To run Maven JOB on Jenkins

1. Upload the MEVAN repo to your GITHUB account.
2. Setup the pre-requisite on Jenkins for Maven projects.
    a. Install JDK1.8 on the VM were the project/job would run
    b. Configure the PATH on Jenkins for the JDK1.8
    c. Configure git PATH on Jenkins
    d. Configure Maven PATH on Jenkins
3. Create the Maven project with only the Maven Build part, and test the output.
4. Install the Tomcat on the Jenkins server for the Auto Deployment.
5. Continue the JOB creation with the “post build action” to create an complete Auto Deploy JOB.
6. Check the Output.
7. Update the GITHUB and it would trigger the build automatically.

Step 1:

Upload the MEVAN repo to your GITHUB account.

From this link --- https://github.com/Vishwanathms/SampleMaven

Step 2:

The Jenkins server should be up and running. **Refer LM4.1 and LM4.2**

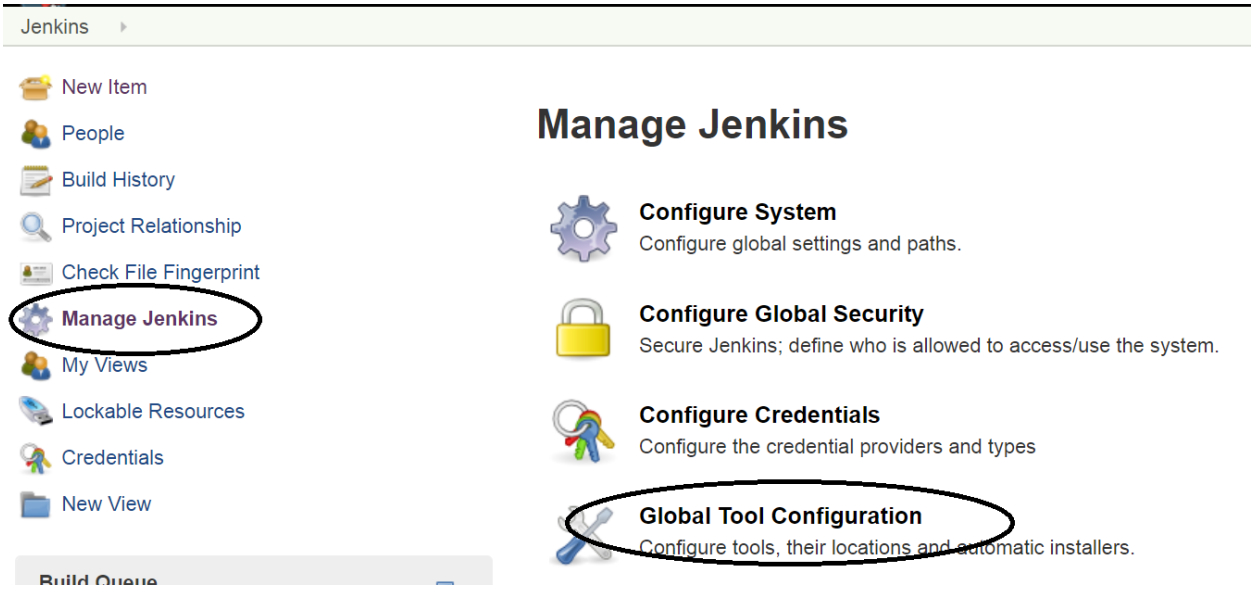
We would be running this Maven project on the Slave.

So below steps need to be done only on the Slave.

Install JDK1.8 on the VM were the project/job would run

```
sudo yum install java-1.8.0-openjdk-devel
```

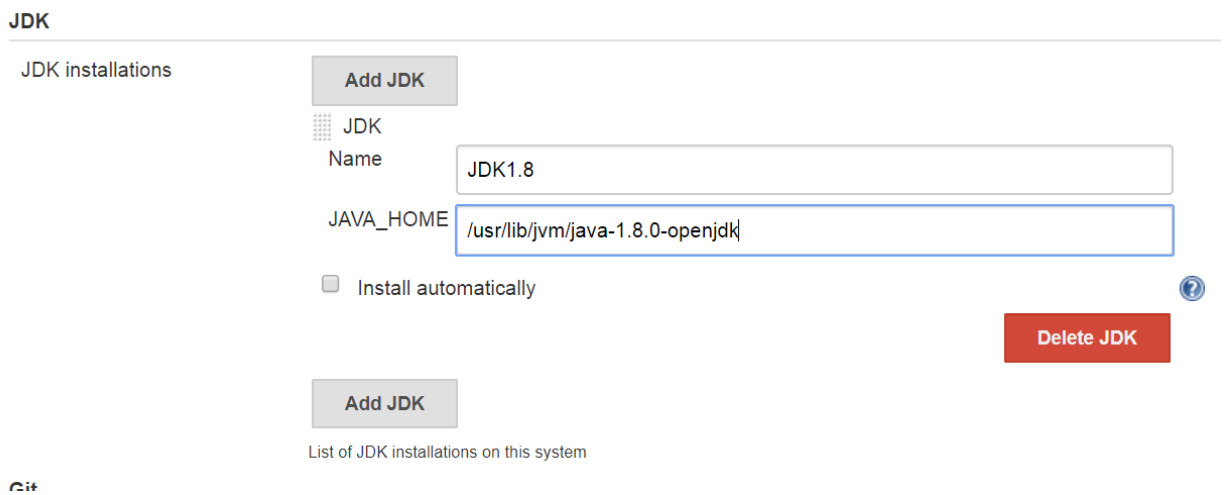
```
echo 'export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk' | sudo tee -a /etc/profile
```



Click on "manage Jenkins"

Then on "Global Tool Configuration".

Configure the JDK on the Jenkins server as below screen

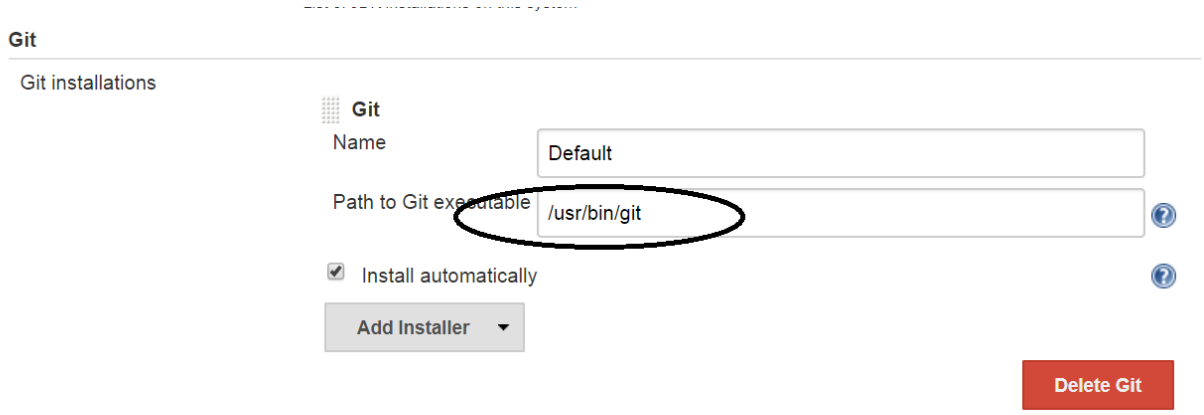


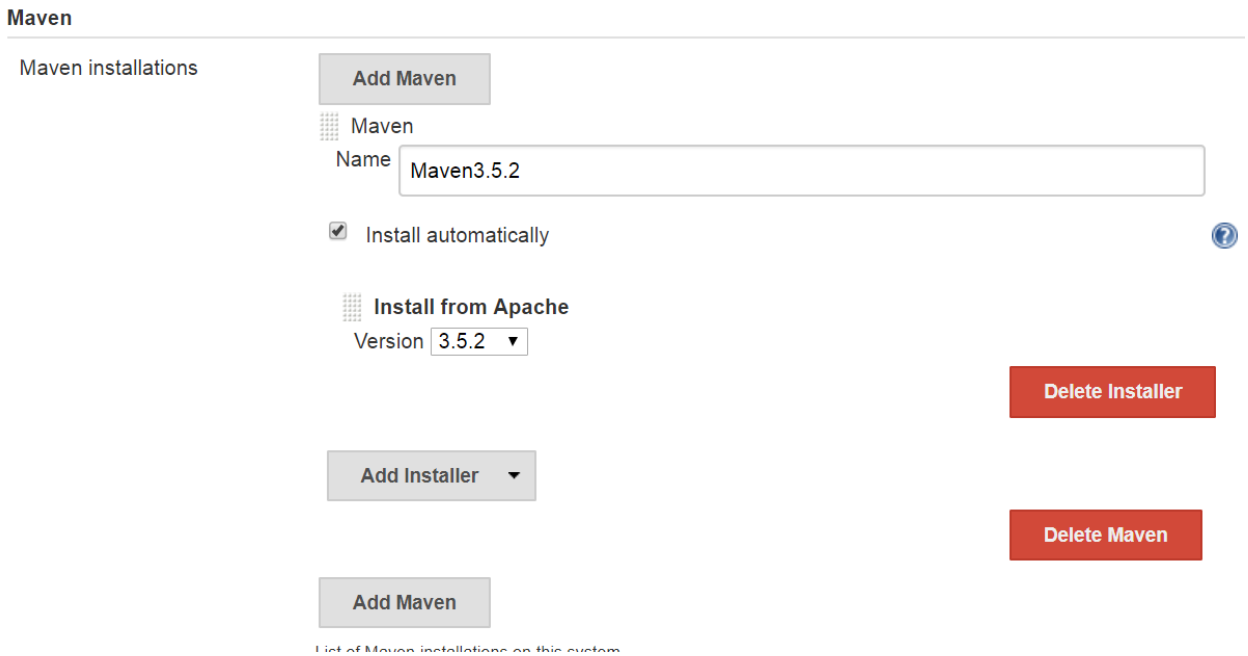
Note:- You may get a warning that the path is invalid. Since we have configured JDK on the Slave , the Jenkins is chcking on the master node for the above mentioned path.

Just ignore and continue

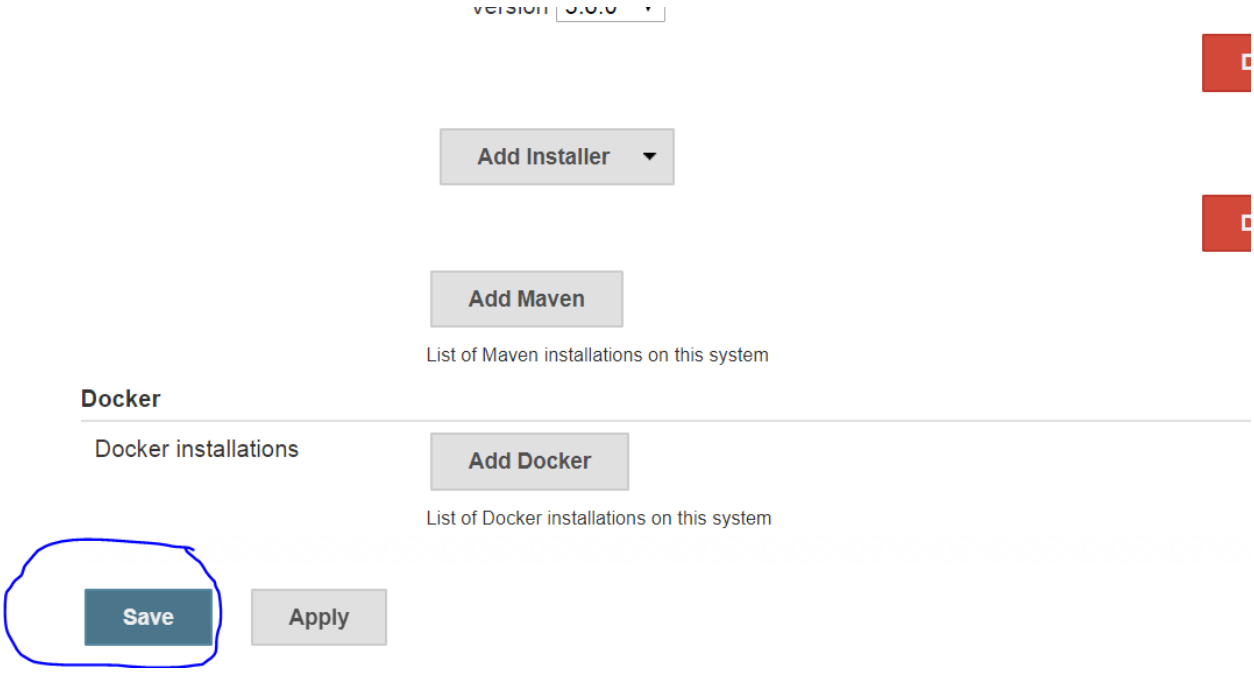
Configure and install GIT, on the local Jenkins machine (master).

In the Path "path of GIT" of the server where GIT is installed.



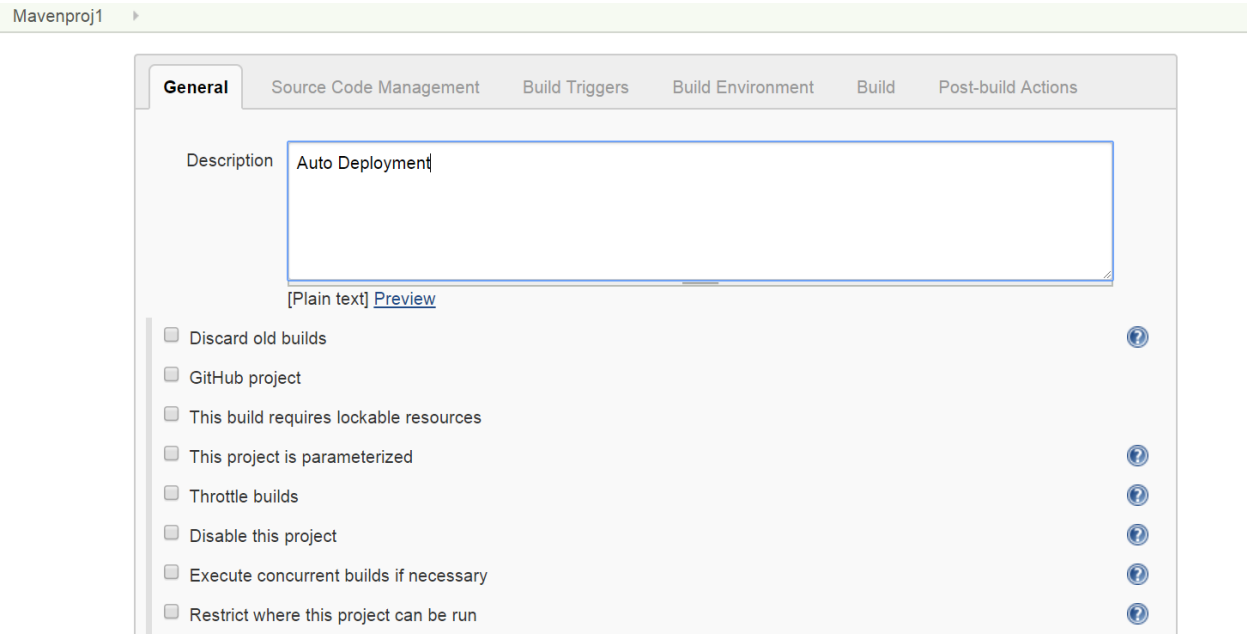


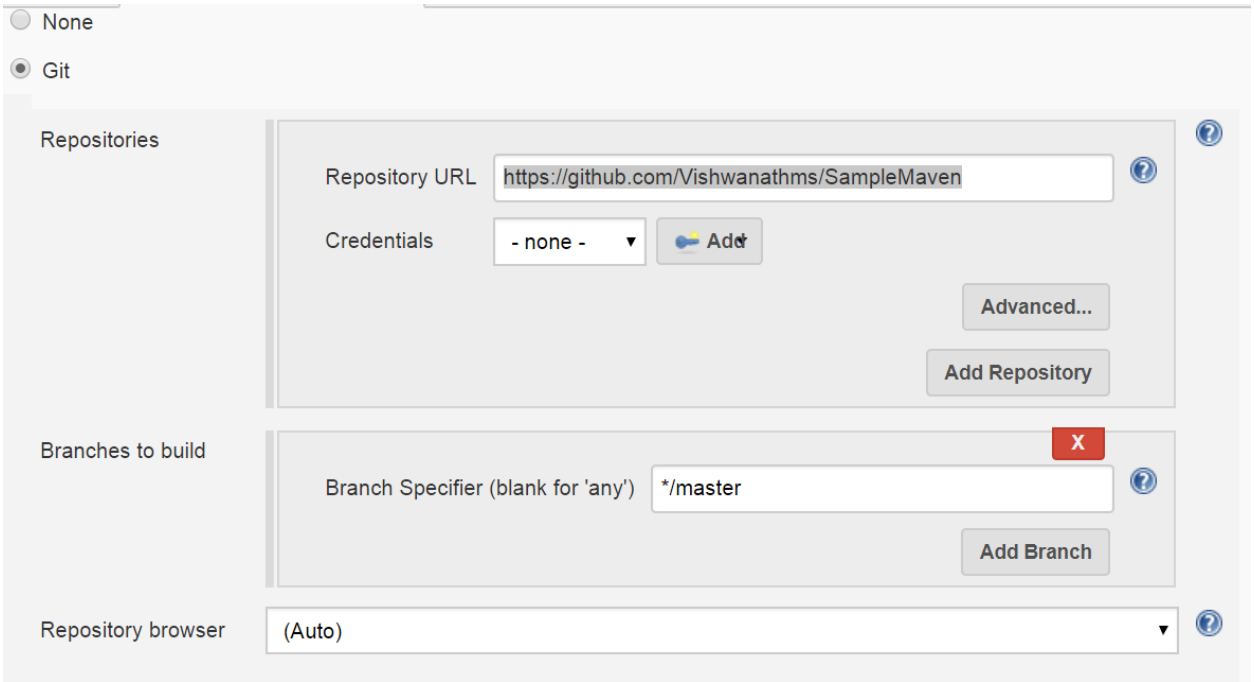
This would install the maven automatically when required to execute the JOB.



Step 3:

Create Maven Project

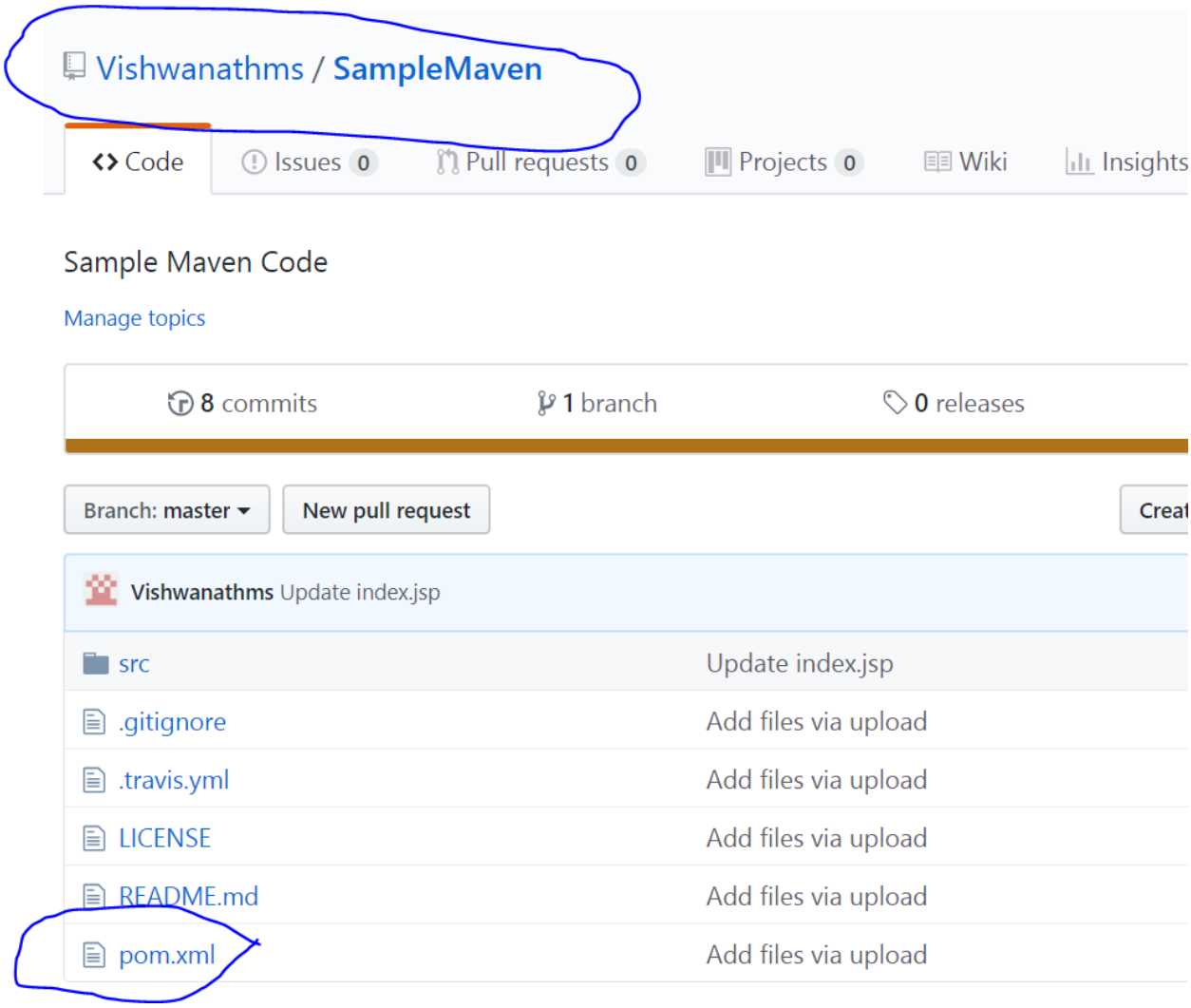




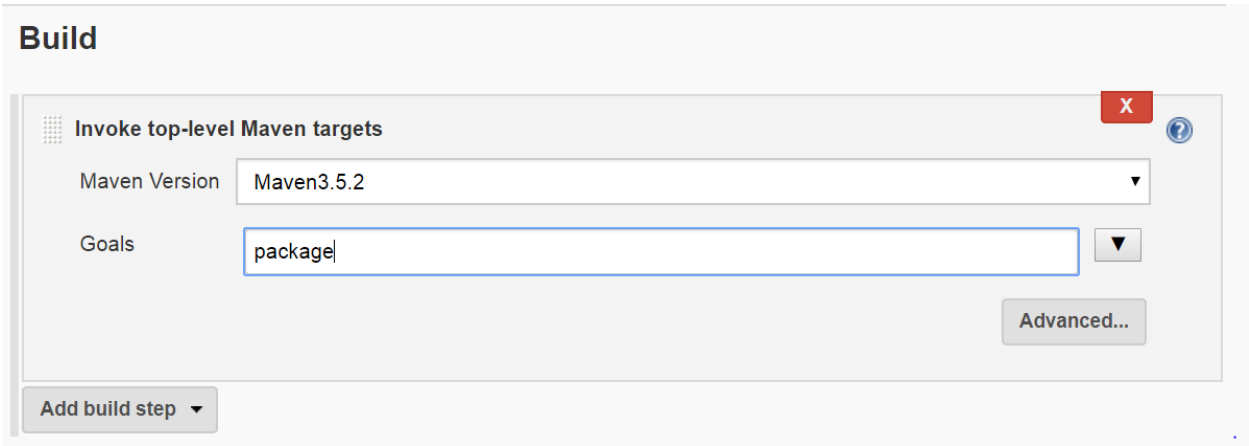
Sample maven code -- https://github.com/Vishwanathms/SampleMaven

Note:-- DO NOT USE MY GITHUB LINK, COPY THE CONTENT TO YOUR OWN GITHUB WITH THE SAME DIRECTORY STRUCTURE.

DON'T PUT THE FILES INSIDE ANOTHER FOLDER.



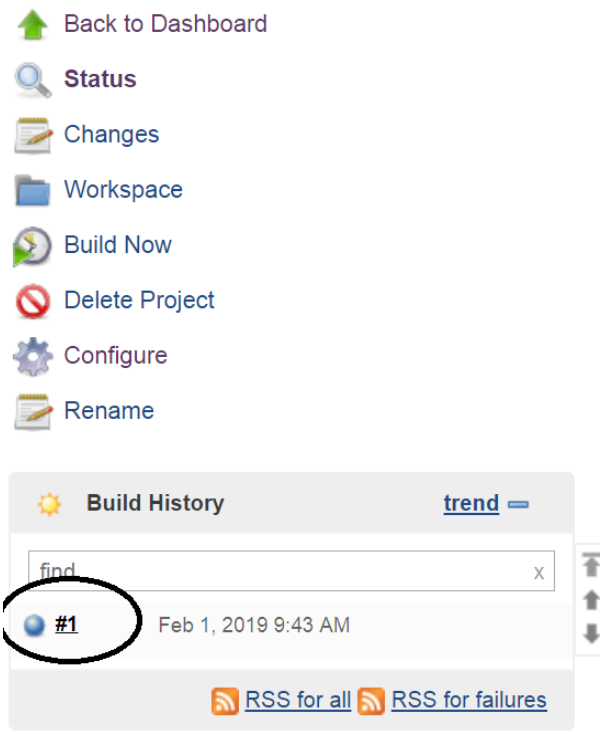
The POM.xml file has to be in the main directory structure.



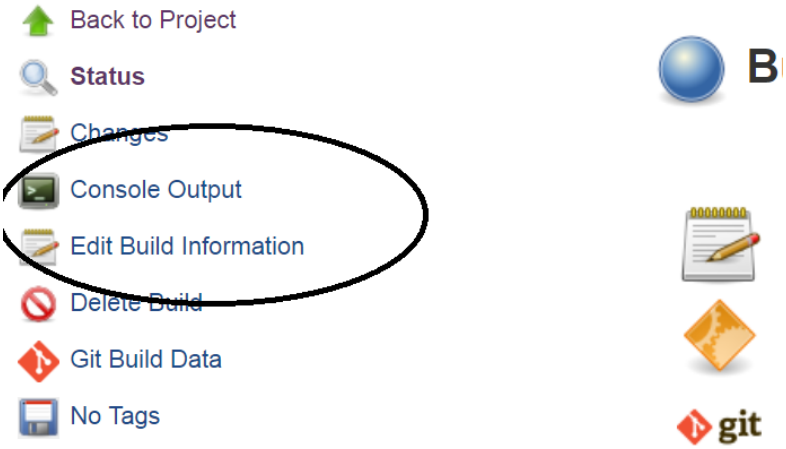
This Build would create an Maven .war file which would be an package file for the source code from GIT.

After this we need to put the jar file to the system to execute it.

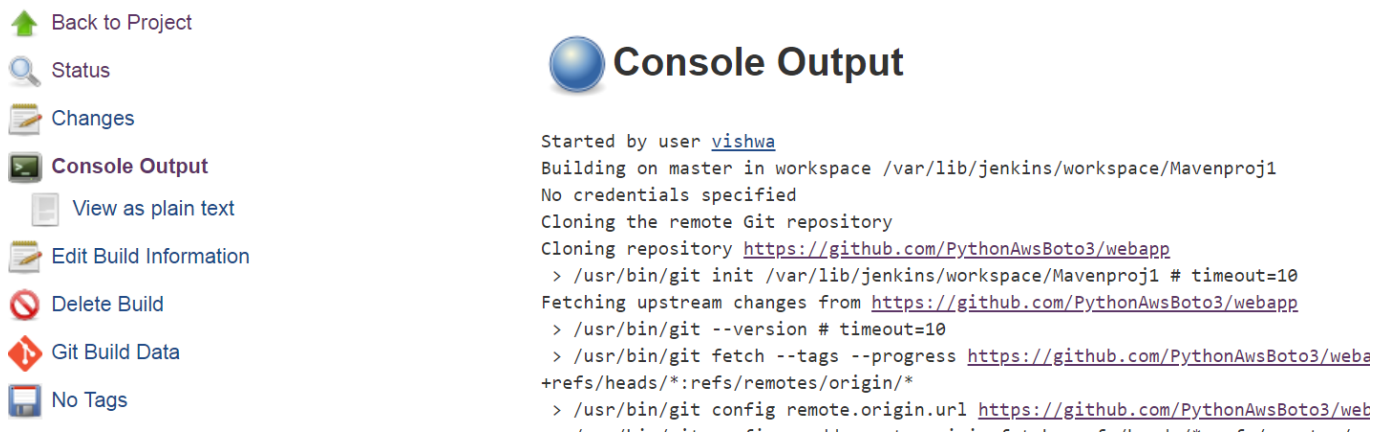
At this stage , we can “save” the project and build the JOB and check if the status is Success. Like below output.



Click on the Build after Clicking on ‘Build NOW”.



Click on “Console Output”



This is the Build output Starting part.

```
Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO]
[INFO] --- maven-war-plugin:2.4:war (default-war) @ mvn-hello-world ---
[INFO] Packaging webapp
[INFO] Assembling webapp [mvn-hello-world] in [/var/lib/jenkins/workspace/Mavenproj1/target/mvn-hello-wor
[INFO] Processing war project
[INFO] Copying webapp resources [/var/lib/jenkins/workspace/Mavenproj1/src/main/webapp]
[INFO] Webapp assembled in [58 msecs]
[INFO] Building war: /var/lib/jenkins/workspace/Mavenproj1/target/mvn-hello-world.war
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 3.810 s
[INFO] Finished at: 2019-02-01T09:43:28-05:00
[INFO] Final Memory: 15M/149M
[INFO] ------------------------------------------------------------------------
Finished: SUCCESS
```

This is the Build Output Ending part.

This shows that the Maven code is fine and the Build of the Code is Successful

But the objective of the Jenkins job is to do “AutoDeployment”.

So, Let’s Continue……..

Step 4:

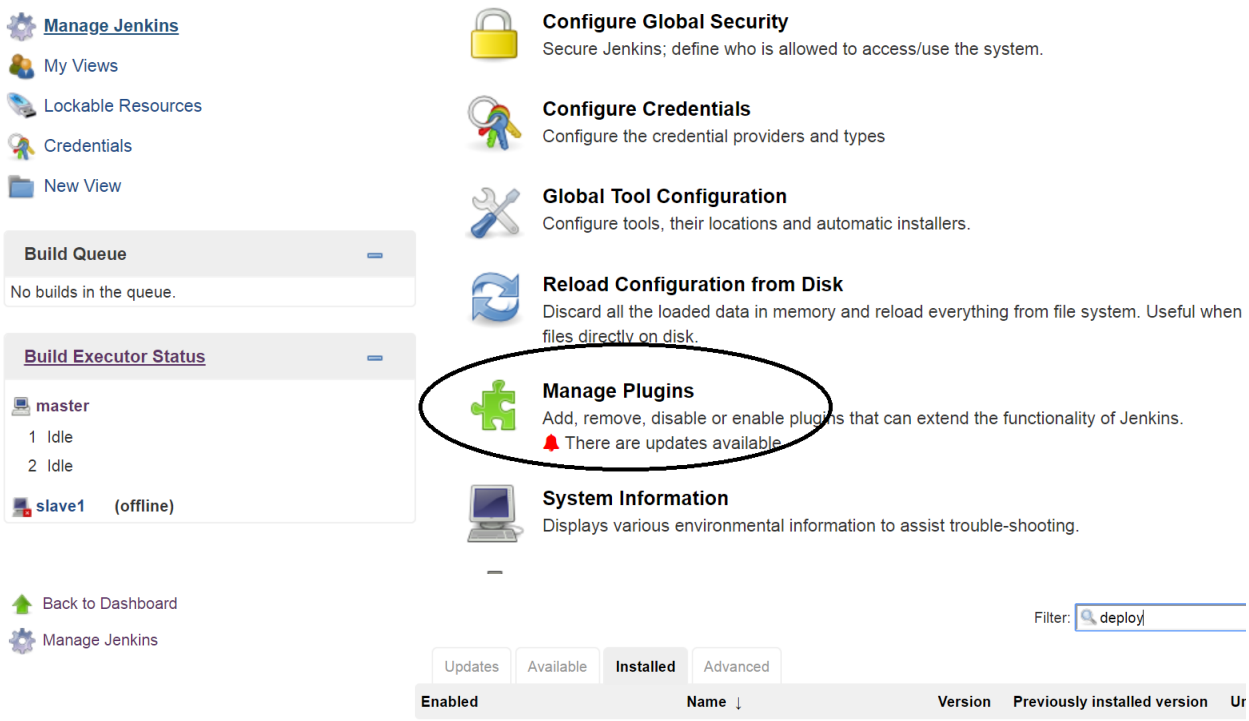
Install the Tomcat on the Jenkins server for the Auto Deployment.

Refer to the Document for **“Tomcat 8 Installation and Configuration on Centos 7”**

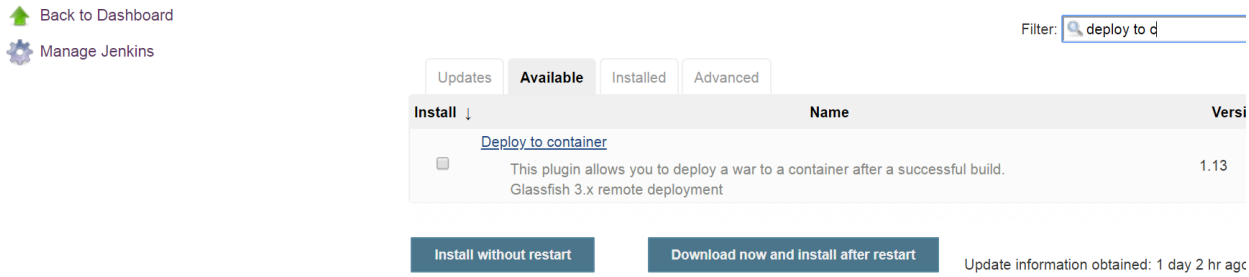
Step 5:

Now we need to deploy the Application (the war file) on tomcat server.

First, we need to install a plugin for "**Deploy to Container**" Plugin.



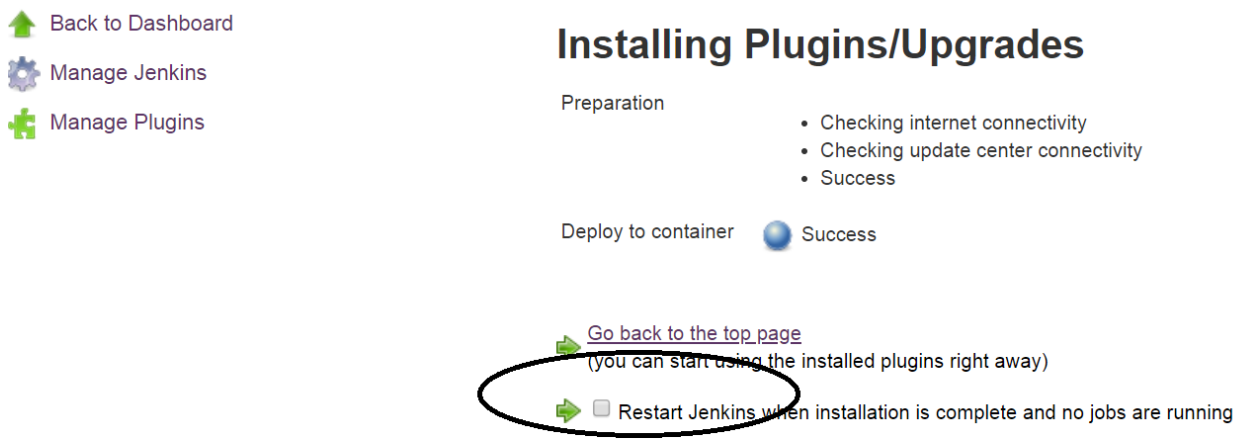
In the “installed” plugin there is no Deploy plugin.



The “Delopy to container” plugin to test the war file is under ‘Available”

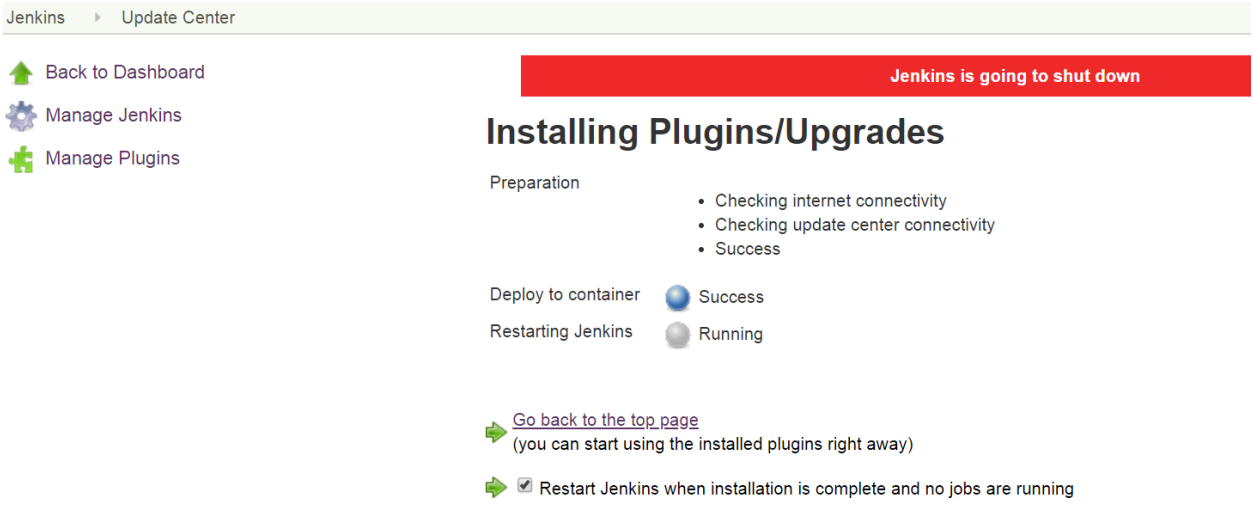
Means it needs to be installed on the jenkins first.

Click on “**install without restart**”.



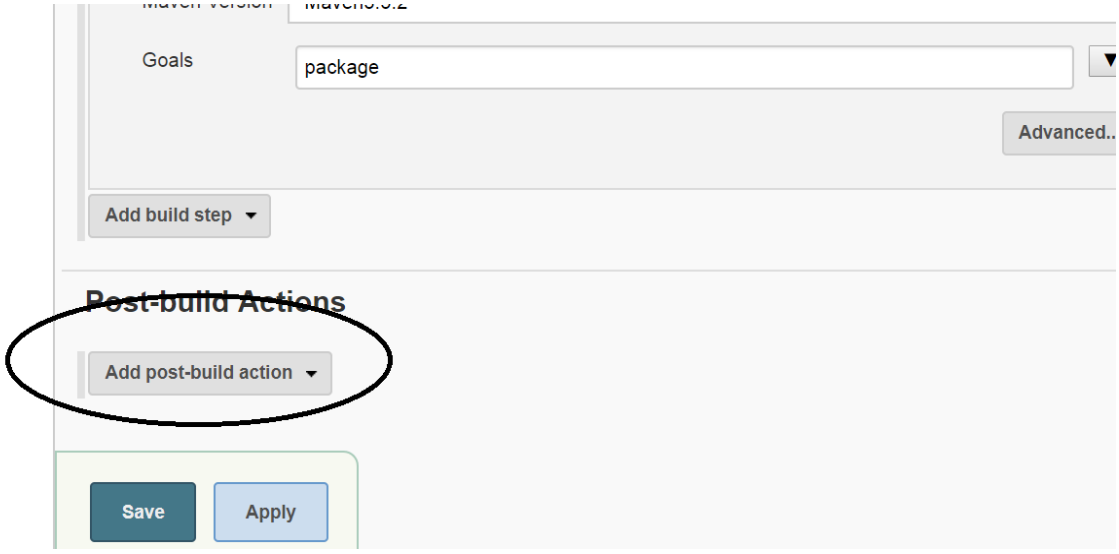
Once the installation is Success, Put a tick mark on “Restart Jenkins”.

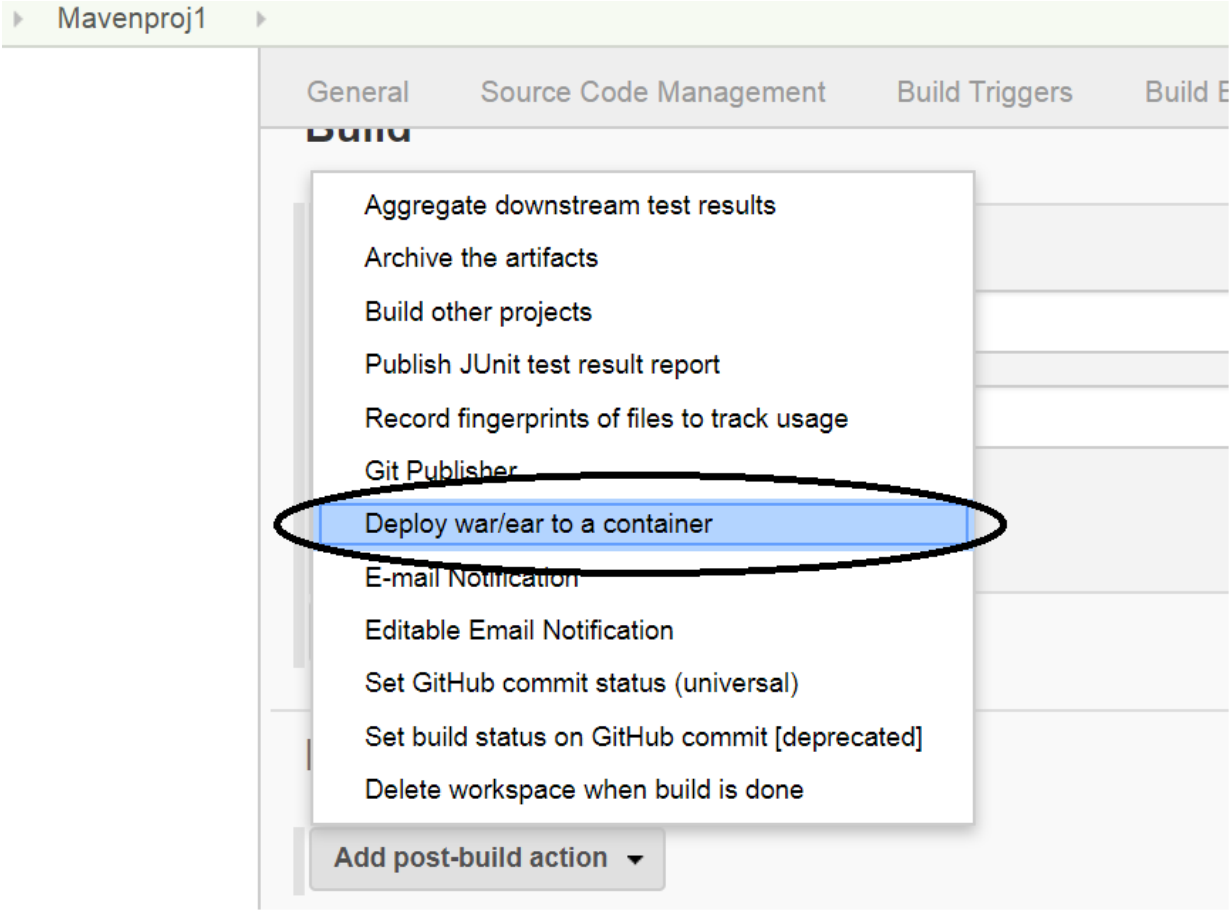
This would restart the Jenkins and Activate the Plugin that is installed.



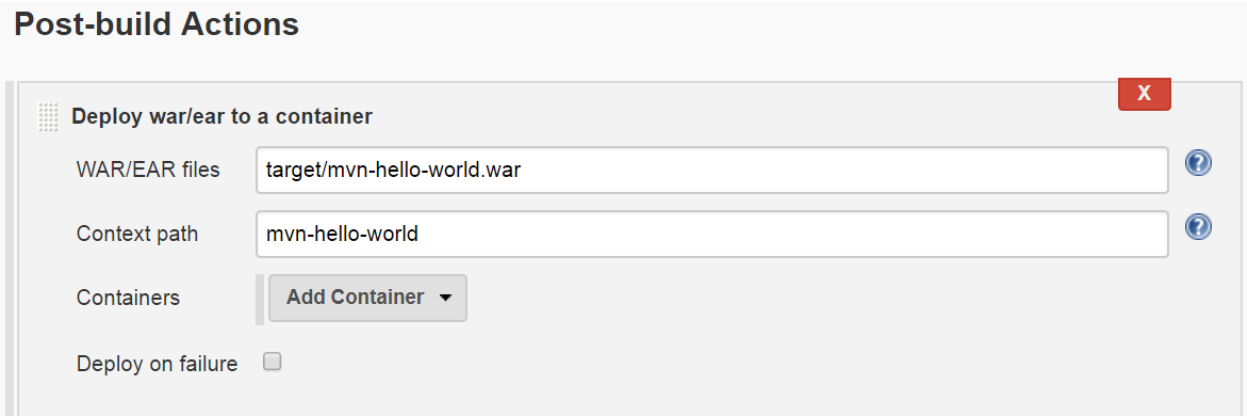
Wait for 3 to 4 min and refresh the page to go back to the login page of Jenkins.

Now, lets edit the project that we had created and go the “Post build” section on it.

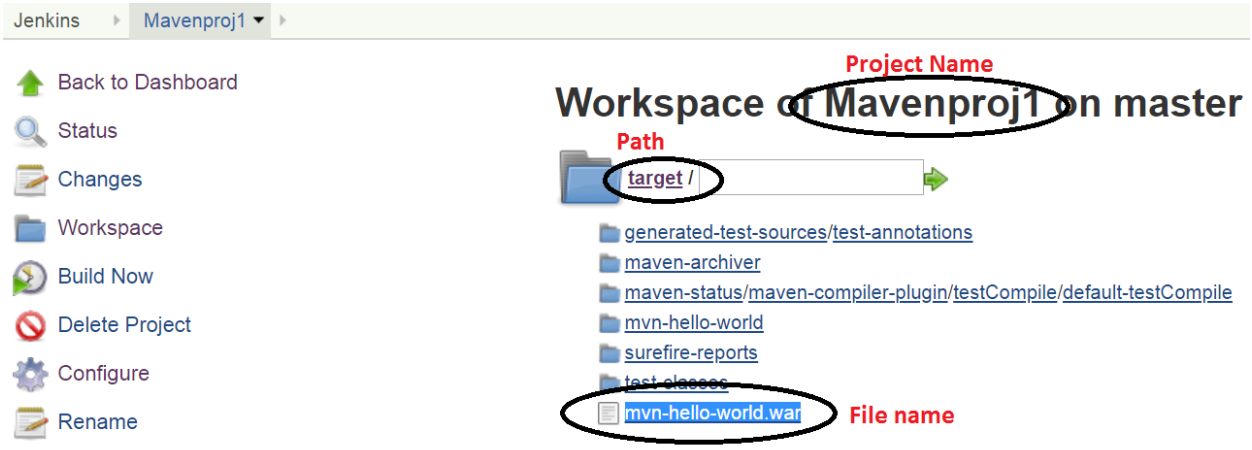




Select “Deploy war to a container”.



The WAR file path can be found from the workspace of the project as below.

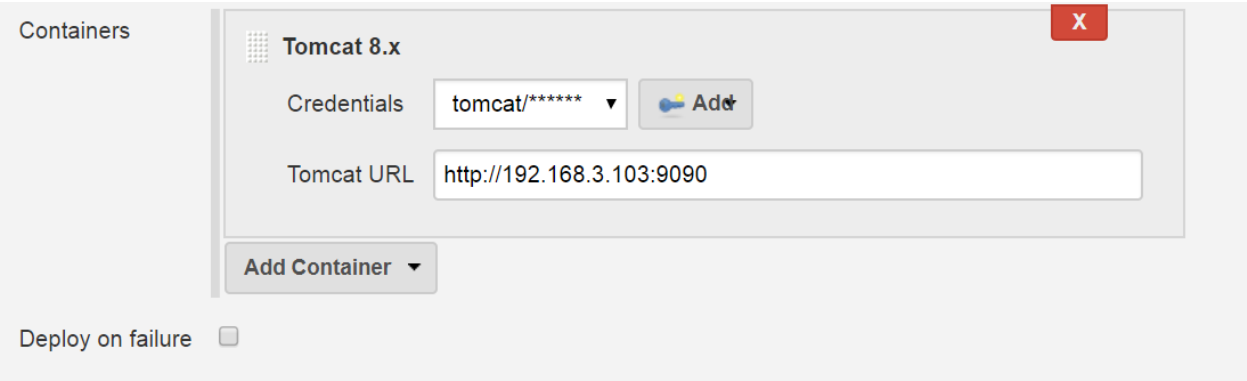


We don't need to mention the **Project Folder Name** as the **default PATH** for the WAR files will start from the **project folder itself.**

I HOPE THERE IS NO CONFUSION. ☺

Now, lets select the 'Container" to be used to Deploy the WAR file and test it.

Here the Container would be Tomcat 7.0, So make sure to install the Tomcat on the Jenkins Master as per the step 3 with username and Pwd.



The Tomcat should be up and running on the VM. The Username and password is what was created for the tomcat. (tomcat/Reset123).

Also, give full access to the “webapps” folder on the tomcat server.

```
drwxrwxrwx. 8 tomcat tomcat   131 Feb  1 13:30 webapps
drwxr-x---. 3 tomcat tomcat    22 Feb  1 11:23 work
[root@localhost tomcat]# chmod 777 webapps
[root@localhost tomcat]#
```

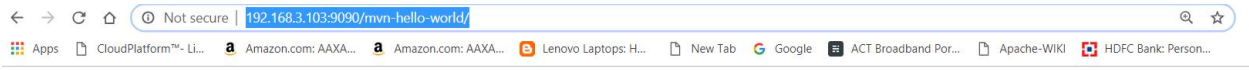
$ chmod 777 webapps

Step 5: Output

```
[INFO] Building war: /var/lib/jenkins/workspace/Mavenproj1/target/mvn-hello-world.war
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 4.894 s
[INFO] Finished at: 2019-02-01T13:30:03-05:00
[INFO] Final Memory: 11M/138M
[INFO] ------------------------------------------------------------------------
Deploying /var/lib/jenkins/workspace/Mavenproj1/target/mvn-hello-world.war to container Tomcat 8.x Remote
with context mvn-hello-world
  [/var/lib/jenkins/workspace/Mavenproj1/target/mvn-hello-world.war] is not deployed. Doing a fresh
deployment.
  Deploying [/var/lib/jenkins/workspace/Mavenproj1/target/mvn-hello-world.war]
Finished: SUCCESS
```

Now try to access the page with

http://192.168.3.103:9090/mvn-hello-world/



Welcome to

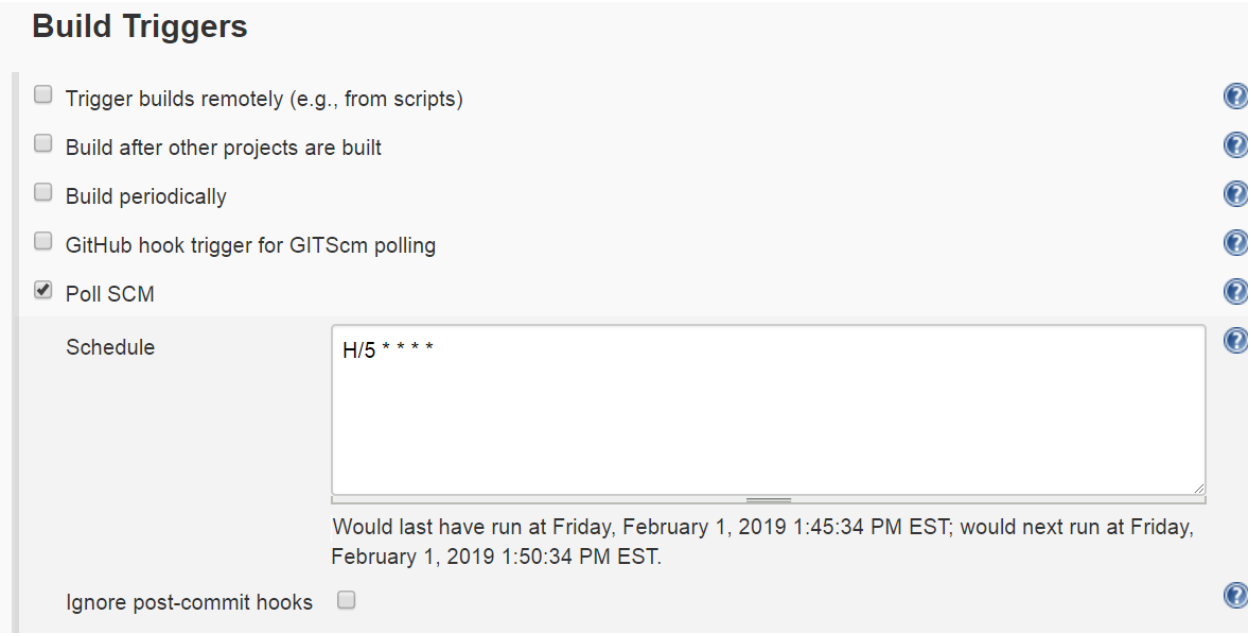
Jenkins

This is a Project on Maven

WELL DONE

Step 6: Update the GITHUB and it would trigger the build automatically

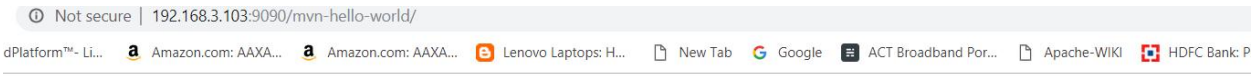
Change the JOB to automatically detect the changes on the GITHUB.



This will keep monitoring the GITHUB every 5 min and will trigger the build if any changes got on it.

After Changing the code on the GITHUB.

And then the build trigers with below output.



Welcome to

Jenkins

This is a Project on Maven

WELL DONE - It is updated NOW