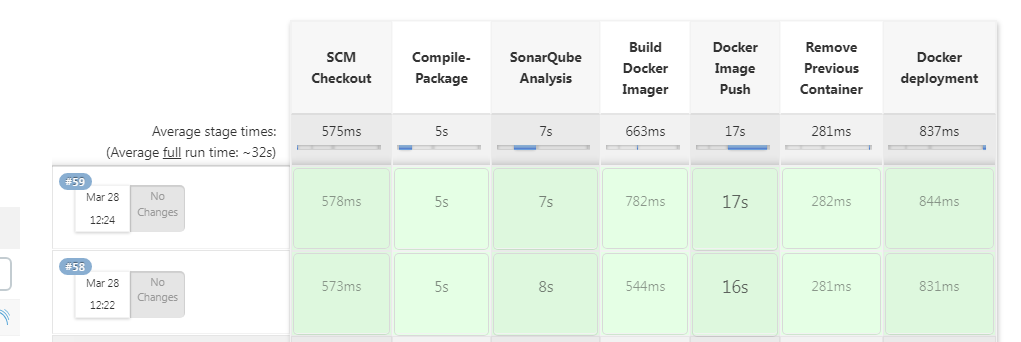
// do nothing if there is an exception

}

****

# Run the pipeline

# New – projectCICD – pipeline – pipeline script – add the script - save

****

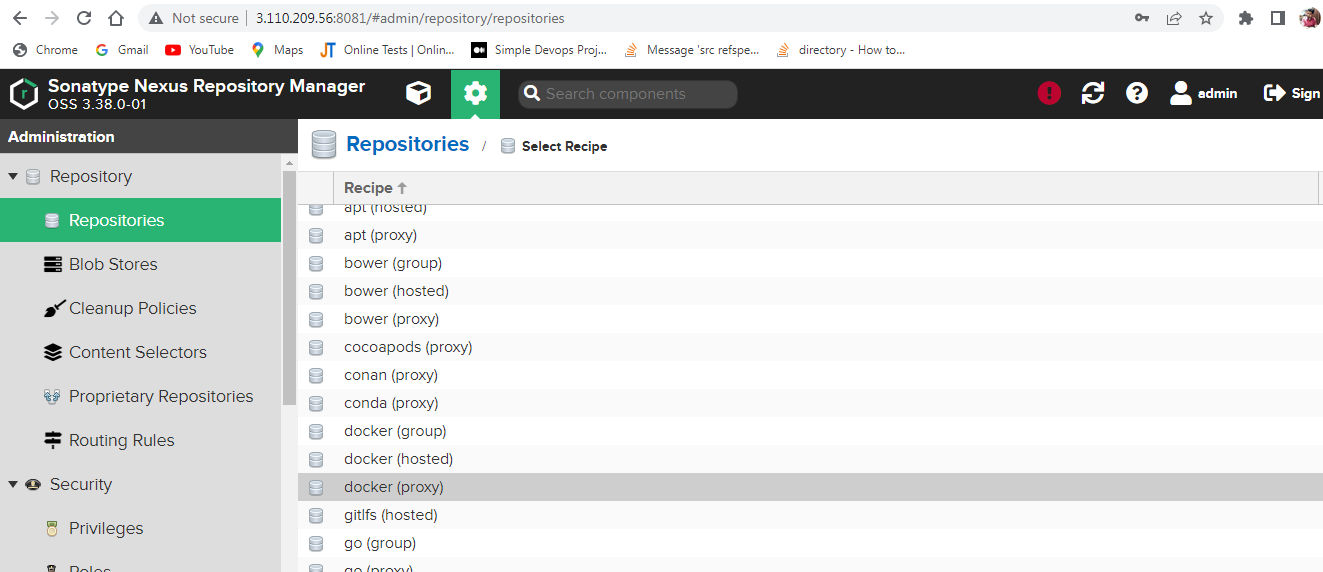
**NEXUS:**

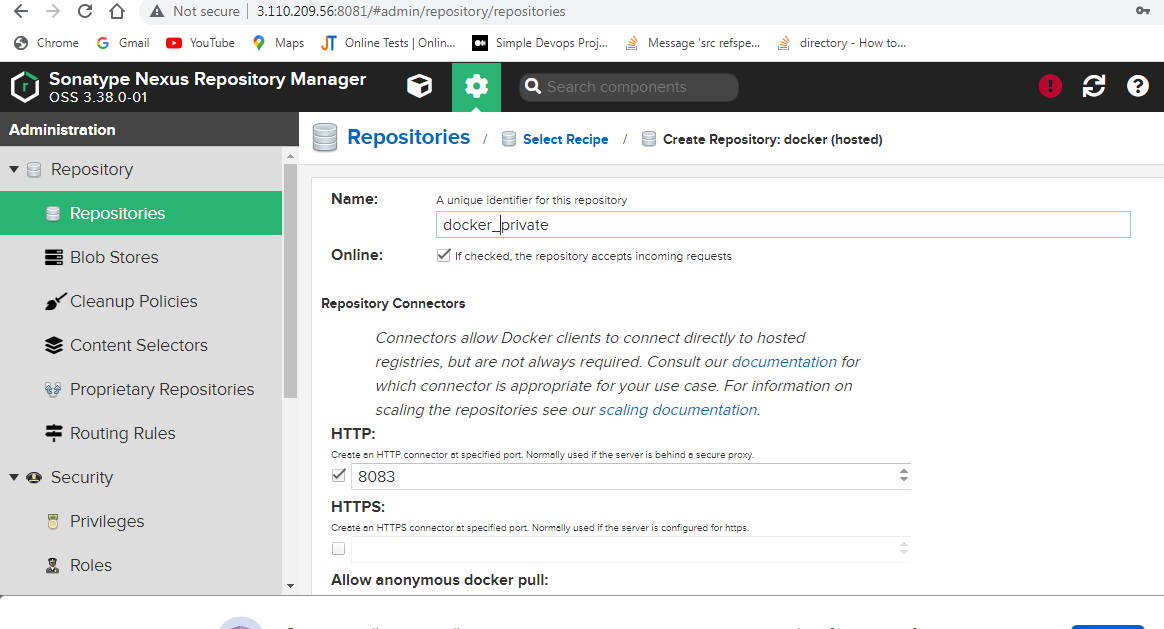
# Go to nexus console – repositories – docker(proxy) – docker(hosted).

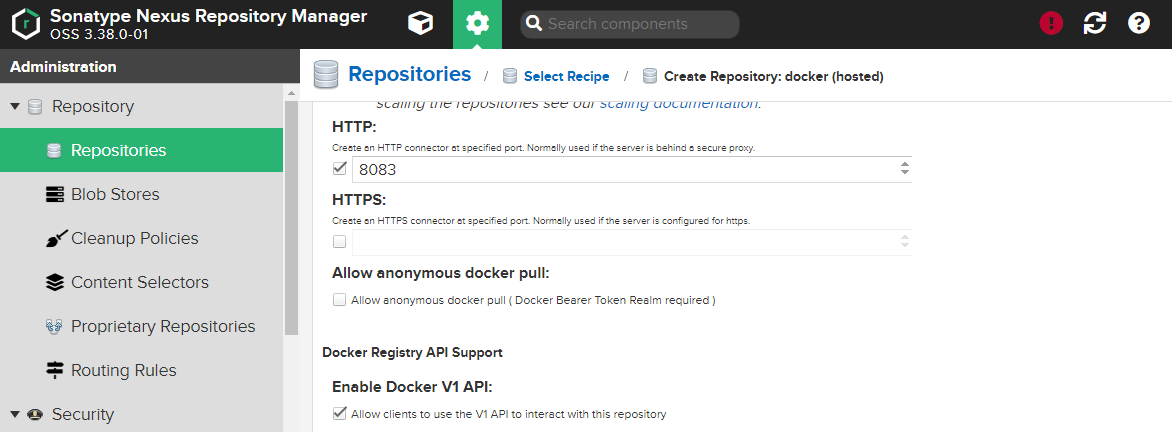
# Name : docker private

# http: 8083

# select docker VIAP1

****

****

****

# Go to jenkins server and create a file called daemon.json.

# cd /etc/docker/daemon.json

# vi daemon.json

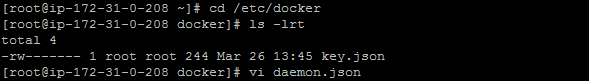
{

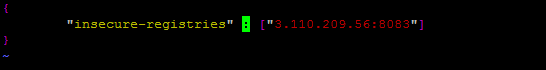
“insecure-registries”:[“nexusserverIP:8083”]

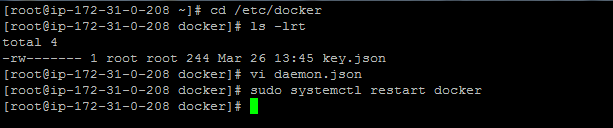
}

# Restart the docker daemon by the following command.

* sudo systemctl restart docker

****

****

****

**#** Run the script.

# Push the image to nexus repository using this script.

stage('Nexus Image Push'){

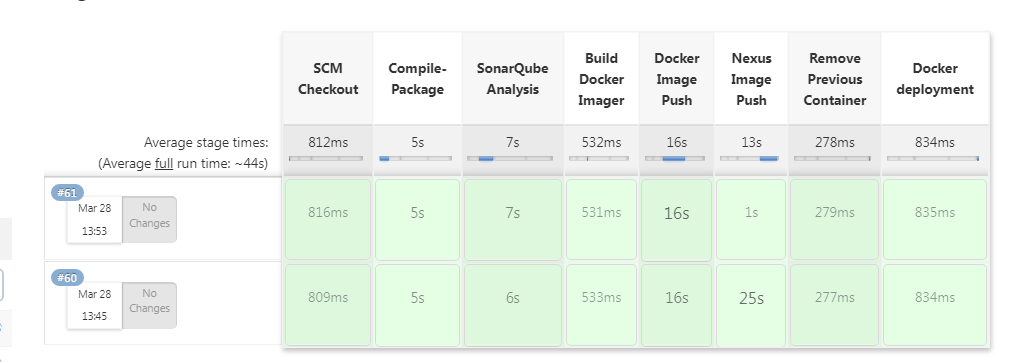
sh "docker login -u admin -p admin123 3.110.209.56:8083"

sh "docker tag premmano/myweb:0.0.2 3.110.209.56:8083/prem:1.0.0"

sh 'docker push 3.110.209.56:8083/prem:1.0.0'

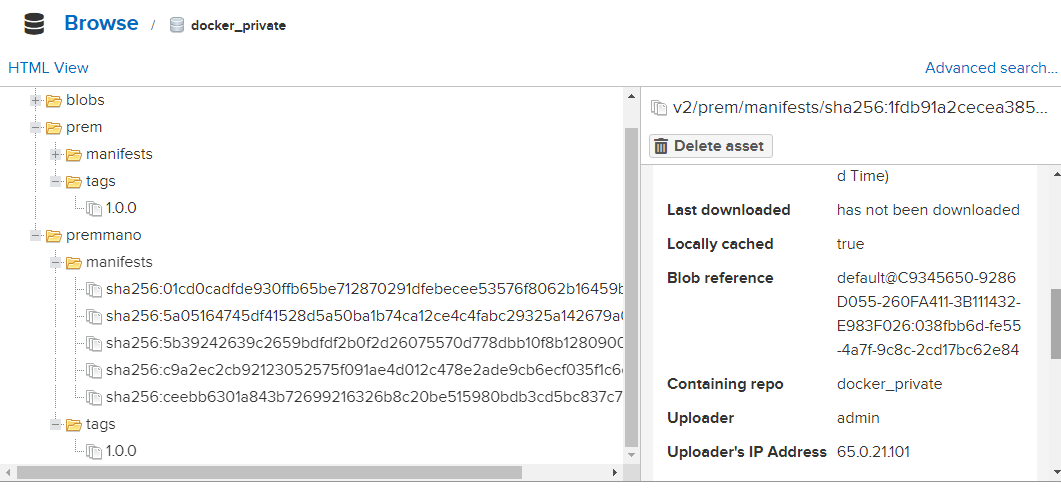
}

****

****

# Go to nexus console – browse – docker private – prem – 1.0.0

# Now docker images pushed to nexusconsole.

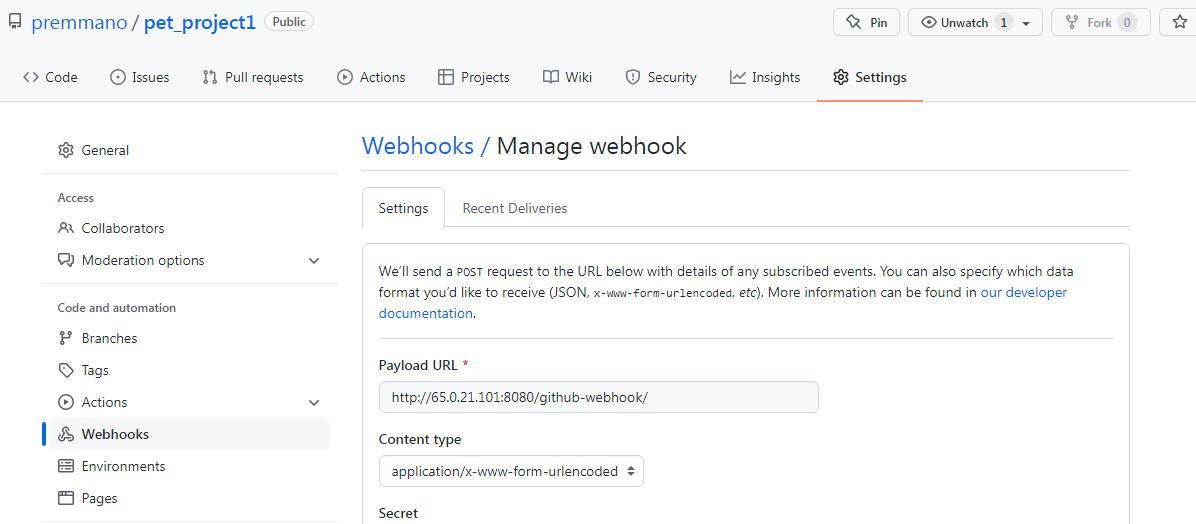
****

**INTEGRATE GITHUB AND JENKINS:**

# Go to github – setings – webhooks – add webhook

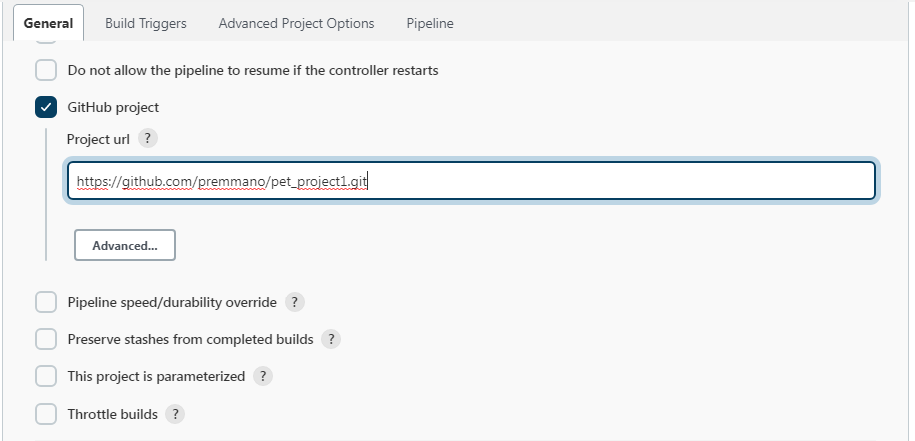
# Attach http://jenkins server ip:8080/github-webhooks/ on webhook place.

# context type: Application/json

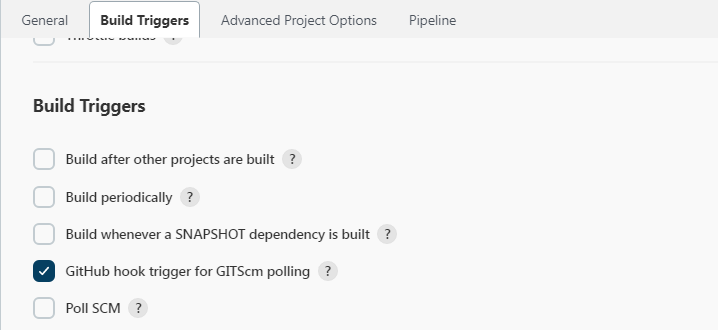
****

# Go to jenkins console – configuration – general – github project

# Copy github pet\_project1 url (https://github.com/premmano/pet\_project1/my-app.git) and paste it in project url place.

****

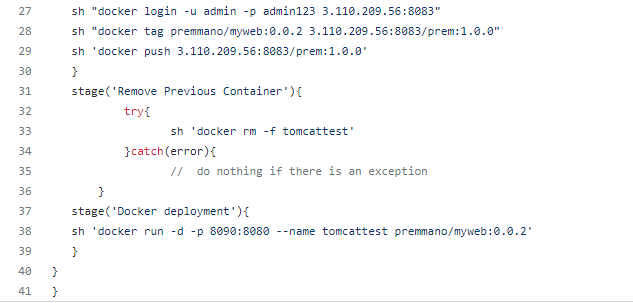
# Go to jenkins console – select “gith hub hook trigger for GIT SCM polling”. whenever commit happens in github, the jenkins pipeline automatically trigger and run the code.

****

# Go to git hub – code – jenkins file

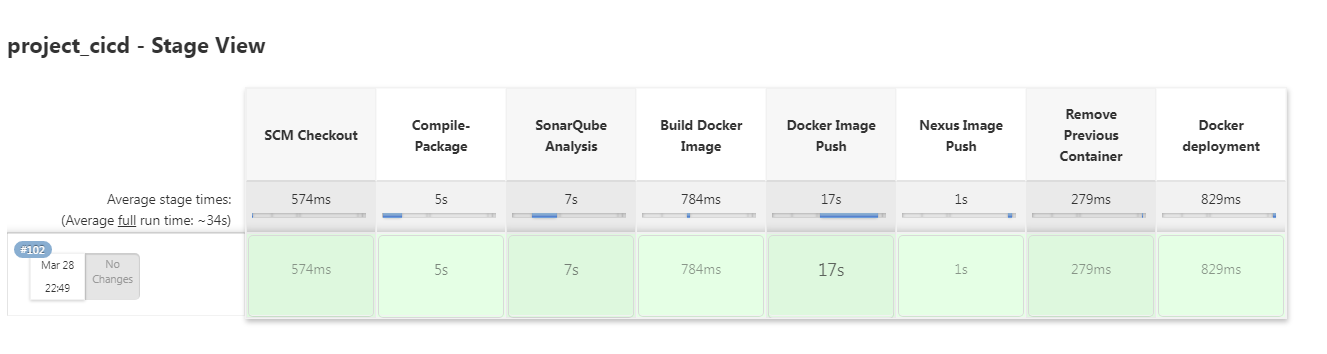
# Do commit or changes in jenkins file.

****

****

# Go to jenkins console

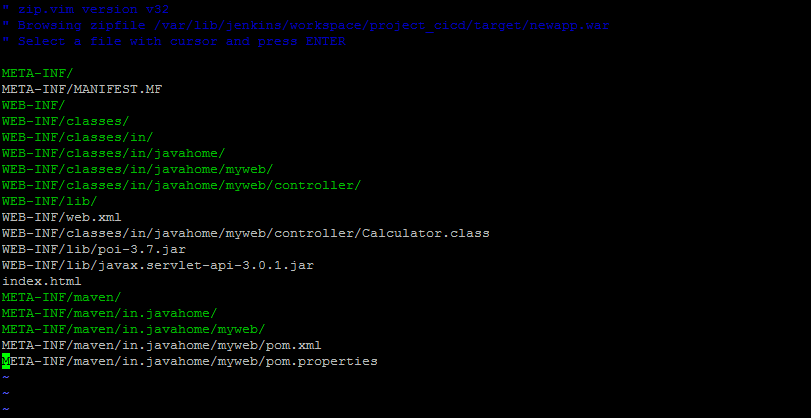
# It automatically run the code.

****

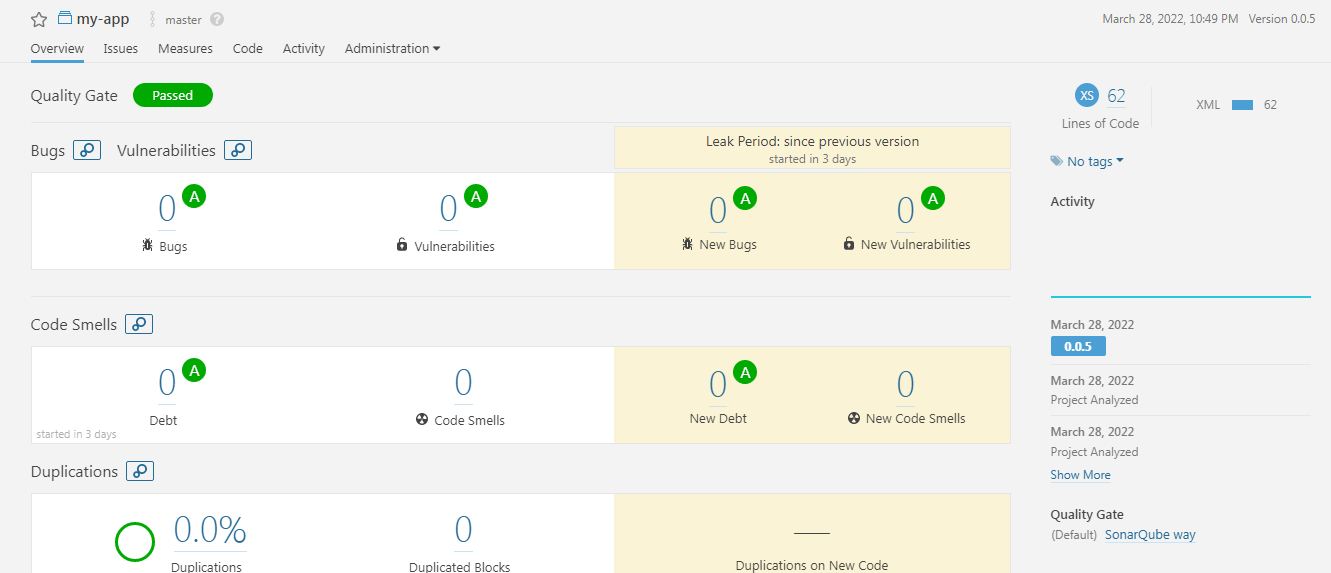
**RESULT:**

The below snapshot gives the result of each tool used.

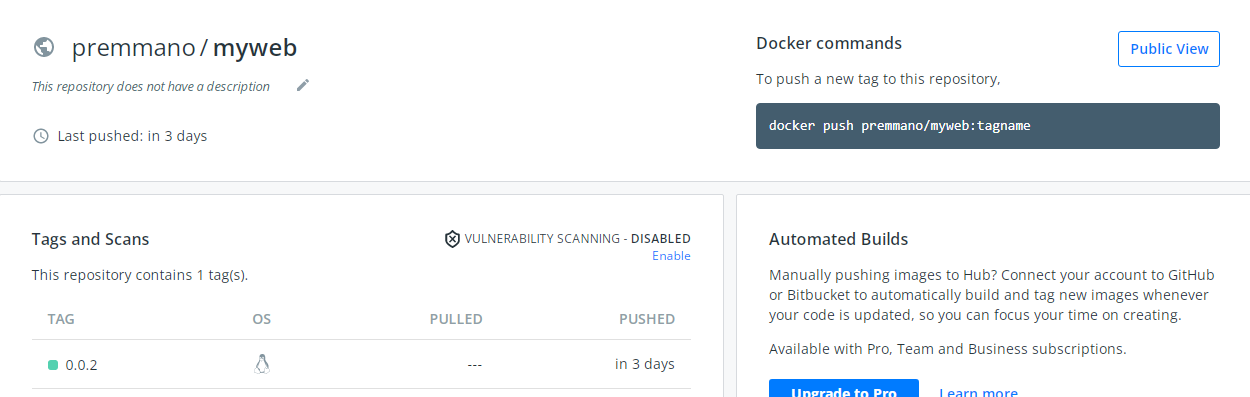
**Maven:**

****

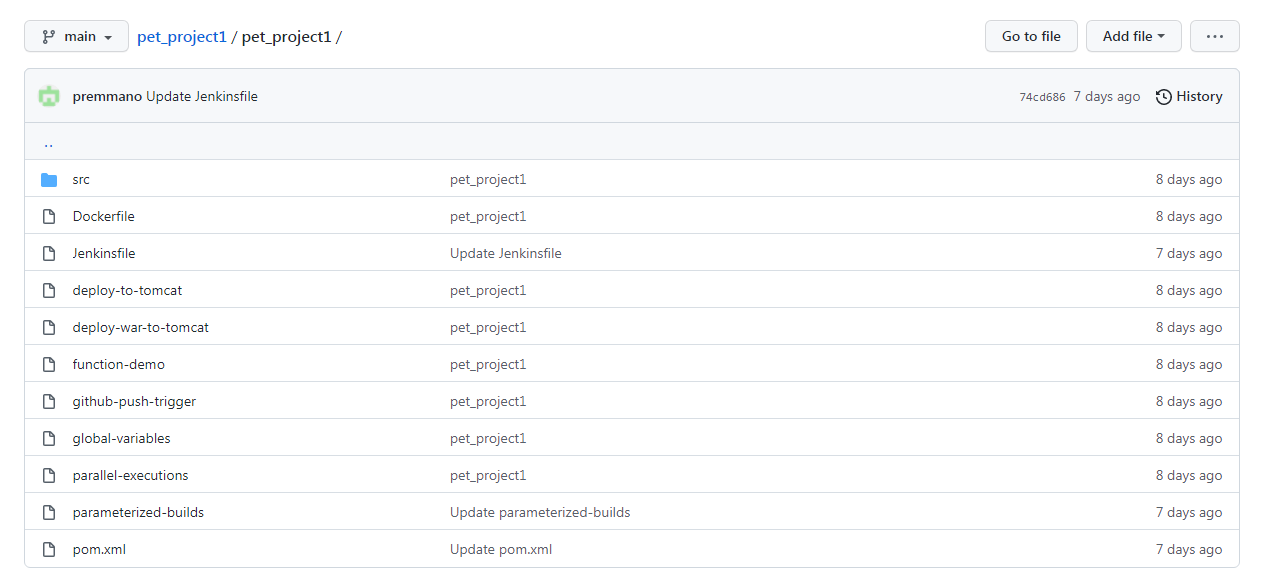
**SonarQube:**

****

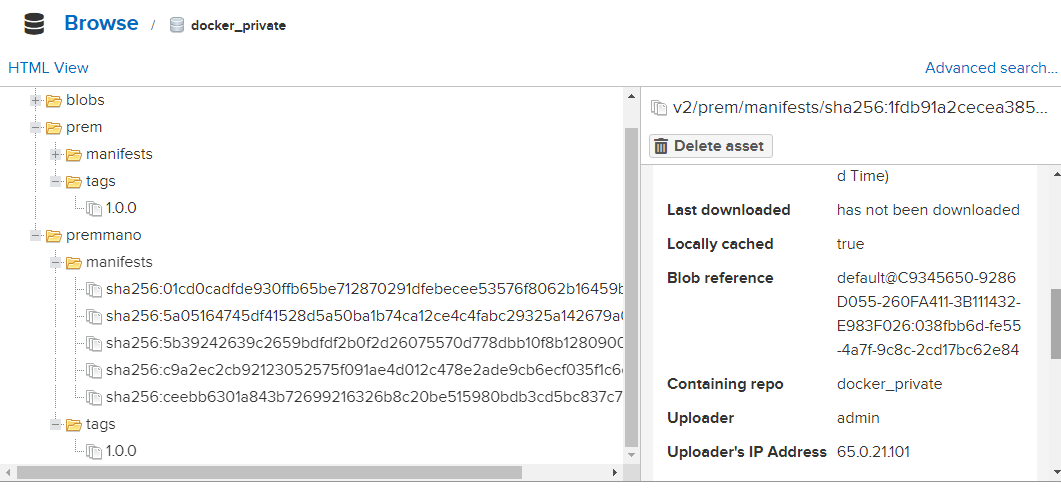
**Docker**

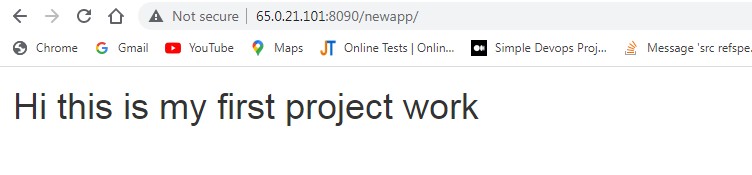
****

**Github**

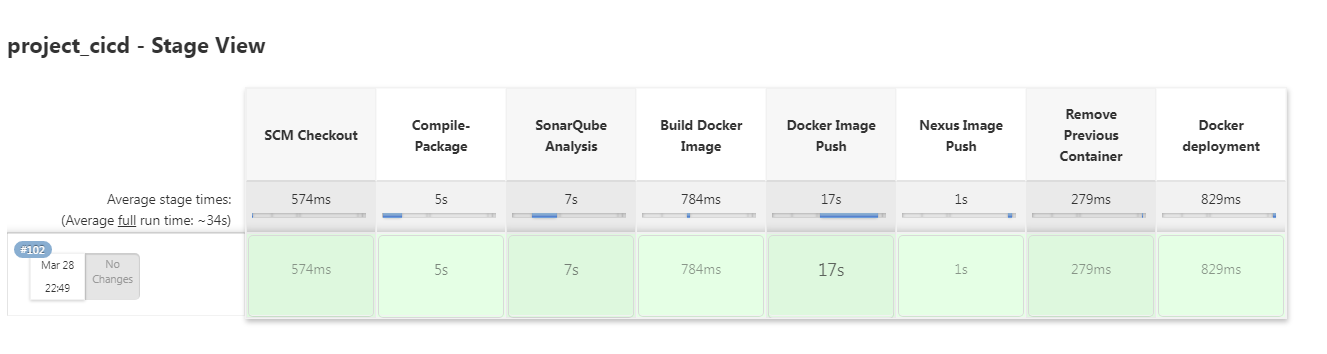
****

**Nexus:**

****

****

**Jenkins:**

****

**CONCLUSION:**

This proposed project is successfully done using DeVops methodology. DevOps is a software development methodology that escalates to the amalgamation between software developers and information technology (IT) operation professionals. Its focuses mainly on delivering software product faster and reducing the failure rate of releases to make the product efficient. This system will be helpful for the developers or testers who need to fix the bugs rapidly and want to add extra features to the existing product according to the client requirement. At present DevOps is the most advanced approach in IT industry than waterfall model and agile model. This proposed project is successfully implemented using the DeVops methodology using aws, git, jenkins, maven, Sonarqube, docker and nexus.