

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**

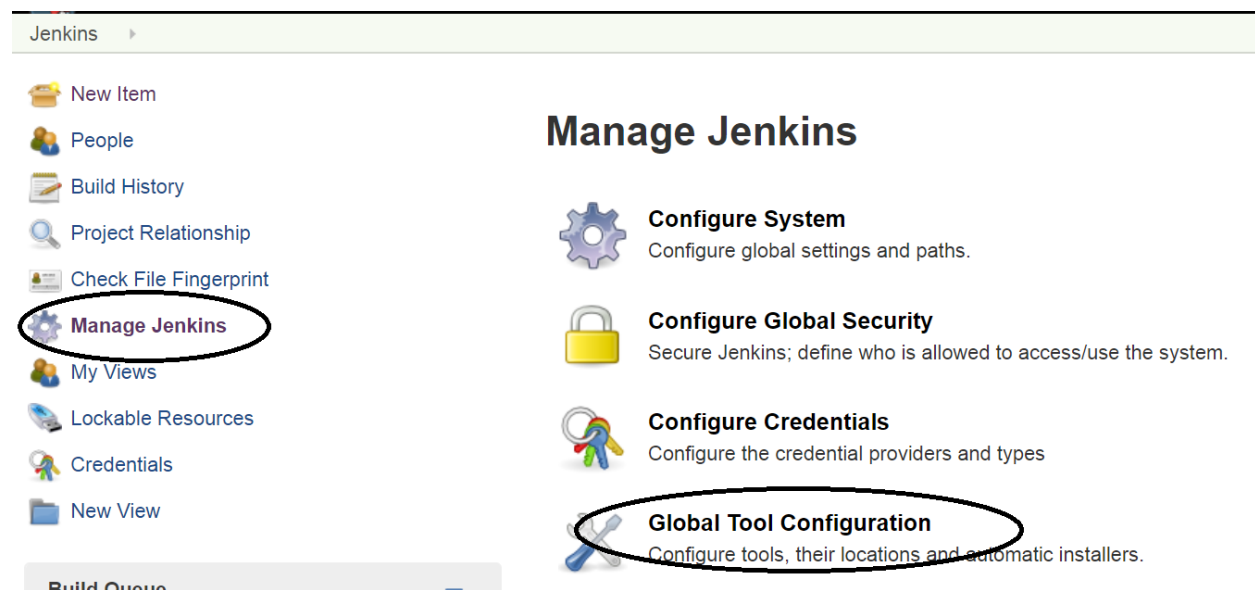
We would be running this Maven project on the Slave.

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

JDK

JDK installations

Add JDK



JDK

Name

JDK1.8

JAVA_HOME

/usr/lib/jvm/java-1.8.0-openjdk



Install automatically



Delete JDK

Add JDK

List of JDK installations on this system

Git

Note:- You may get a warning that the path is invalid. Since we have configured JDK on the Slave, the Jenkins is checking 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.

Git

Git installations



Git

Name

Default

Path to Git executable

/usr/bin/git



Install automatically



Add Installer

Delete Git

Maven

Maven installations

Add Maven

Maven

Name

☒ Install automatically [?](#)

Install from Apache

Version

Delete Installer

Add Installer

Delete Maven

Add Maven

List of Maven installations on this system

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

version

Add Installer

Add Maven

List of Maven installations on this system

Docker

Docker installations

Add Docker

List of Docker installations on this system

Save

Apply

Step 3:

Create Maven Project

Mavenproj1 ▶

General

Source Code Management

Build Triggers

Build Environment

Build

Post-build Actions

Description

Auto Deployment

[Plain text] [Preview](#)

☐ Discard old builds

☐ GitHub project

☐ This build requires lockable resources

☐ This project is parameterized

☐ Throttle builds

☐ Disable this project

☐ Execute concurrent builds if necessary

☐ Restrict where this project can be run

?

?

?

?

?

?

Jenkins – Maven -Jenkins Job

☐ None

☒ Git

Repositories

Repository URL

Credentials

Branches to build

Branch Specifier (blank for 'any')

Repository browser

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.

Vishwanathms / SampleMaven

<> Code Issues 0 Pull requests 0 Projects 0 Wiki Insights

Sample Maven Code

[Manage topics](#)

8 commits 1 branch 0 releases

Branch: master New pull request Create

Vishwanathms Update index.jsp	
src	Update index.jsp
.gitignore	Add files via upload
.travis.yml	Add files via upload
LICENSE	Add files via upload
README.md	Add files via upload
pom.xml	Add files via upload

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

Build

Invoke top-level Maven targets

Maven Version Maven3.5.2

Goals package

Advanced...

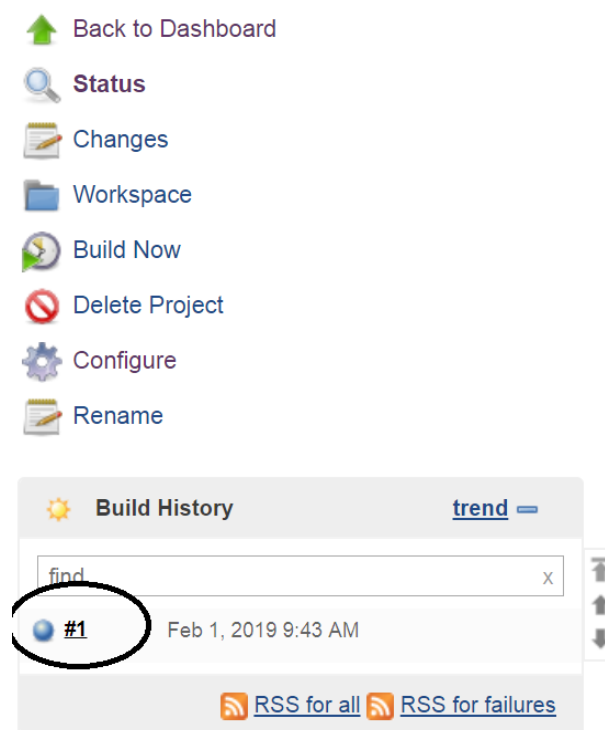
Add build step

Jenkins – Maven -Jenkins Job

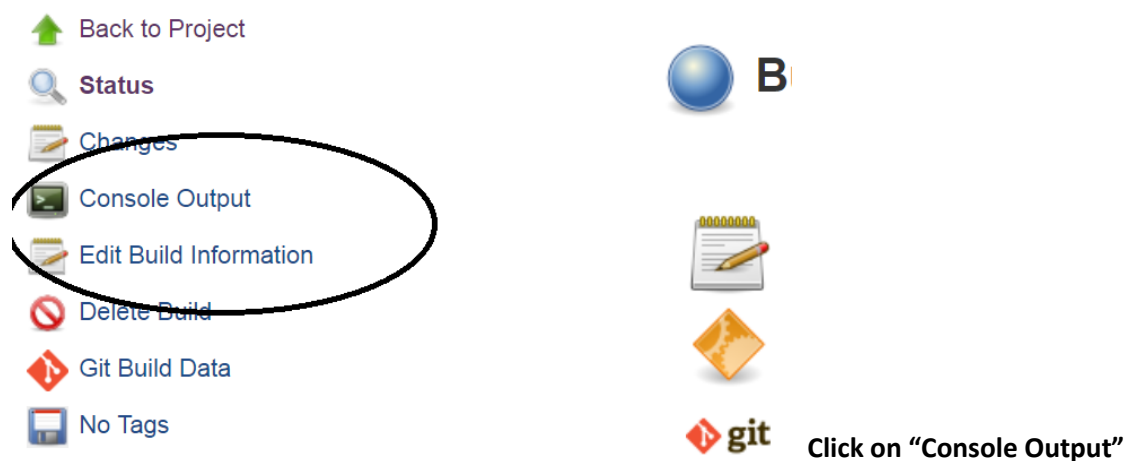
This Build would create an Maven .war file which would be an package file for the source code from GIT.

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’.



 [Back to Project](#)

 [Status](#)

 [Changes](#)


 **Console Output**

 [View as plain text](#)

 [Edit Build Information](#)

 [Delete Build](#)

 [Git Build Data](#)

 [No Tags](#)

Console Output

```
Started by user vishwa
Building on master in workspace /var/lib/jenkins/workspace/Mavenproj1
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/PythonAwsBoto3/webapp
> /usr/bin/git init /var/lib/jenkins/workspace/Mavenproj1 # timeout=10
Fetching upstream changes from https://github.com/PythonAwsBoto3/webapp
> /usr/bin/git --version # timeout=10
> /usr/bin/git fetch --tags --progress https://github.com/PythonAwsBoto3/webapp
+refs/heads/*:refs/remotes/origin/*
> /usr/bin/git config remote.origin.url https://github.com/PythonAwsBoto3/webapp
```

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-world]
[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:

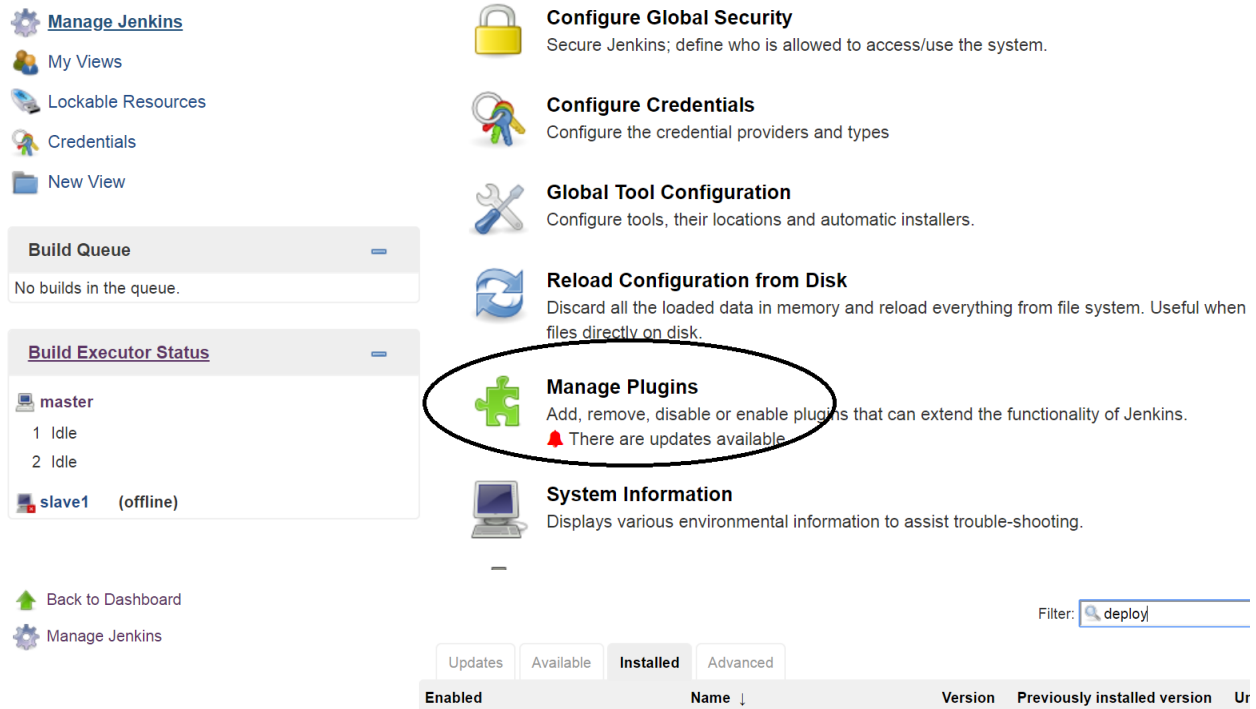
Install the Tomcat on the Jenkins server for the Auto Deployment.

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.



[Manage Jenkins](#)

[My Views](#)

[Lockable Resources](#)

[Credentials](#)

[New View](#)

Build Queue

No builds in the queue.

Build Executor Status

master

- 1 Idle
- 2 Idle

slave1 (offline)

[Back to Dashboard](#)

[Manage Jenkins](#)

Configure Global Security
Secure Jenkins; define who is allowed to access/use the system.

Configure Credentials
Configure the credential providers and types

Global Tool Configuration
Configure tools, their locations and automatic installers.

Reload Configuration from Disk
Discard all the loaded data in memory and reload everything from file system. Useful when files directly on disk.

Manage Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
There are updates available

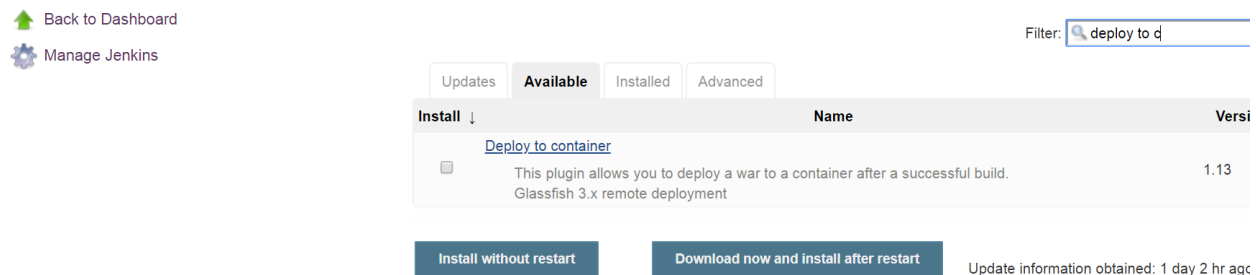
System Information
Displays various environmental information to assist trouble-shooting.

Filter:

Updates Available **Installed** Advanced

Enabled	Name ↓	Version	Previously installed version	Ur
---------	--------	---------	------------------------------	----

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



[Back to Dashboard](#)

[Manage Jenkins](#)

Filter:

Updates **Available** Installed Advanced

Install ↓	Name	Versi
Deploy to container		
<input type="checkbox"/>	This plugin allows you to deploy a war to a container after a successful build. Glassfish 3.x remote deployment	1.13


[Install without restart](#) [Download now and install after restart](#)

Update information obtained: 1 day 2 hr ago

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**”.

 [Back to Dashboard](#)


 [Manage Jenkins](#)

 [Manage Plugins](#)

Installing Plugins/Upgrades


Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

Deploy to container  Success

 [Go back to the top page](#)

(you can start using the installed plugins right away)


 ☐ Restart Jenkins when installation is complete and no jobs are running


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

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

Jenkins [Update Center](#)

 [Back to Dashboard](#)

 [Manage Jenkins](#)


 [Manage Plugins](#)

Jenkins is going to shut down


Installing Plugins/Upgrades

Preparation


- Checking internet connectivity
- Checking update center connectivity
- Success

Deploy to container  Success

Restarting Jenkins  Running

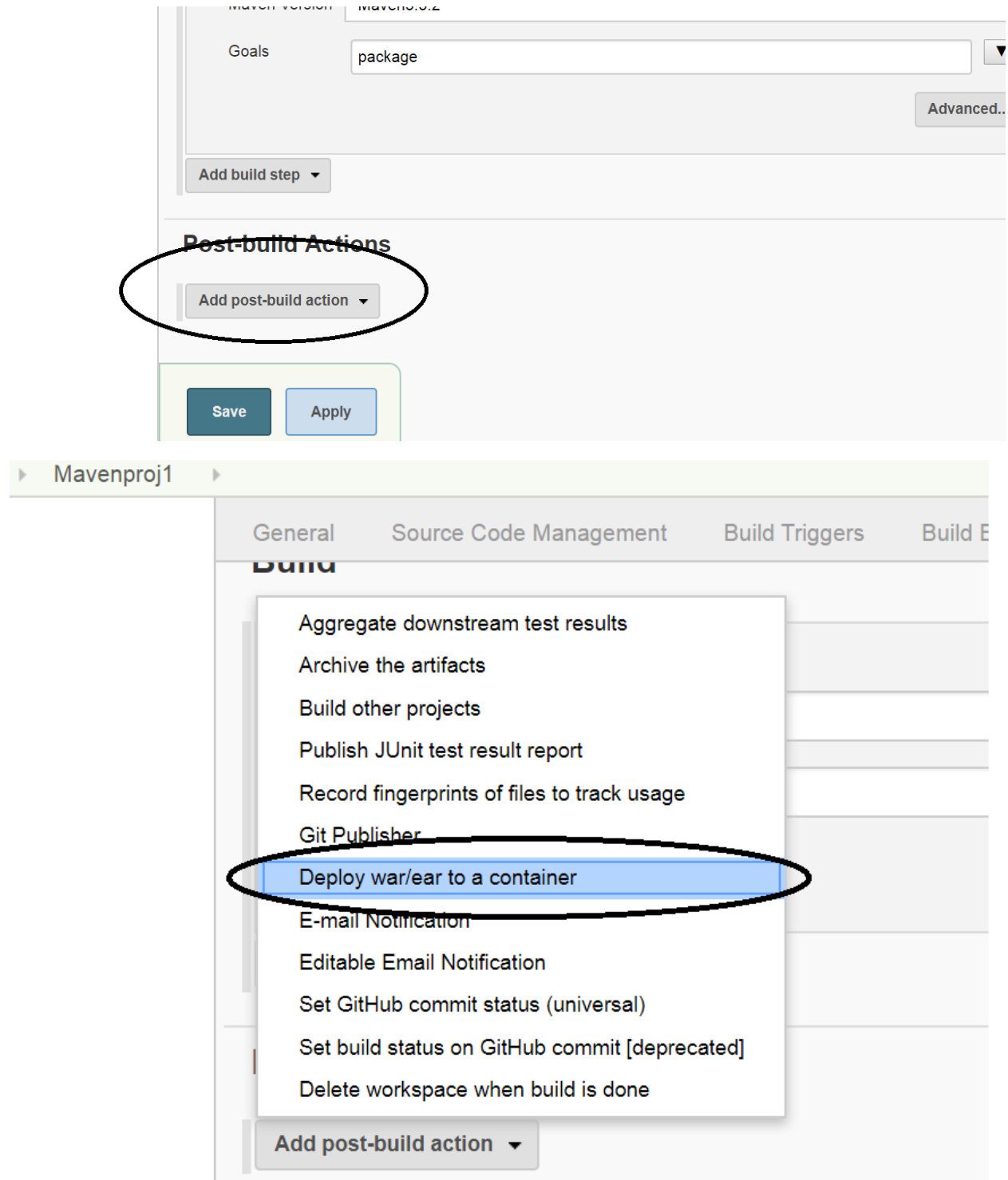
 [Go back to the top page](#)

(you can start using the installed plugins right away)

 ☒ Restart Jenkins when installation is complete and no jobs are running

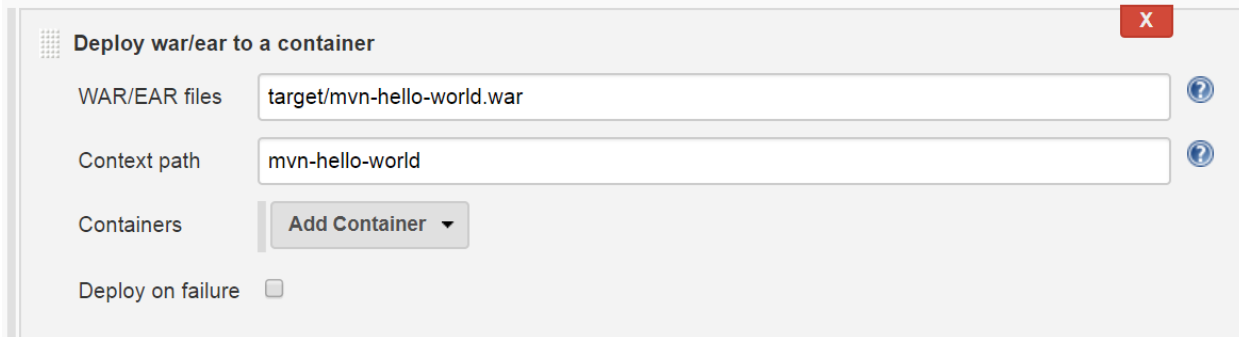
Wait for 3 to 4 min and refresh the page to go back to the login page of Jenkins.

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



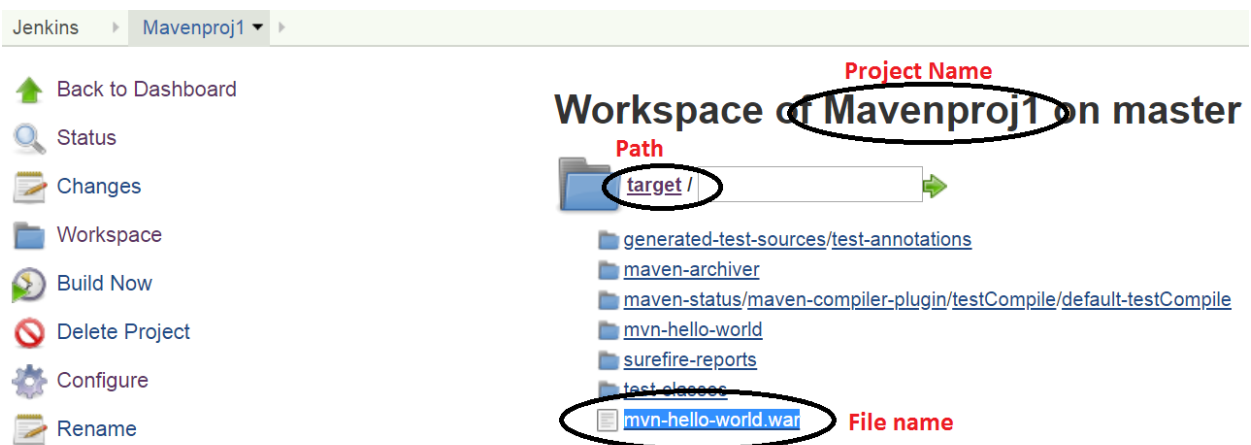
Select “Deploy war to a container”.

Post-build Actions



The screenshot shows the 'Post-build Actions' section of a Jenkins job configuration. Under the 'Deploy war/ear to a container' action, the 'WAR/EAR files' field contains 'target/mvn-hello-world.war' and the 'Context path' field contains 'mvn-hello-world'. There is an 'Add Container' button and a 'Deploy on failure' checkbox which is currently unchecked.

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

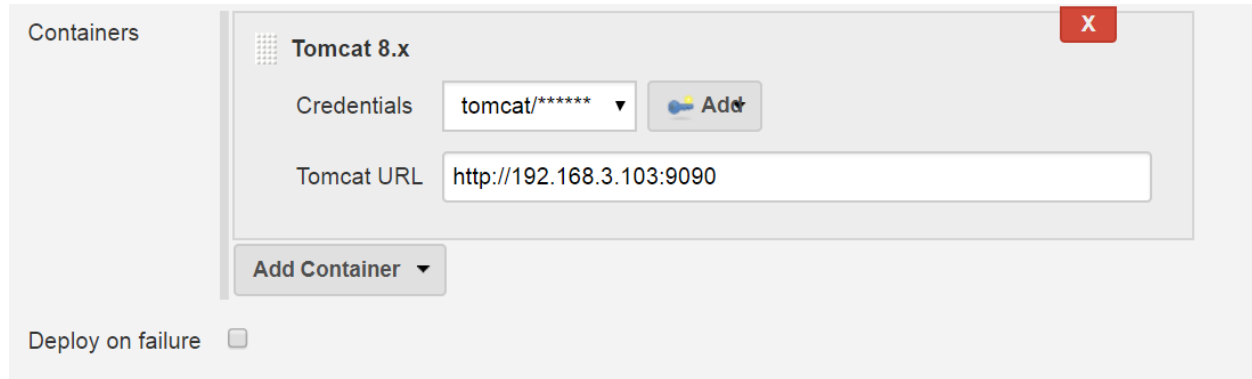


We don't need to mention the **Project Folder Name** as the **default PATH** for the WAR files will start from the **project folder itself**.

I HOPE THERE IS NO CONFUSION. 😊

Now, let's 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 screenshot shows the Jenkins configuration page for a Tomcat 8.x container. On the left, a sidebar labeled 'Containers' is visible. The main area is titled 'Tomcat 8.x' and contains a 'Credentials' dropdown menu set to 'tomcat/*****' with an 'Add' button next to it. Below this is a 'Tomcat URL' text input field containing 'http://192.168.3.103:9090'. At the bottom of the configuration area is an 'Add Container' button. Below the configuration area, there is a checkbox labeled 'Deploy on failure' which is currently unchecked.

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]#
```

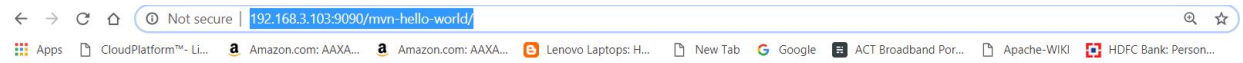
\$ chmod 777 webapps

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.

Build Triggers

- ☐ Trigger builds remotely (e.g., from scripts)
- ☐ Build after other projects are built
- ☐ Build periodically
- ☐ GitHub hook trigger for GITScm polling
- ☒ Poll SCM

Schedule:

Would last have run at Friday, February 1, 2019 1:45:34 PM EST; would next run at Friday, February 1, 2019 1:50:34 PM EST.

Ignore post-commit hooks ☐

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 triggers with below output.

Not secure | 192.168.3.103:9090/mvn-hello-world/
dPlatform™- Li... Amazon.com: AAXA... Amazon.com: AAXA... Lenovo Laptops: H... New Tab Google ACT Broadband Por... Apache-WIKI HDFC Bank: P

Welcome to



Jenkins

This is a Project on Maven

WELL DONE - It is updated NOW