DEVOPS CI &CD PROJECT

**Project Objective:**

To create a Pipeline project for Automating Continuous Integration and Continuous Deployment in Jenkins.

**Tools Used:**

1. GIT & GIT Hub -Version Control Tool & GIT Repository
2. JENKINS -Integration Tool
3. MAVEN -Build Tool
4. DOCKER -Container
5. TOMCAT -Server For Run Application
6. SONARQUBE -code quality check Tool
7. NEXUS - Private Repository For Source Code
8. AWS Resources -Ec2, RDS

**Steps to follow:**

1. Instance Creation on AWS:

* Instance Required – AWS Linux, t2.medium, All TCP.
* Login on the Instance

2. Install command GIT:

yum install git –y

3. Installation Command for JENKINS:

|  |
| --- |
| REFER: <https://www.jenkins.io/doc/tutorials/tutorial-for-installing-jenkins-on-AWS/> |

* amazon-linux-extras install epel –y
* yum update –y
* wget -O /etc/yum.repos.d/jenkins.repo <http://pkg.jenkins.io/redhat-stable/jenkins.repo>
* rpm --import <http://pkg.jenkins.io/redhat-stable/jenkins.io.key>
* sudo amazon-linux-extras install java-openjdk11
* yum install jenkins –y
* systemctl start Jenkins

Jenkins Ini PWd: a9fa9718d044a8b8586b2d1223918d7

4. Installation Steps for MAVEN:

\* Maven Download from web, we go to Setup the file path in instance

File Path: cd /opt/

Refer: <https://mirrors.estointernet.in/apache/maven/maven-3/>

Version: maven 3.8.5

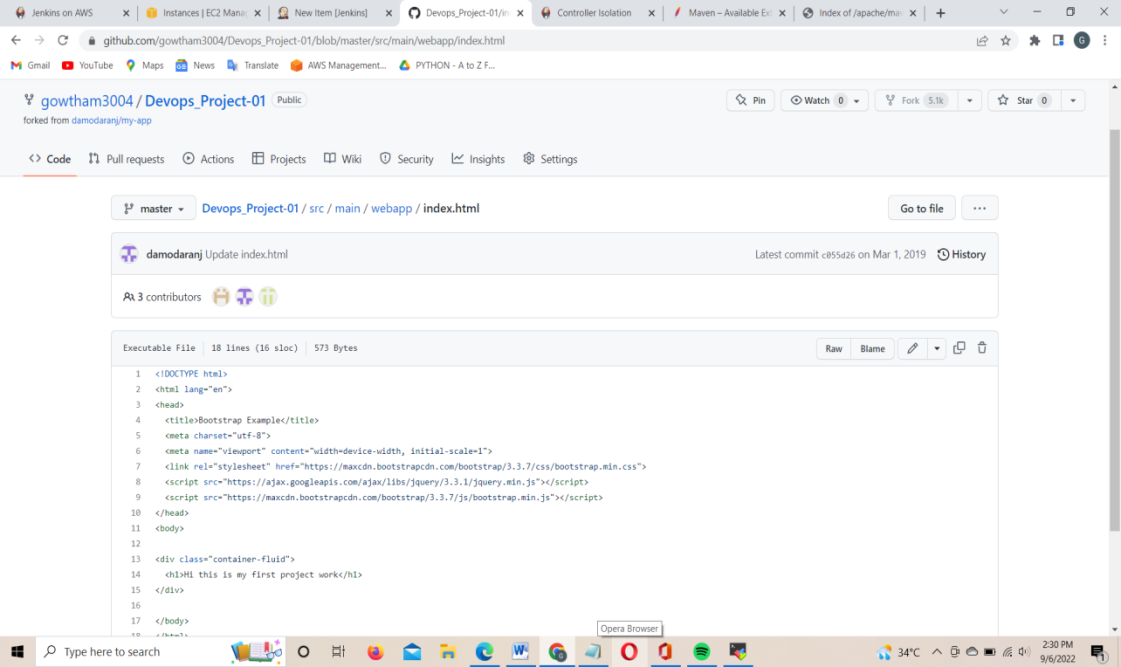
Commands

* wget <https://mirrors.estointernet.in/apache/maven/maven-3/3.8.5/binaries/apache-maven-3.8.5-bin.tar.gz>
* tar –xvf apache-maven-3.8.5-bin.tar.gz

5. Install DOCKER:

* Yum install docker –y

**PUSH or FORK THE SOURCE CODE INTO GIT\_HUB REPOSITORY**



# In Real Time DEVOPS Project, take care the Three Important Files in Source Code are

1. POM.xml---ProjectObjectModel----MavenFile

-Contains [PACKAGE NAME, PACKAGE VERSION, File FORMAT]

Example : [ MyWEB-0.0.5.war]

-Code Dependencies (code, remote, central)

TARGET Folder has Stored the .WAR file

1. Docker File:

-Deployment Process

|  |
| --- |
| From tomcat:8  Copy target/newapp.war /usr/local/tomcat/webapps/  |-># Take the war and copy to webapps of tomcat |

3) JENKINS File:

--Make the Pipeline by Groovy Scripting

**STEPS FOR MAKE A PIPELINE IN JENKINS:**

1. In Jenkins Console, Add New Item 🡪 Name, Project Type: PIPELINE.

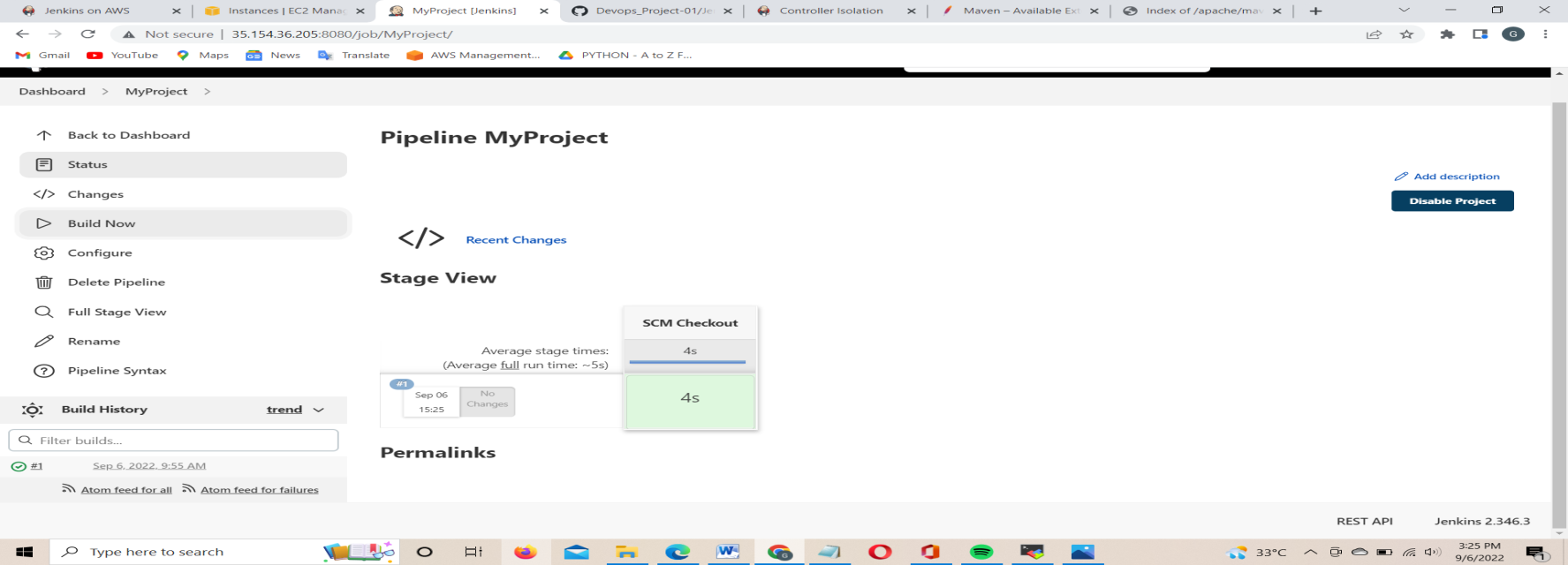
2. Go to Jenkins File and run the Script step by step process:

**Step 1:** **Get The Source code from github to Jenkins local Machine**

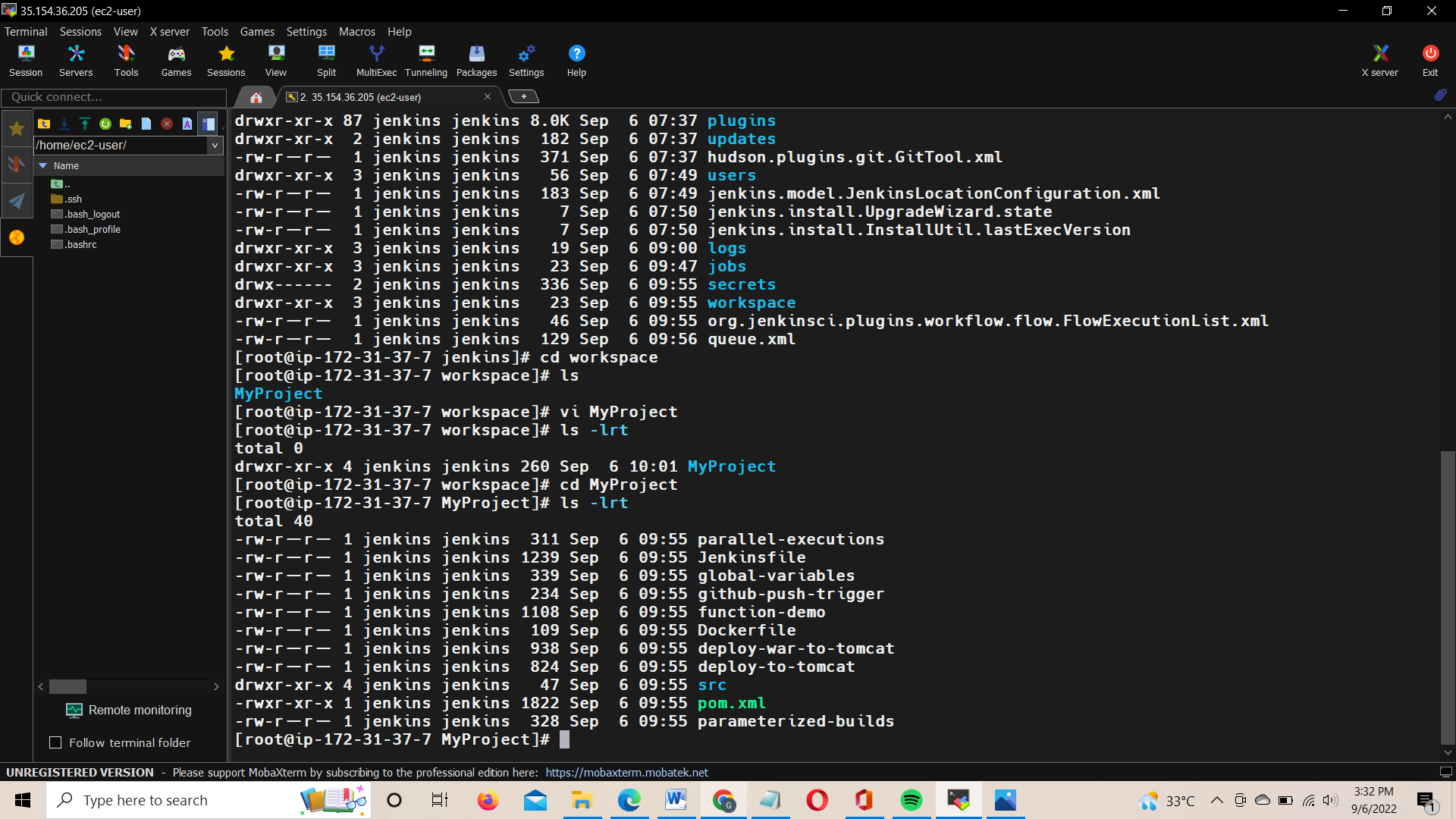
SCRIPT1 :

|  |
| --- |
| node{ |
|  | stage('SCM Checkout'){ |
|  | git 'https://github.com/damodaranj/my-app.git' |
|  | }  } |

Run Script:



Open Local Machine Source Code File is Stored:



**STEP 2: COMPILE PACKAGE**

SCRIPT 2: Groovy Script

|  |
| --- |
| stage('CompilePackage'){ |
|  |  |
|  | def mvnHome = tool name: 'maven3', type: 'maven' |
|  | sh "${mvnHome}/bin/mvn clean package" |
|  | sh 'mv target/myweb\*.war target/newapp.war' |
|  | } |

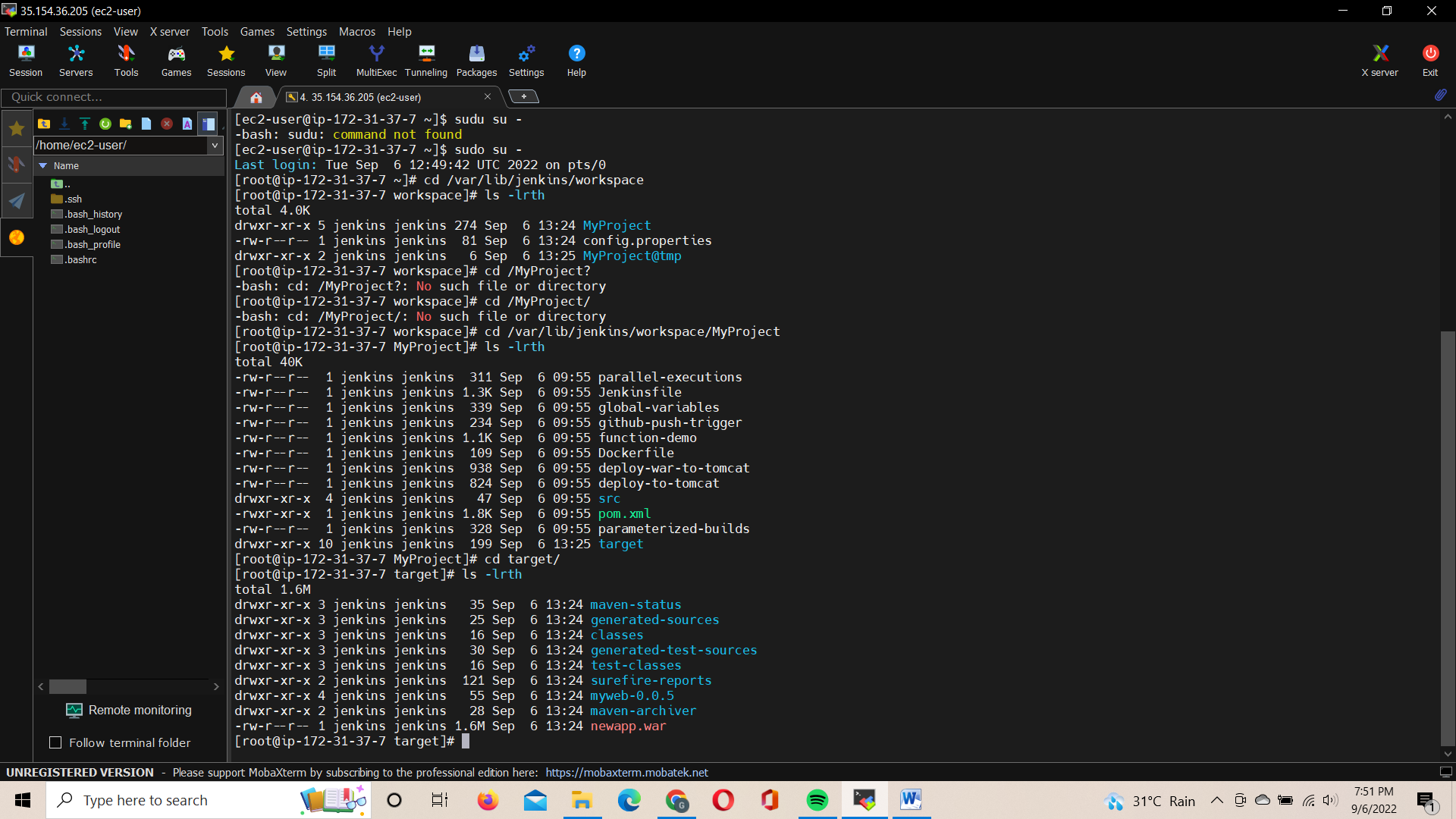
Manually Run Script : ./sh mvn [Path: opt/bin/mvn/]

Configuring Environmental Variable in JENKINS by access Third Party tool (Maven):

* In Jenkins Console: Plug-ins Install- Available [MAVEN INTEGRATION]
* Specific Maven Path in JENKINS By :

Manage Jenkins 🡪Sys.Config 🡪 Global Tool Configuration 🡪 ctrl+f (maven) 🡪Add maven [alias Name:Maven3, Path: /opt/apache…./ ]🡪Save.

* Run the Script
* Maven and Jenkins are integrated
* TARGET Folder created: [var/lib/Jenkins/workspace/MyProject/target/newapp.war]



**STEP3: DOCKER LAUNCH**

**SCRIPT 3 : DOCKER IMAGE & PUSH IMAGE INTO DOCKERHUB**

**stage('Build Docker Imager'){**

**sh 'docker build -t gowtham6666/myweb:0.0.2 .'**

**}**

**stage('Docker Image Push'){**

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

**sh "docker login -u gowtham6666 -p ${dockerPassword}"**

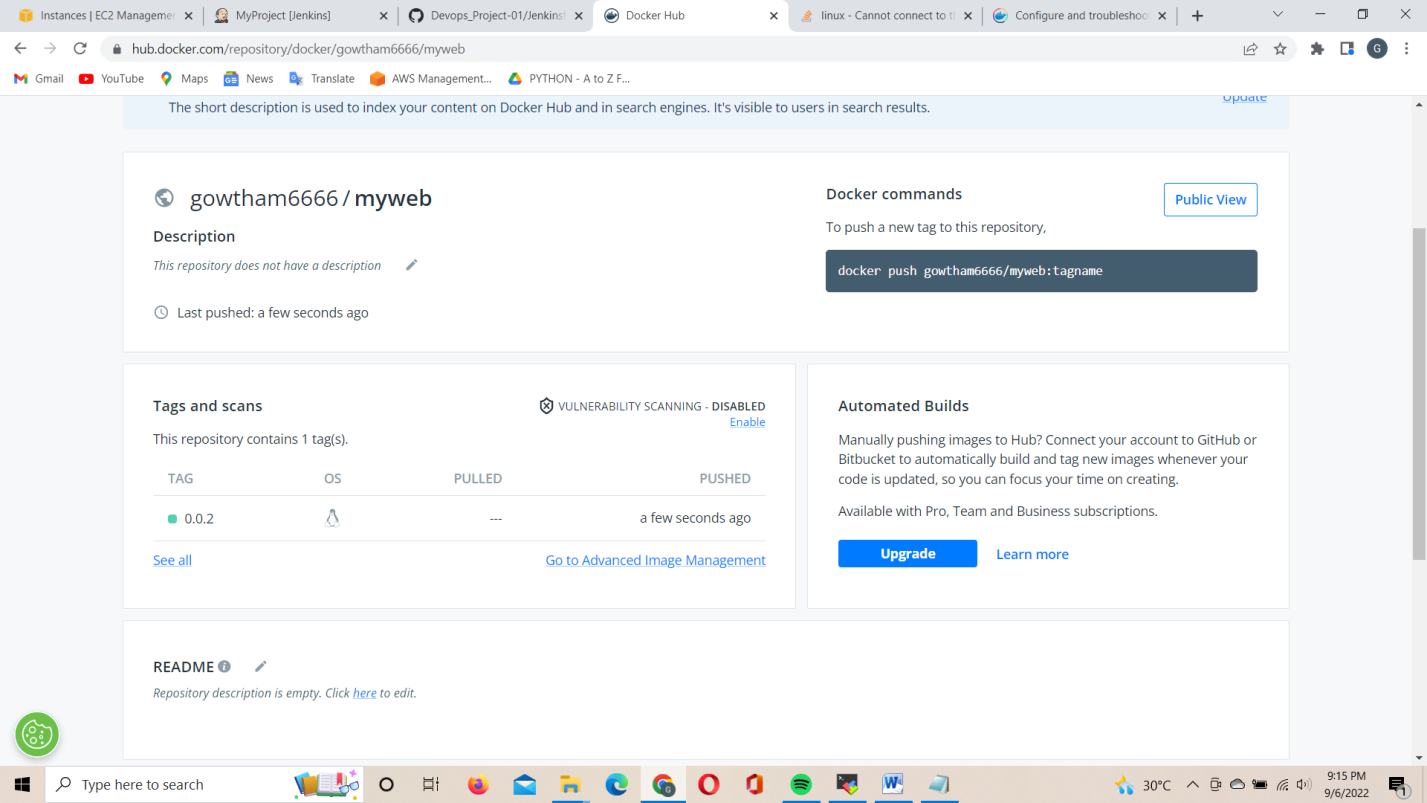
**}**

**sh 'docker push gowtham6666/myweb:0.0.2'**

**}**

**NOTE:** Here Credentials given to Jenkins to access DOCKERHUB

|  |
| --- |
| IQ: How to pass Credential to Jenkins?  Manage Jenkins 🡪 Manage Credentials 🡪Jenkins 🡪 Global Credential(restricted) 🡪 Add credentials 🡪 Kind (Secret Text) 🡪 Secret[DockerHub Password], ID[variable alias Name eg:DockerPass] |



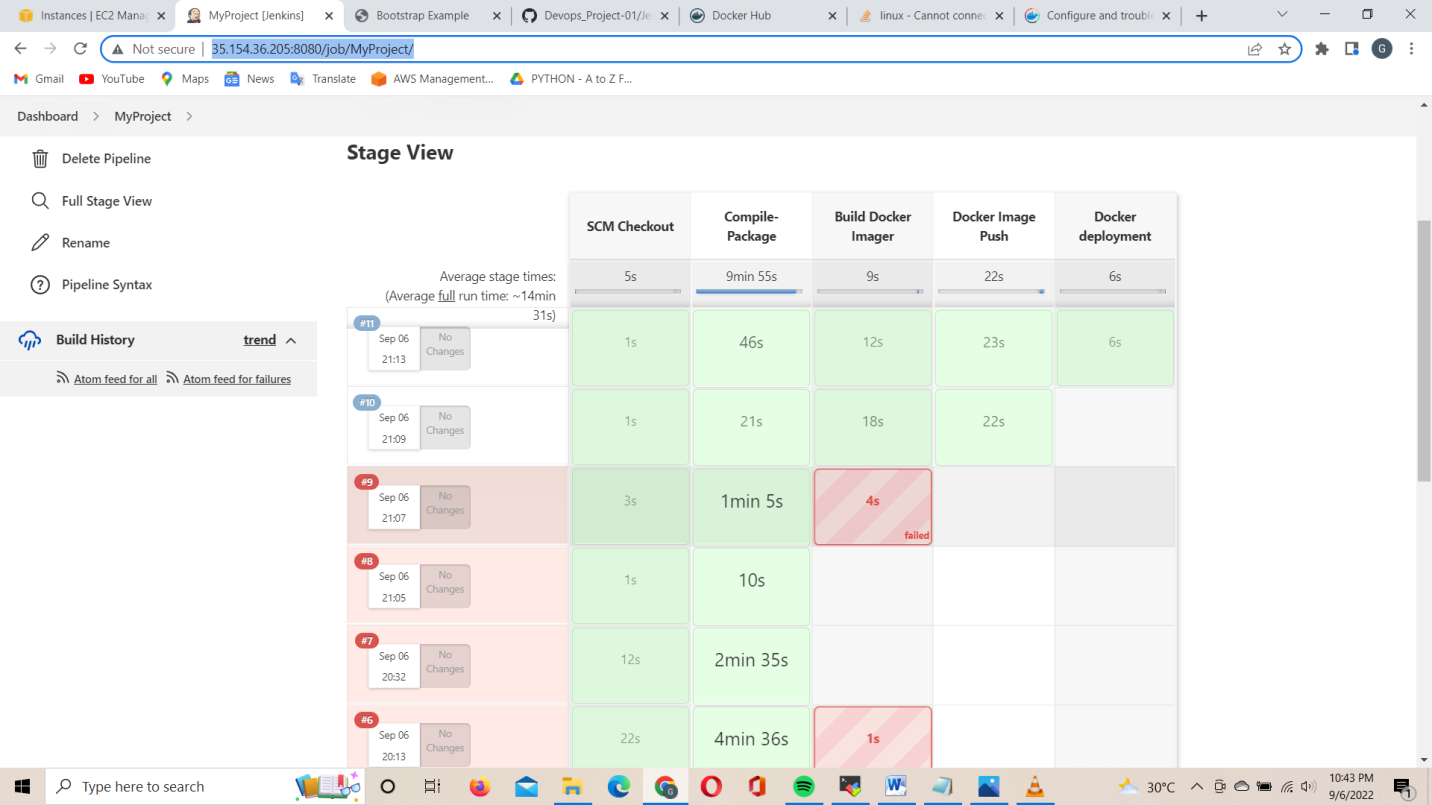
**SCRIPT 4 : DOCKER DEPLOYMENT**

**stage('Docker deployment'){**

**sh 'docker run -d -p 8090:8080 --name tomcattest gowtham6666/myweb:0.0.2'**

**}**

* **Run The Script in pipeline**



* **Docker has done deployment, ACCESS :** [**http://35.154.36.205:8080/job/newapp**](http://35.154.36.205:8080/job/newapp)

### SONARQUBE INSTALLATION:

-It is CODE quality check Tool and Validating Developer Code[Clean & safe Code]

- It support most integration tools and version control tool.

**Pre-Requisites :**

1. Database Server [Github🡪SonarQube🡪Database(store validating code)]
2. Min 3GB Ram machine Need
3. JAVA Open JDK
4. SONARQUBE two important Files[1. Wrapper.config, 2.sonar.properties]
5. DATABASE – create users and access the data from database
6. SONARQUBE Not Start as Root User

**Steps to follow:**

1. Create RDS Database [ Mysql v5.7.33, Free-tier]

2. Launch a EC2 linux instance for database server(t2.medium)

3. Install Jdk 1.8.0 on the Sonar Machine

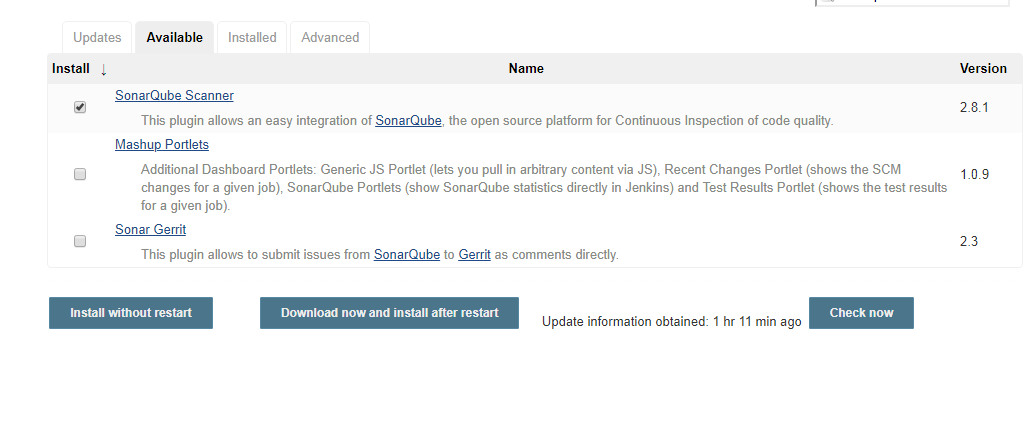
4. **COMMANDS To follow configuration in sonarqube server**

* Integrate Database to Sonarqube by add End point of Mqsql.
* Run query on Mysql
* Installation command for Sonarqube
* # cd /opt
* # wget <https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-6.7.6.zip>
* # unzip sonarqube-6.0.zip
* # mv /opt/sonarqube-6.0 /opt/sonar
* **Configure two important files in sonarqube**
* **Wrapper.config**

Set the java file path

Java command = /usr/lib/jvm/java-1.8.0-amazon-corretto.x86\_64/jre/bin/java

|  |
| --- |
| Command for check java installation path:  Sudo update –alternatives –confif java |

* **Sonar.properties**
* File Name: /opt/sonar/conf/sonar.properties
* sonar.jdbc.username=sonar
* sonar.jdbc.password=sonar
* sonar.jdbc.url=jdbc:mysql://<RDS\_DATABAE\_ENDPOINT>:3306/sonar?useUnicode=true&characterEncoding=utf8&rewriteBatchedStatements=true&useConfigs=maxPerformance&useSSL=false
* sonar.web.host=0.0.0.0
* sonar.web.context=/sonar
* Start the sonarqube in default user not root user, so we change the ownership by: **chown –R ec2-user:ec2-user /opt/sonarqube-6.7.6**
* Start SonarQube service
* Install sonar plugins in jenkins
* Go to sonarqube console with 9000 port and create token

**d1c4d7a3920b36f644ede905dd922877b4de9092**

Add Sonar credential in jenkins:

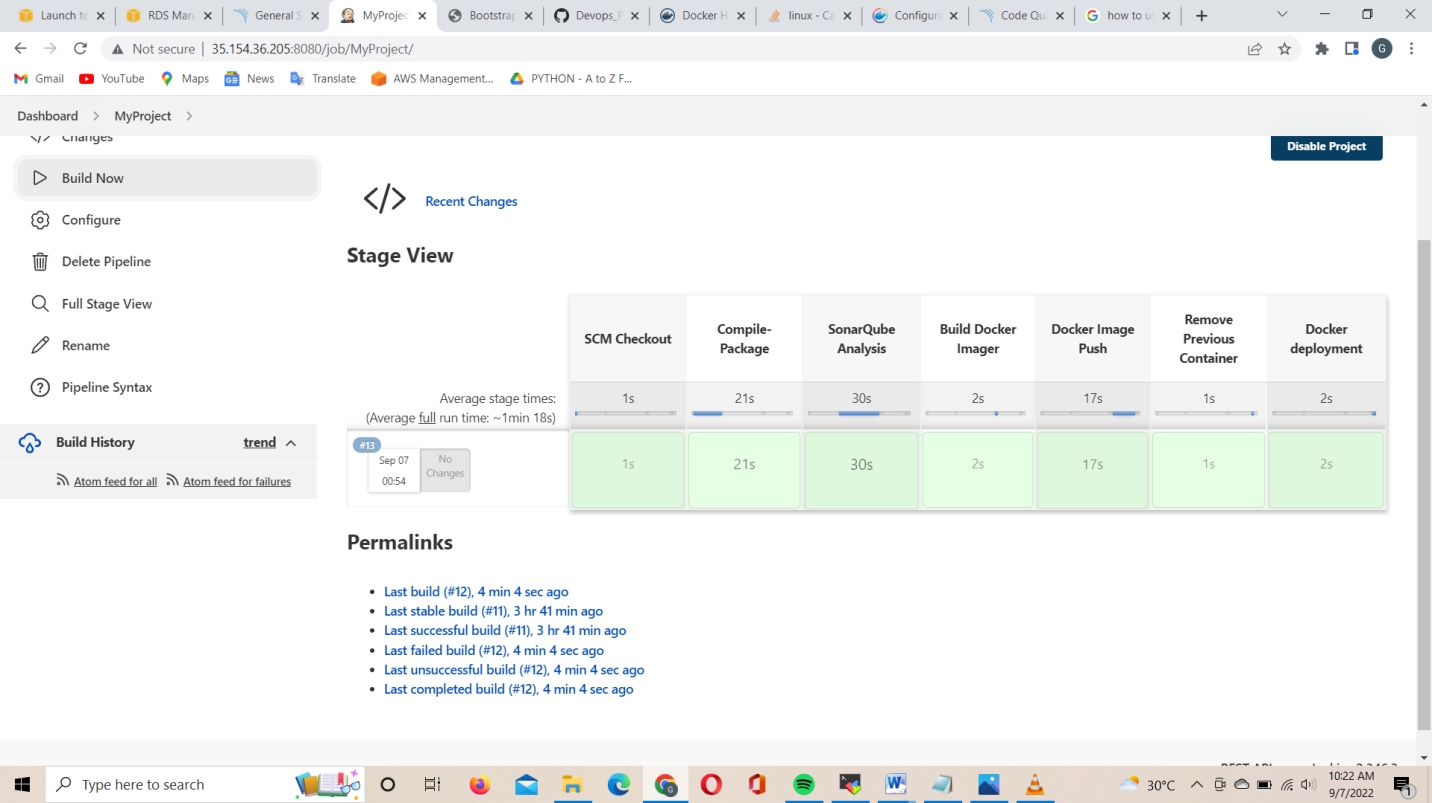
**Manage Jenkins 🡪manage credentials 🡪 Jenkins🡪 global credentials 🡪Secret text[ secret: token, Id: sonar] 🡪 Save.**

**Integrate Sonarqube in Jenkins:**

**Manage Jenkins 🡪Configure system🡪Search Sonar🡪 Name: Sonar, URL: sonar Ip , Add created token🡪 save**

**Run the script in pipeline:**

|  |
| --- |
| stage('SonarQube Analysis') { |
|  | def mvnHome = tool name: 'maven3', type: 'maven' |
|  | withSonarQubeEnv('sonar') { |
|  | sh "${mvnHome}/bin/mvn sonar:sonar" |
|  | } |
|  | } |



**Nexus Installation:**

NEXUS Pre-requisites are :

* Any OS, with 4 GB RAM
* Installation of Java

**Steps to follow:**

* Create EC2 instance (For Nexus) – Linux OS, Instance – t2.medium, SG – All TCP. 10. Open the machine…
* sudo su – root cd /opt
* wget [https://download.sonatype.com/nexus/3/latest-unix.tar.gz](https://download.sonatype.com/nexus/3/latest-unix.tar.gz%20%20)
* tar -xvzf latest-unix.tar.gz yum
* install java-1.8.0
* cd nexus-3.33.1-01 cd bin
* ./nexus start
* ./nexus status
* Now check in Google search “Public IP:8081”
* Open Nexus console Username : admin Password : Paste the text available in ( /opt/sonatypework/nexus3/admin.password)

[\*After that you can change your password\* Select “Enable anonymous access” Go to setting => Create repository - Select ”docker (hosted)” - Name : nexus - HTTP : 8083 - Enable Docker VI API]

* Installation of docker in nexus machine yum install docker –y service docker start vi /etc/docker/daemon.json { "insecure-registries" : ["Public IP of NEXUS machine:8083"] } service docker restart
* Open Jenkins machine vi /etc/docker/daemon.json

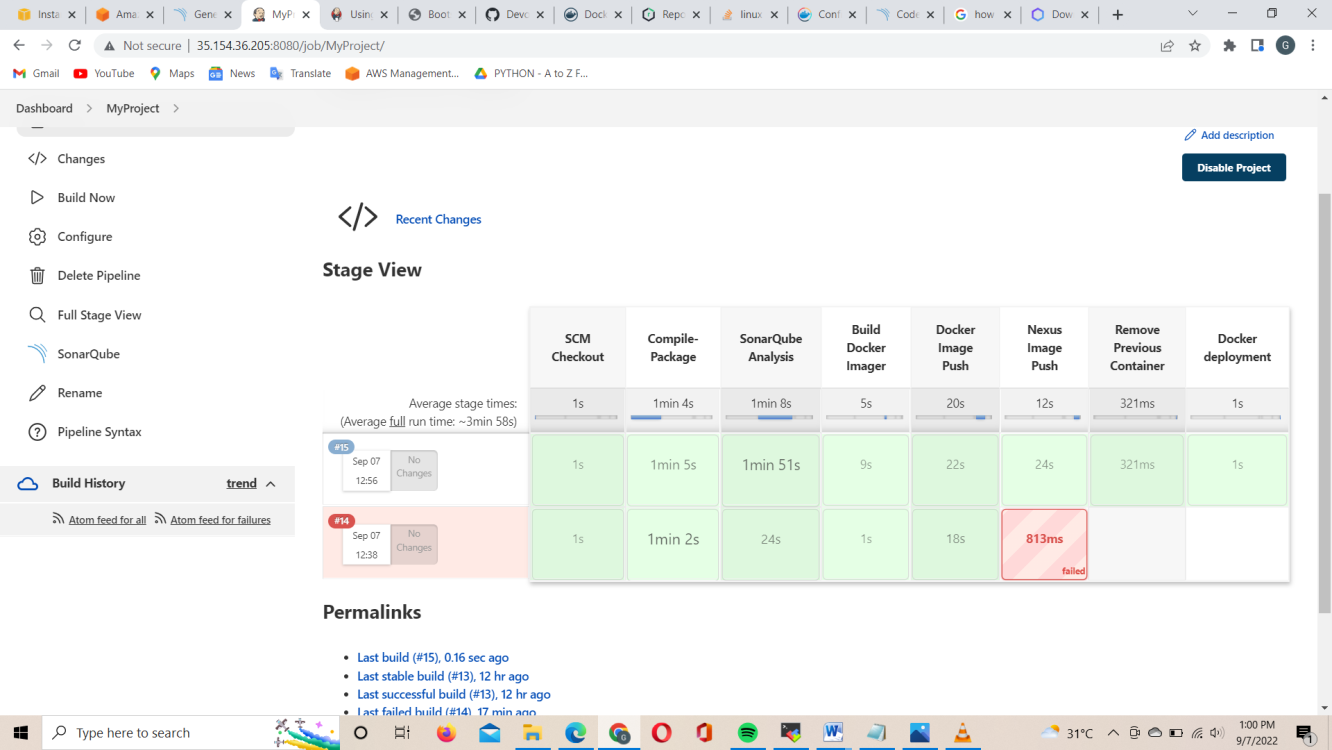
{

"insecure-registries" : ["Public IP of NEXUS machine:8083"]

}

* service docker restart

Finally run the Jenkins file Script and get the output:



INTEGRATION OF GITHUB & JENKINS FOR AUTOMATION PROCESS

=> Steps involved in GITHUB

* Go to your github account
* Open this project file
* Select “Setting” => Webhooks => Payload URL : <http://52.66.238.26:8080/github-webhook/>

=> Steps involved in JENKINS CONSOLE

1. Open the project configure => Build Trigger => Project URL : Paste your github page URL => Choose “Github hook trigger for git SCM policy”

‘NOW, HEREAFTER WHILE YOU COMMIT IN YOUR GITHUB ACCOUNT IT WILL AUTOMATICALLY START THE INTEGRATION PROCESS’

GOWTHAM S

07/09/2022