**Jenkins Set-up**

1. Install Java 8 and Java Environment

2. Install Jenkins

3. Install Maven and Maven Environment

4. Configure Jenkins

5. Install and Configure Apache Tomcat

6. Create & Clone Git Repository (In Git-Hub document)

7. Jenkins Standalone Mode

Project 1: Build MVN Artifact & Deploy MVN Artifact on the Tomcat Container

8. Jenkins Master/Slave Setup

Project 1: Build Project in the Salve Mode

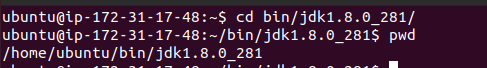
Project 2: Build MVN Artifact in the Salve Node & Deploy MVN Artifact in the Master Node

**Install Java 8**

1. Launch an EC2 Instance and open ports 22/8080/8081
2. Login to Oracle site and either download or wget link e.g. jdk-8u281-linux-x64.tar.gz for the Linux Ubuntu, tar it tar –zxvf jdk-8u281-linux-x64.tar.gz
3. Java Home: java –version will through error because we need to set Java Environment variable to set the PATH as Global in the Bash Profile.

The **Bash profile** is a file that runs every time a new **Bash** session is created. This is useful because we may need to run certain code every time system starts. OS doesn't include a **Bash profile** by default.

Bash **profile** file is located at /home/<user>/. bash\_profile

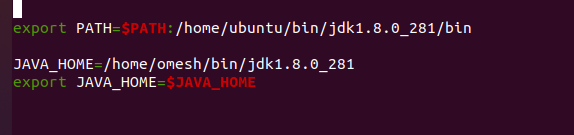


ubuntu@ip-172-31-17-48:~$ sudo nano ~/.bash\_profile

export PATH=$PATH: /home/ubuntu/bin/jdk1.8.0\_281/bin

JAVA\_HOME=/home/omesh/bin/jdk1.8.0\_281

export JAVA\_HOME=$JAVA\_HOME



Note: /bin is added as executables are here and also JAVA Home environment set.

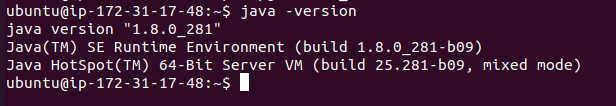
ubuntu@ip-172-31-17-48:~$ source ~/.bash\_profile

source ~/.bash\_profile

ubuntu@ip-172-31-17-48:~/bin/jdk1.8.0\_281$ echo $PATH

/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/home/ubuntu/bin/jdk1.8.0\_281/bin

Java version running successfully suggests that Java is installed.



**Install Jenkins**

Use official Jenkins site Jenkins.io and select debian platform, follow instructions listed based on the OS type.



1.

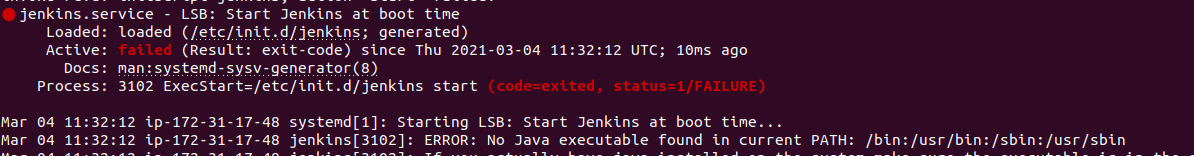


2. ubuntu@ip-172-31-17-48:~$ sudo nano /etc/apt/sources.list

3. ubuntu@ip-172-31-17-48:~$ sudo apt update

ubuntu@ip-172-31-17-48:~$ sudo apt install jenkins –y

Jenkins will fail to start



Jenkins will give java path error so we need to provide the correct path by editing the file /etc/init.d/Jenkins and the correct is provide by the echo $PATH.

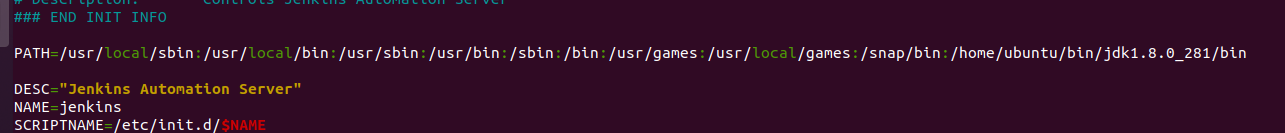
ubuntu@ip-172-31-17-48:~$ echo $PATH

/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/home/ubuntu/bin/jdk1.8.0\_281/bin

ubuntu@ip-172-31-17-48:~$ sudo nano /etc/init.d/Jenkins

Change the PATH to new path found in echo command.

PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/home/ubuntu/bin/jdk1.8.0\_281/bin

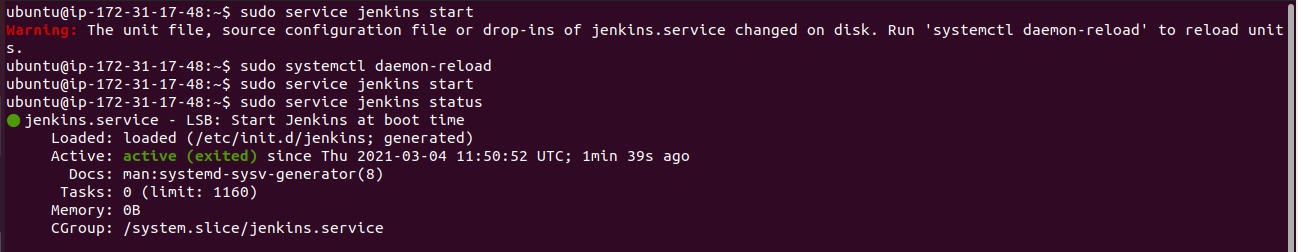


ubuntu@ip-172-31-17-48:~$ sudo service jenkins start

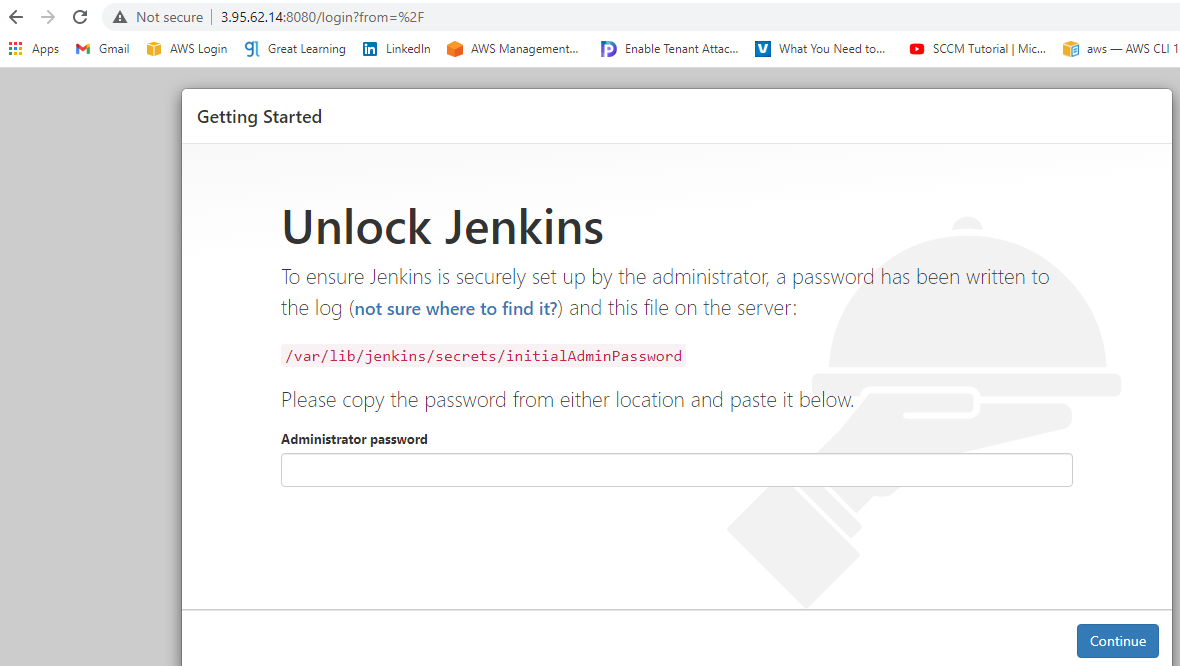
ubuntu@ip-172-31-17-48:~$ sudo systemctl daemon-reload

ubuntu@ip-172-31-17-48:~$ sudo service jenkins start

ubuntu@ip-172-31-17-48:~$ sudo service jenkins status



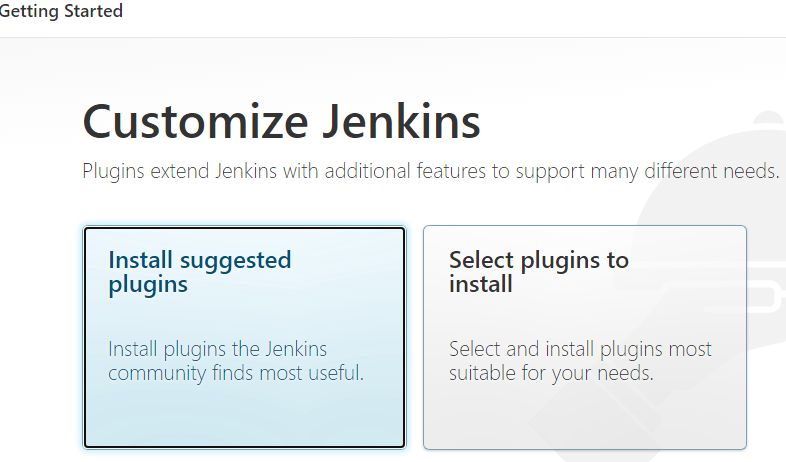
Login to Jenkins using Publicip:8080

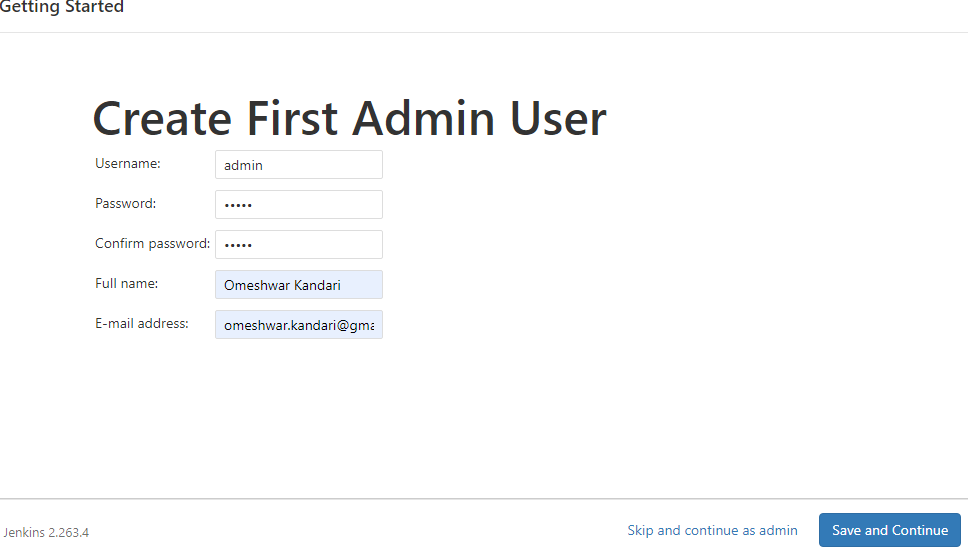


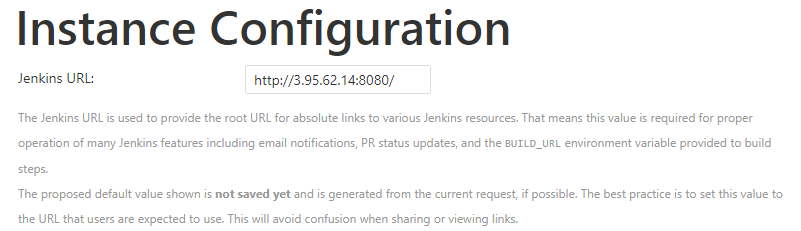
Copy the initial password from the file, paste and continue

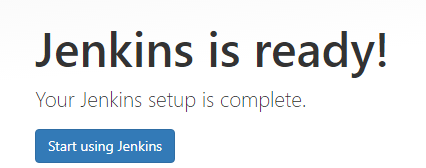
ubuntu@ip-172-31-17-48:~$ sudo nano /var/lib/jenkins/secrets/initialAdminPassword

Install the plugin and create the First Admin user









**Install Maven**

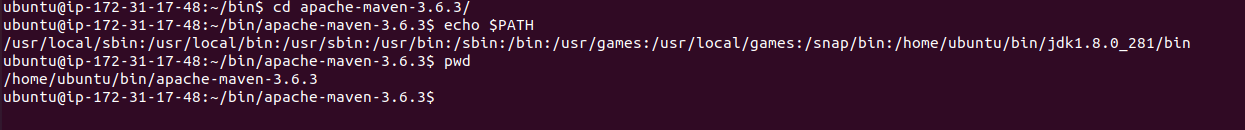
1. Copy the download the link for binary tar.gz from Apache Maven site

e.g. <https://downloads.apache.org/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz>

2. Install Maven: ubuntu@ip-172-31-17-48:~/bin$ wget <https://downloads.apache.org/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz>

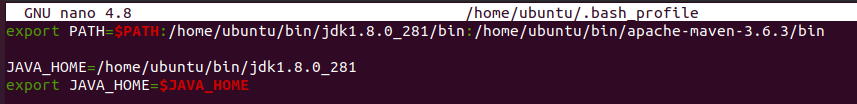
ubuntu@ip-172-31-17-48:~/bin$ tar -zxvf apache-maven-3.6.3-bin.tar.gz

(tar will extract the executables)

3. Set the MVN Path 

Add maven path /home/ubuntu/bin/apache-maven-3.6.3/bin in Java path with : and this way we can add anyother environment as needed.

ubuntu@ip-172-31-17-48:~/bin/apache-maven-3.6.3$ sudo nano ~/.bash\_profile



ubuntu@ip-172-31-17-48:~/bin/apache-maven-3.6.3$ source ~/.bash\_profile

ubuntu@ip-172-31-17-48:~/bin/apache-maven-3.6.3$ mvn -v

Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)

Maven home: /home/ubuntu/bin/apache-maven-3.6.3

Java version: 1.8.0\_281, vendor: Oracle Corporation, runtime: /home/ubuntu/bin/jdk1.8.0\_281/jre

Default locale: en, platform encoding: UTF-8

OS name: "linux", version: "5.4.0-1038-aws", arch: "amd64", family: "unix"

**Configure Jenkins**

Global Tool Configuration:

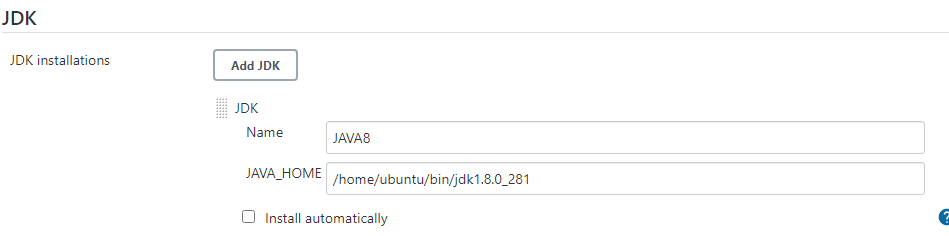
Installation packages needed to build the artifacts are set in here so that Build runs successfully and this configuration is also followed by the Slave Nodes which means the path for packages installed in Slave Nodes should be in line with this configuration. Since Git Repository is already configured so we will ser only Java and MVN for our MVN Program

JDK :

We have already installed Java so installation check box is unchecked and other variables like Name and Java\_Home path are entered based on the Java path in the bash\_profile.

ubuntu@ip-172-31-17-48:~/bin/apache-maven-3.6.3$ echo $JAVA\_HOME

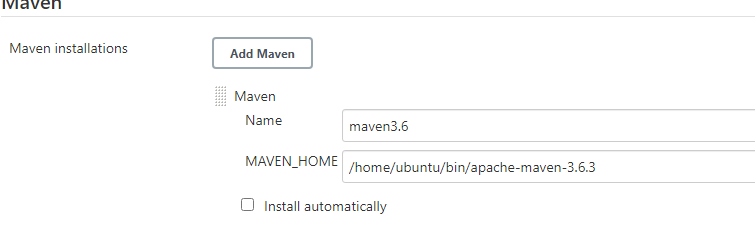
/home/ubuntu/bin/jdk1.8.0\_281



MAVEN:

MVN also installed so we need to update Name and MVN Path.

ubuntu@ip-172-31-17-48:~/bin/apache-maven-3.6.3$ mvn –version



**Install and configure Tomcat**

1. Install Tomcat using download link from Tomcat Apache tar gz

2. Port Change: By default Tomcat runs on 8080 and Jenkins already using it so we need to assign another port e.g. 8081 to Tomcat by editing Server.xml file

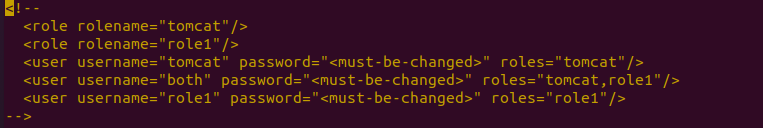
ubuntu@ip-172-31-17-48:~/bin/apache-tomcat-9.0.43/conf$ sudo nano server.xml



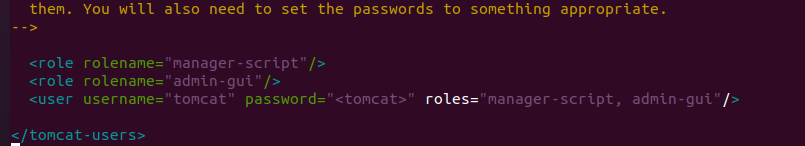
3. Permissions: Jenkins Server needs access to deploy artifacts into Tomcat container by editing the file tomcat-users.xml

ubuntu@ip-172-31-17-48:~/bin/apache-tomcat-9.0.43/conf$ sudo nano tomcat-users.xml

Uncomment and add user with role:



Add two roles and set the user with password:



1. Container Access Permission Error:

In case Deployment issue inside the container then a security permission issue might be the reason and we need to edit context.xml file.

ubuntu@ip-172-31-17-48:~/bin/apache-tomcat-9.0.43/webapps/manager/META-INF$ sudo nano context.xml

Below line needed to be commented because it means “only allow deployment from within the container using local-host but we want to deploy using Public IP”.

Default Settings:



Updated Settings:



1. Stop and start Tomcat to load the changes:

Use bin directory which has executable

ubuntu@ip-172-31-17-48:~/bin/apache-tomcat-9.0.43/bin$ ./shutdown.sh

ubuntu@ip-172-31-17-48:~/bin/apache-tomcat-9.0.43/bin$ ./startup.sh

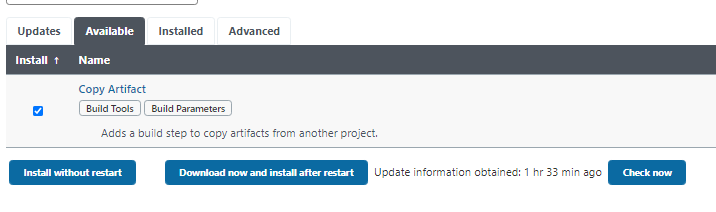
1. Check tomcat @ public ip:8081

**Project 1: Build MVN Artifact & Deploy MVN Artifact on the Tomcat Container**

1. Install Plug-in on the Jenkins:

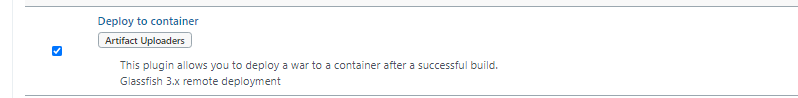
Copy Artifact:

It enables copy artifacts from the other projects during the Build Process.

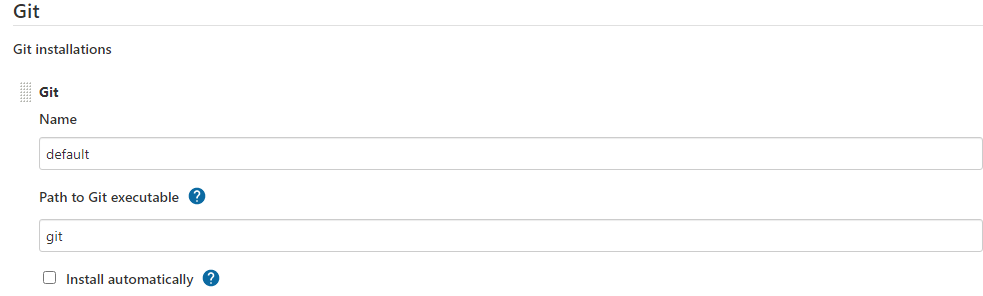


Deploy To Container:

It enables to deploy artifacts to another project during the Deploy Process.

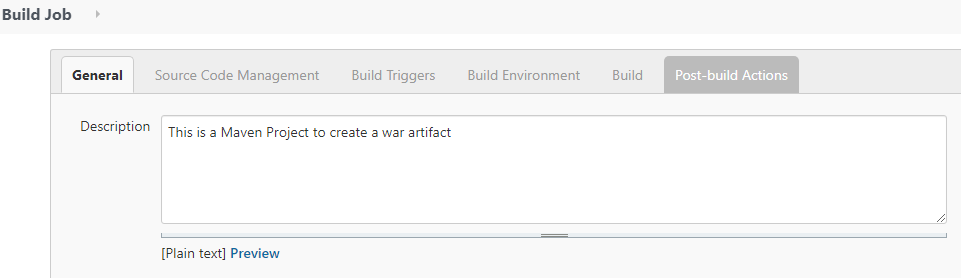


Github: Install github plug-in and in the global config below should be git settings.

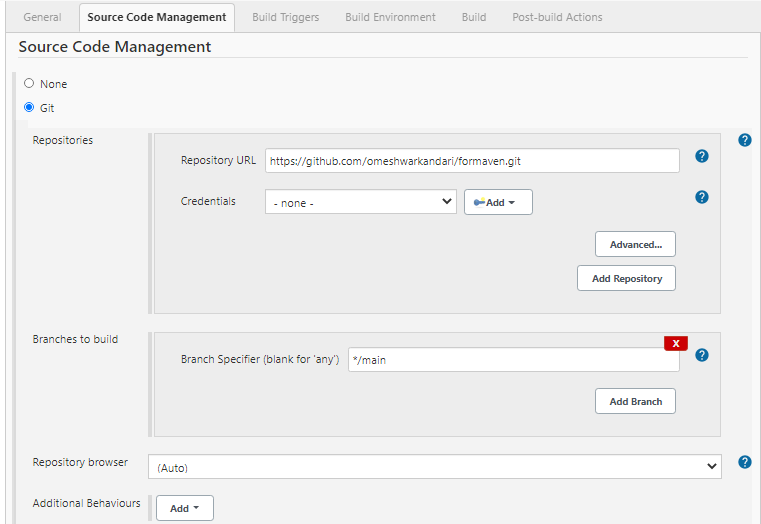


2. Create the Build Project:

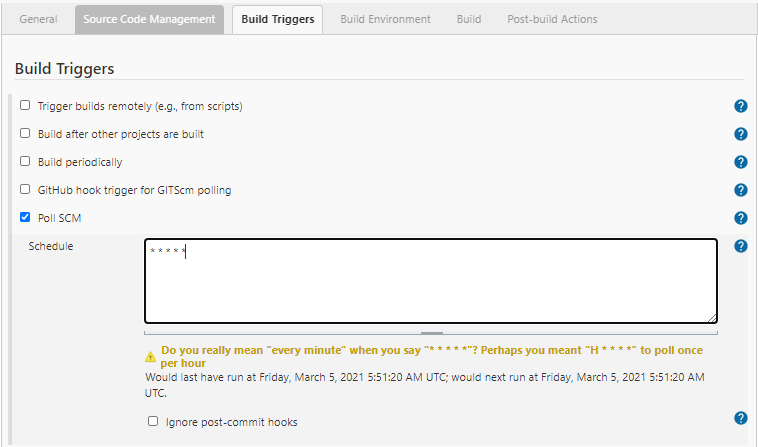
Create a new job Build MVN Artifact and describe the purpose.



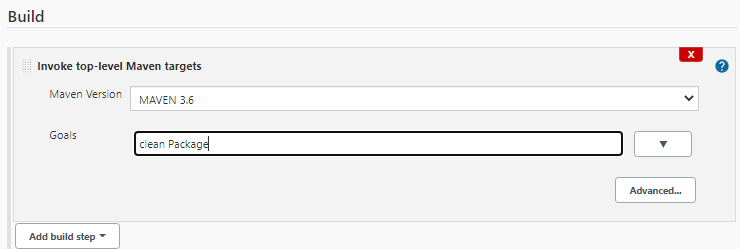
Source Code – Enter the path from Git Repo which is cloned on the local system



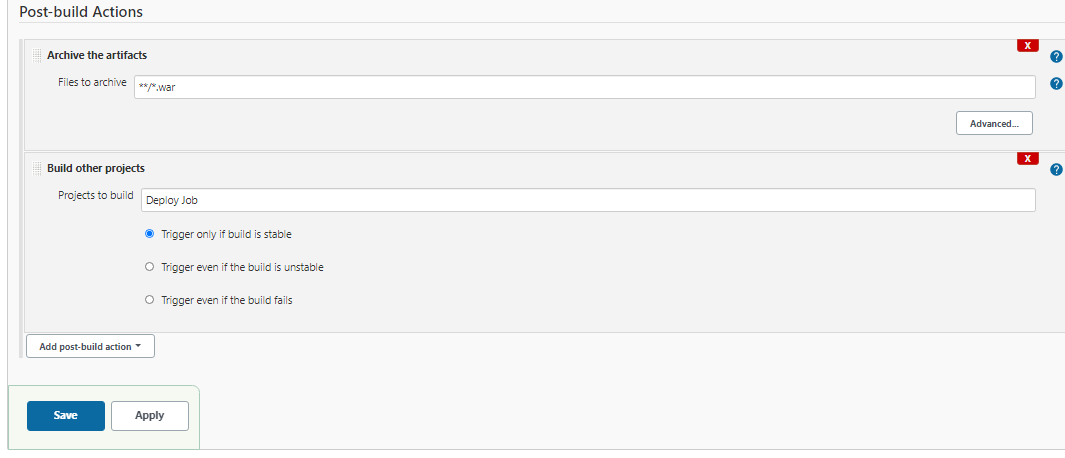
Build Trigger is set to Poll SCM (Source Code Manager i.e. Repo) with a crone job expression where 5 stars meaning polling will be every minute for new commit. Build Environment is not set for this test case.



Build: Invoke top-level Maven target which is Maven version 3.6 and it will clean the package before every new build.

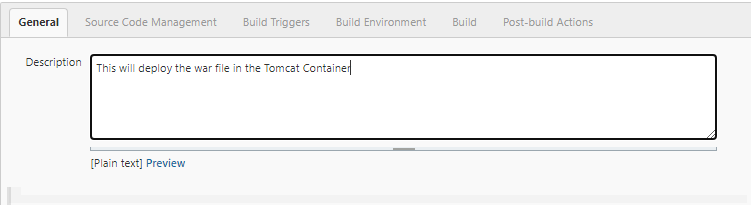


Post Build Actions: It will search every build with .war and archive it as a Build for other project which we need to mention e.g. Deploy Project

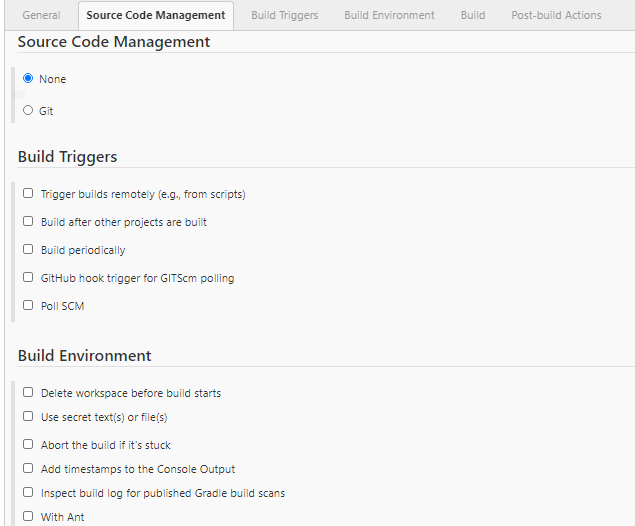


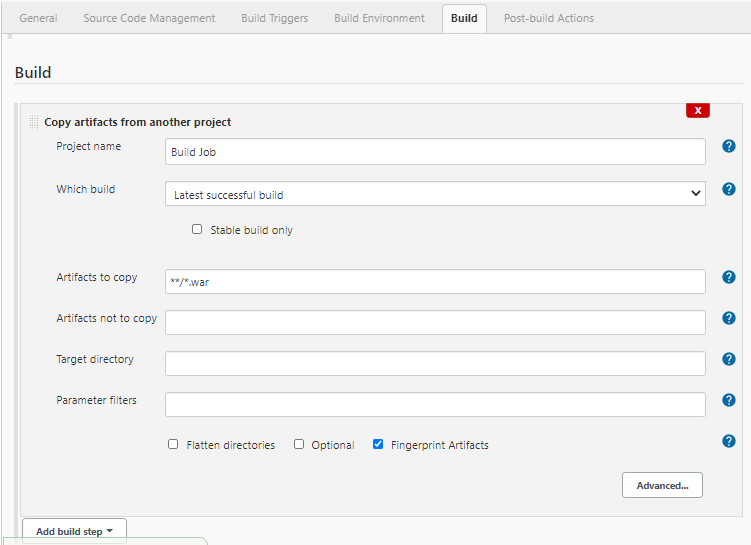
3. Create A Deploy Job: Deploy MVN Artifact on the Tomcat Container

New Item and create a Deploy Job in freestyle.

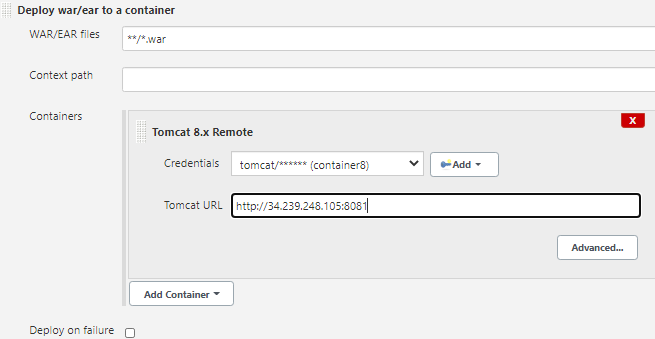


Source Code is None as Build will provide war file and no Build Trigger/ Build Environment selected/needed.

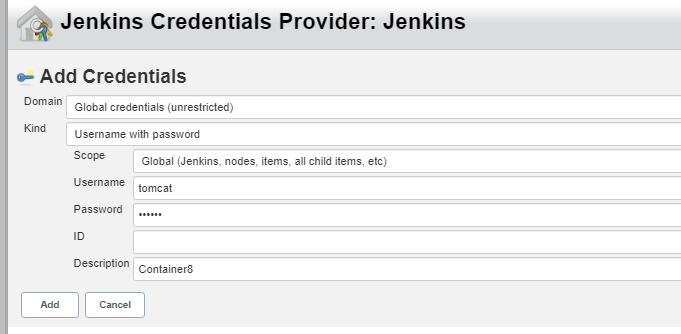


Build: Copy artifacts from another project 

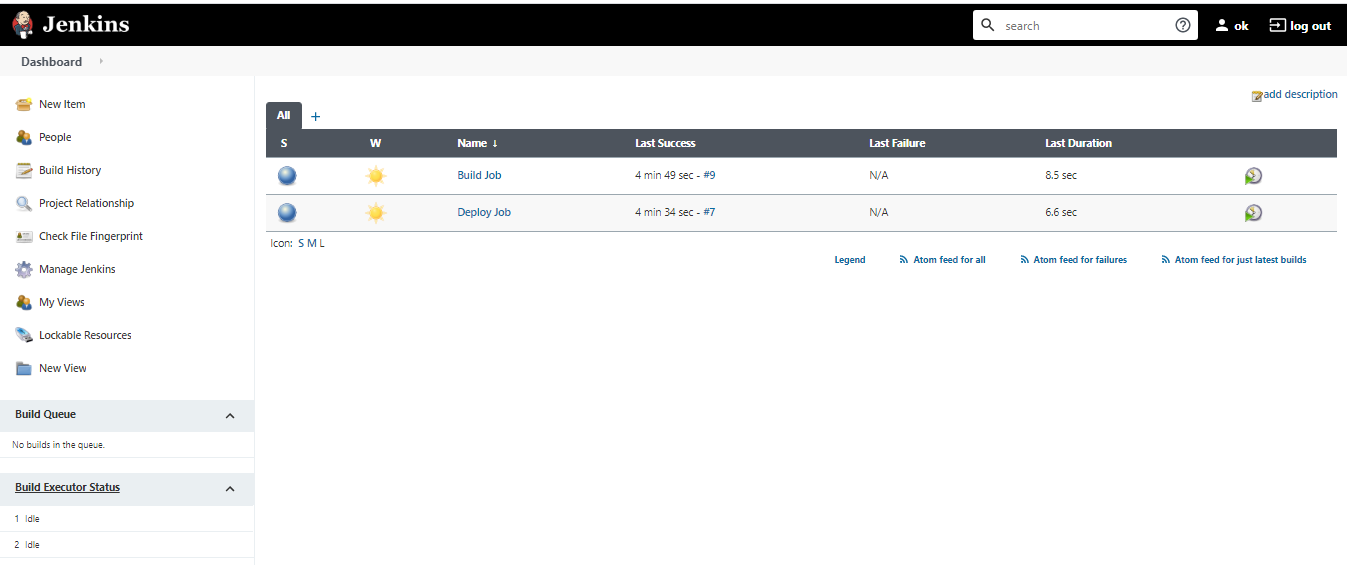
Post Build Actions: Deploy war to the tomcat container.



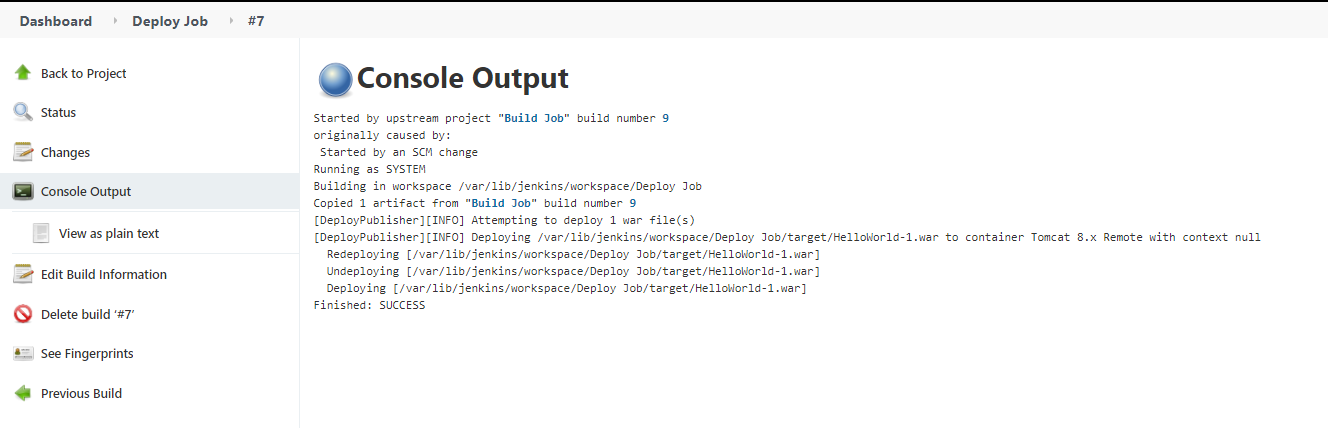
Add the tomcat user settings in the add section and give name to the container

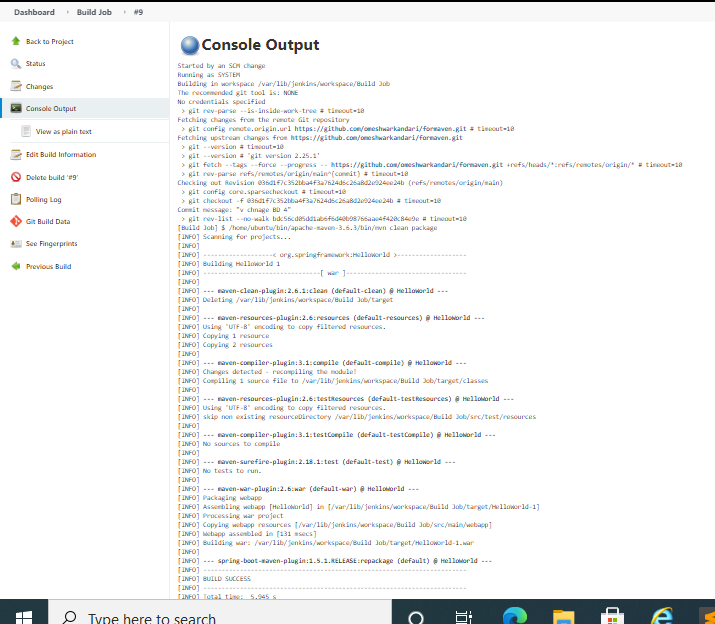


**Sample Output:** Git commit/add (-am) will commit the code change and Git Push will create a new Build within 1 minute based on the SCM Poll settings which is 5-star(every minute) and new build will deploy it automatically into the Tomcat Container. Sample Output also attached.



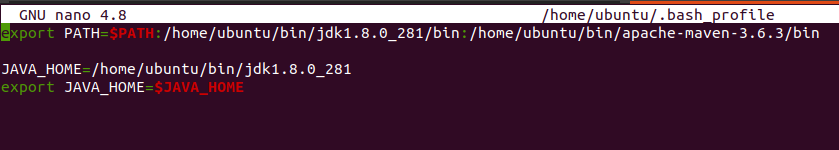
**Deploy Output:**



**Build Output:**

**Jenkins Master Slave Configuration:**

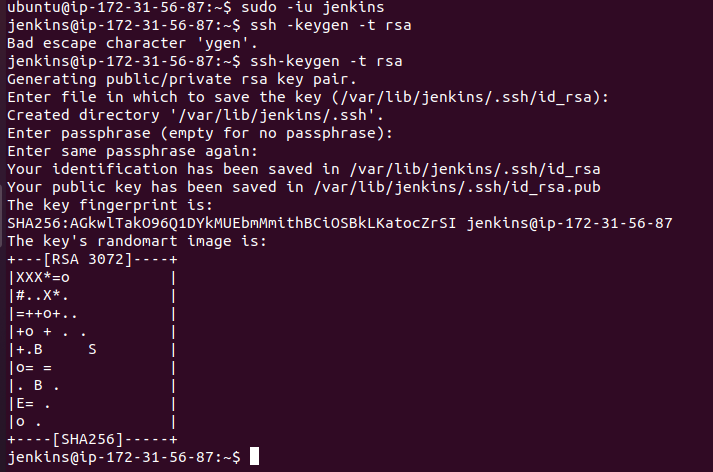
1. Create a Slave machine and install Java/MVN with environment set as per the path in the Jenkins Global Configuration Settings.



1. Create a key pair (Pub/Private): This is needed for the user Jenkins to ssh into Slave Node from the Master Node. Jenkins user Public key from the Master Node will be saved in the root .ssh/authorized\_key of the Slave Node.
2. Public Key Access: Login as Jenkins user in the Master Node use Keygen to access public/private keys.

ubuntu@ip-172-31-56-87:~$ sudo -iu Jenkins

jenkins@ip-172-31-56-87:~$ ssh-keygen -t rsa (-t is type)



jenkins@ip-172-31-56-87:~$ ls –al (it will show .ssh folder)

jenkins@ip-172-31-56-87:~$ ls .ssh

id\_rsa id\_rsa.pub

1. Copy Public Key from the Master Node:

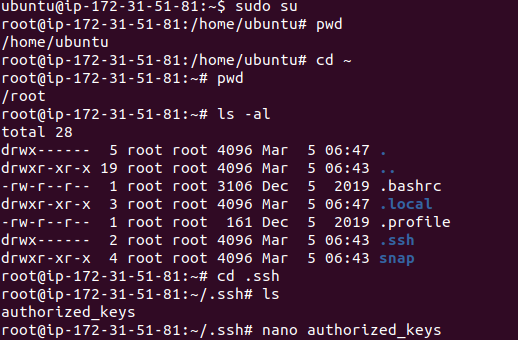
jenkins@ip-172-31-56-87:~$ nano .ssh/id\_rsa.pub

or

jenkins@ip-172-31-56-87:~$ cat .ssh/id\_rsa.pub

1. Paste in the Slave Node:

root@ip-172-31-51-81:~/.ssh# nano authorized\_keys



1. SSH into Slave from Master with Jenkins login: ssh root@salve public ip

jenkins@ip-172-31-56-87:~$ ssh [root@100.27.11.225](mailto:root@100.27.11.225)

Confirm with below command to match the Private IP of the Slave Node:

root@ip-172-31-51-81:~# hostname

ip-172-31-51-81

1. Download a jar file slave.jar which is required to communicate between Jenkins Master Node and Slave Nodes.

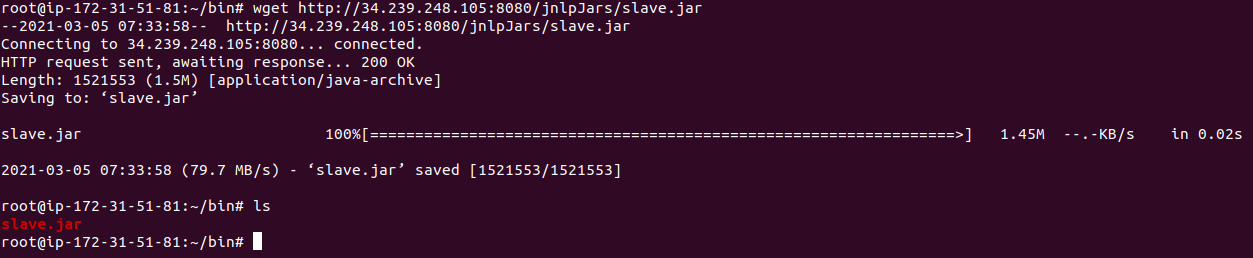
root@ip-172-31-51-81:~# mkdir bin

root@ip-172-31-51-81:~# cd bin/

root@ip-172-31-51-81:~/bin# ls

root@ip-172-31-51-81:~/bin# wget <http://34.239.248.105:8080/jnlpJars/slave.jar>

( [http://](http:// ) Pub-ip Master Node : Jenkin Default Port / jnlpJars / slave.jar)



1. Install a Jre file which will create JVM environment for Master/Slave connection

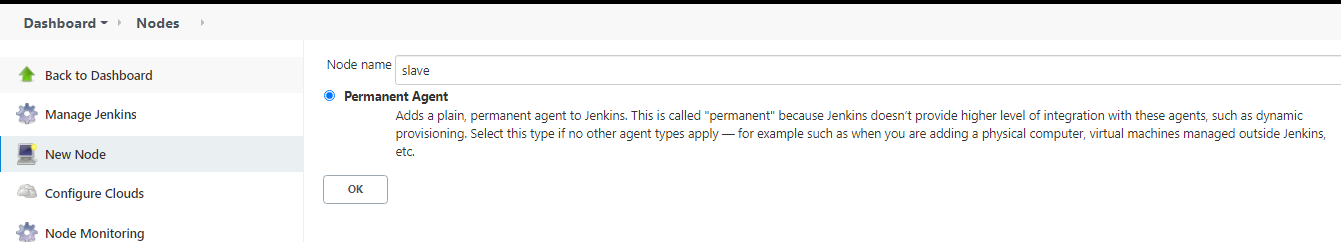
root@ip-172-31-51-81:~/bin# sudo apt update

root@ip-172-31-51-81:~/bin# apt install default-jre

1. Login to Jenkins Console: Add/configure Slave Nodes.

Go to Manage Jenkins -🡪 Manage Nodes & Clouds

1. Create a New Node slave as Permanent Agent



1. Node Configuration:

Remote Root Directory: /var/Jenkins

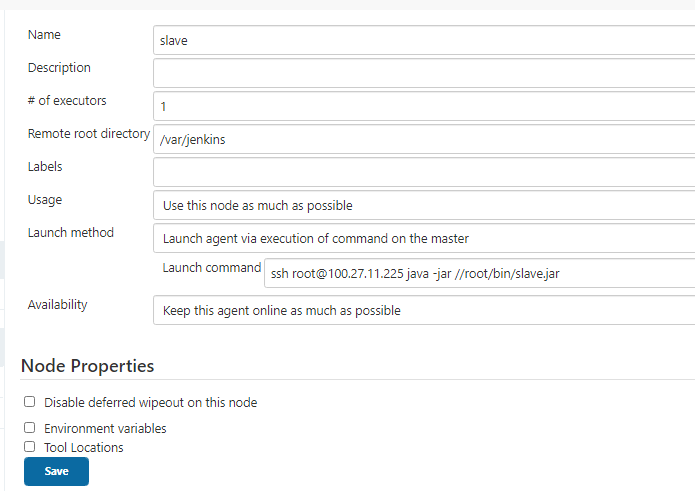
This directory is kind of a storage/volume which will be used by Master to create Workspace where built artifacts will be stored.

Launch Method:

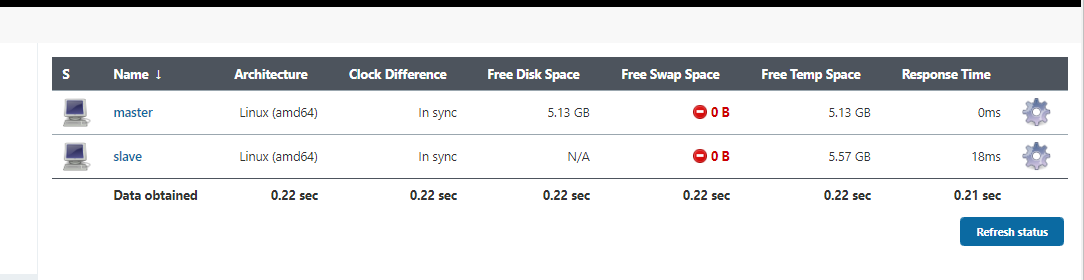
Execution of the command on the master

ssh root@100.27.11.225 java -jar /root/bin/slave.jar

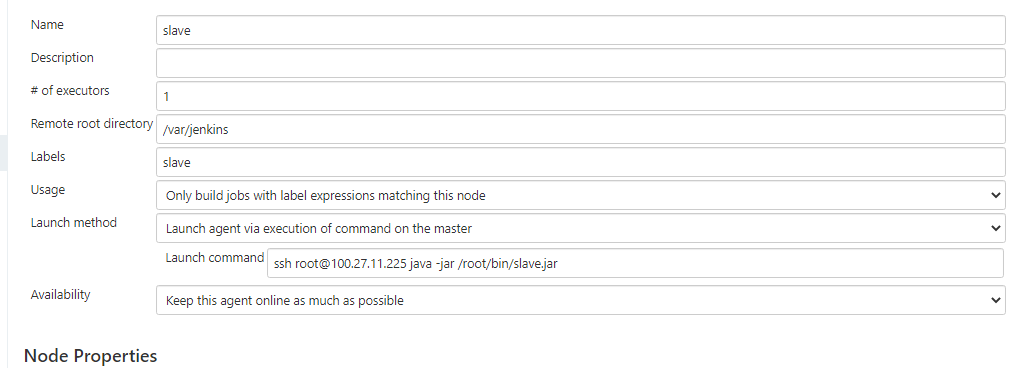
(ssh root @ slave node Public IP java -jar /root/bin/slave.jar )

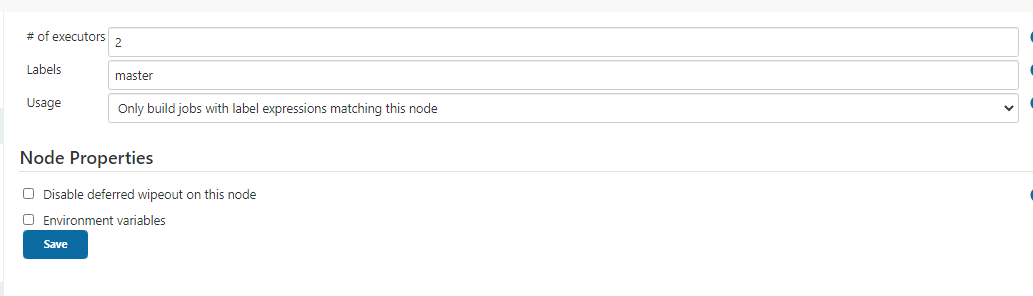


1. Check Node status



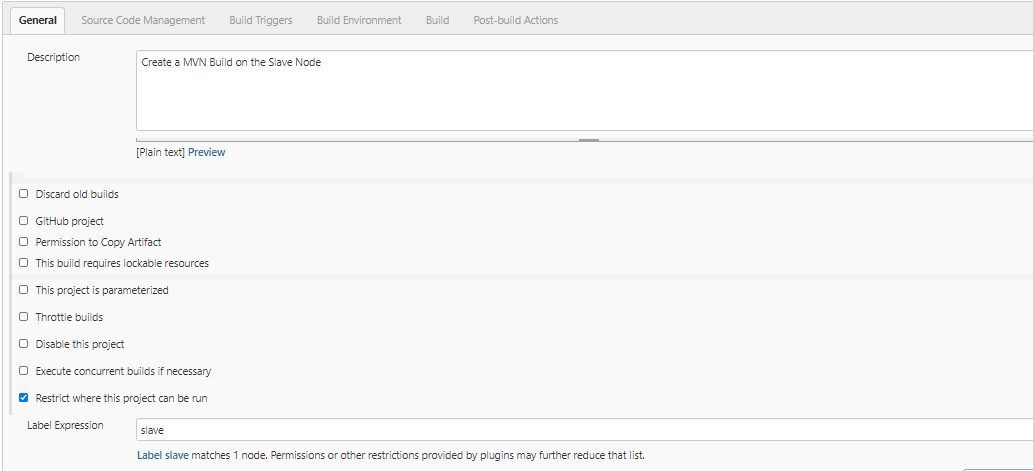
1. Labels: Setting Label and related information is critical to ensure Build/Deploy roles of the Master and Slave.

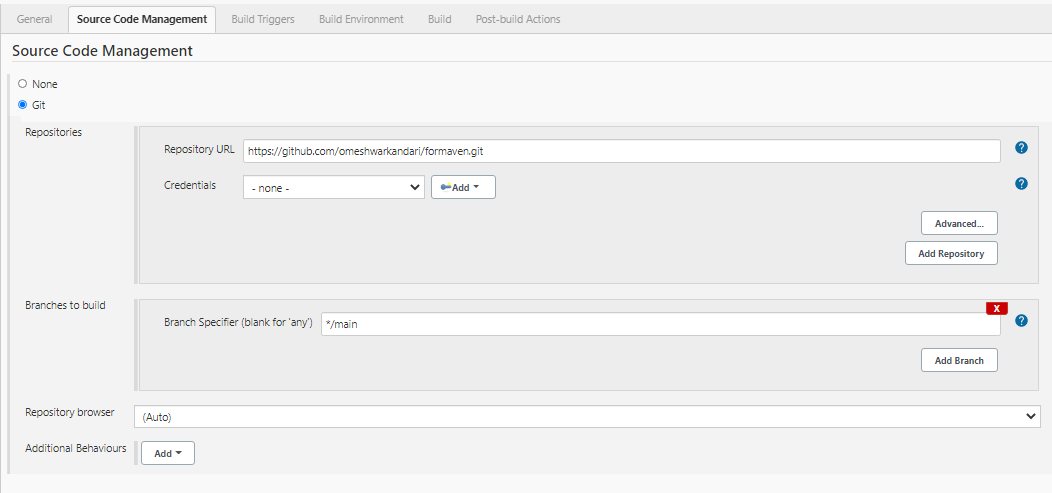
Slave Config:

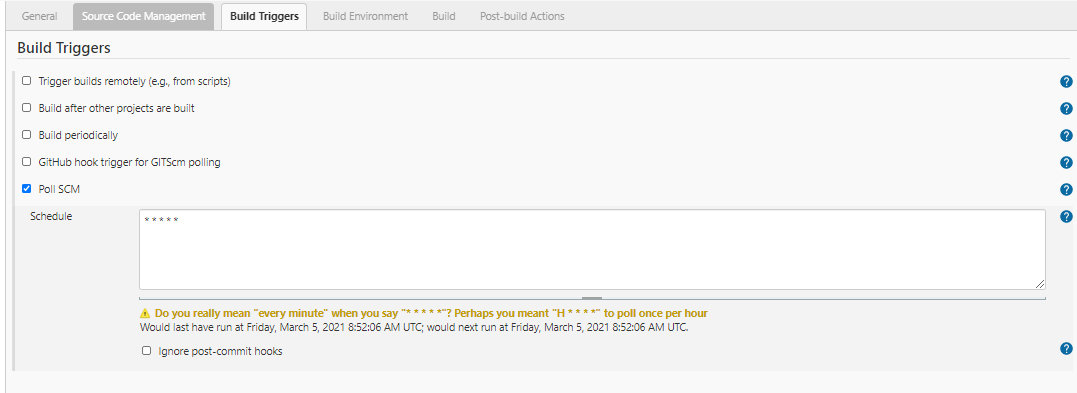
Master Config:

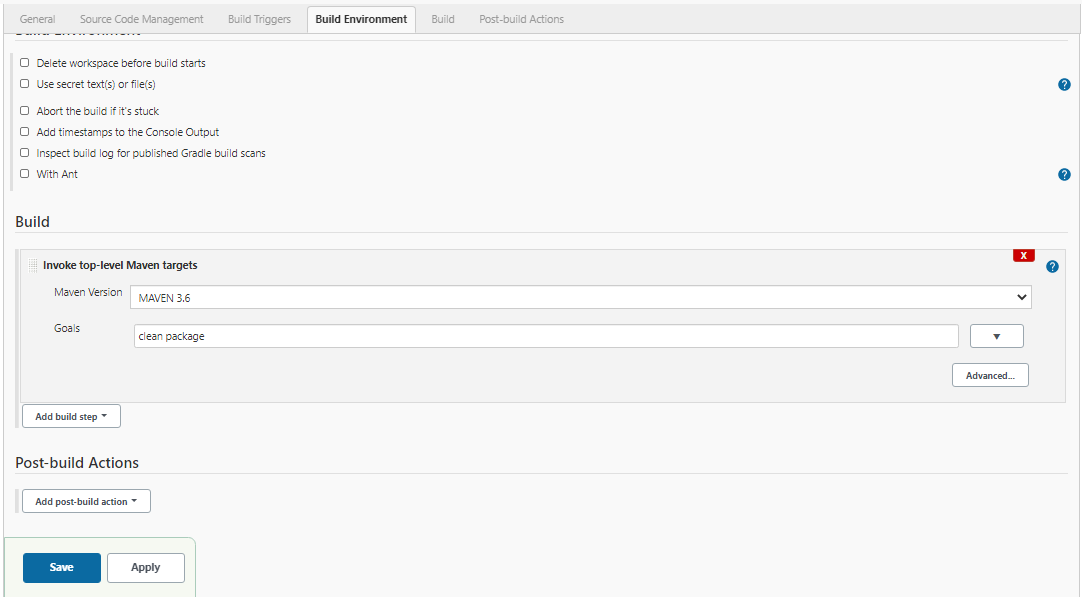
**Build a Maven Project “New Project” in Slave Node**

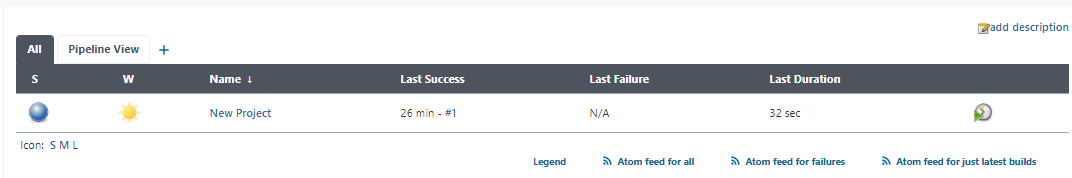
1. Create a new build job “New Project” to create a MVN Build in the Slave Node.











Build was successful and artifact should be in the Workspace under /var/Jenkins as visible below as HellowWorld-1 artifact is created in the directory accessed in slave node.



ubuntu@ip-172-31-56-87:~$ sudo -iu jenkins

jenkins@ip-172-31-56-87:~$ ssh [root@100.27.11.225](mailto:root@100.27.11.225)

root@ip-172-31-51-81:~# cd /var/jenkins/

root@ip-172-31-51-81:/var/jenkins# ls

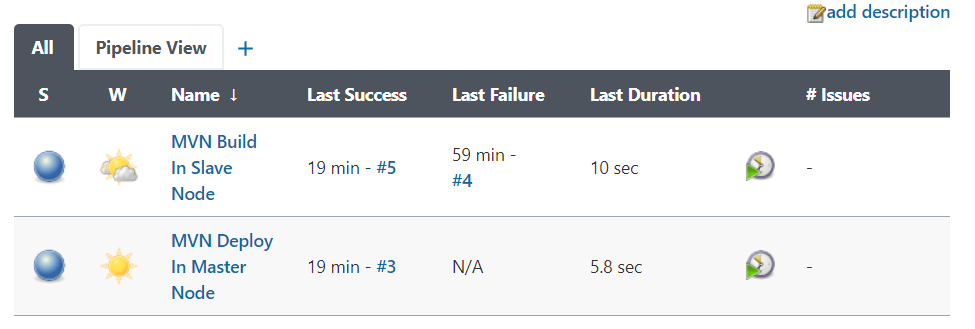
workspace

root@ip-172-31-51-81:/var/jenkins/workspace/New Project/target# ls

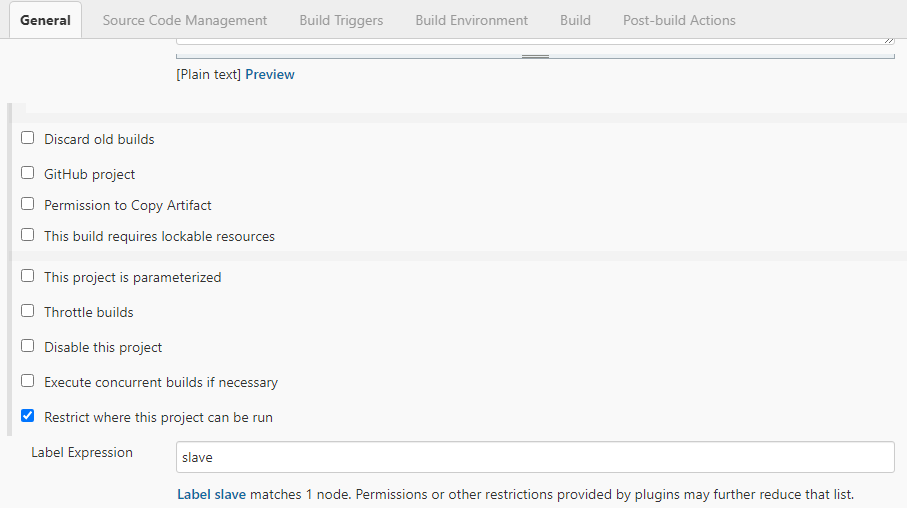
HelloWorld-1 HelloWorld-1.war HelloWorld-1.war.original classes generated-sources maven-archiver maven-status

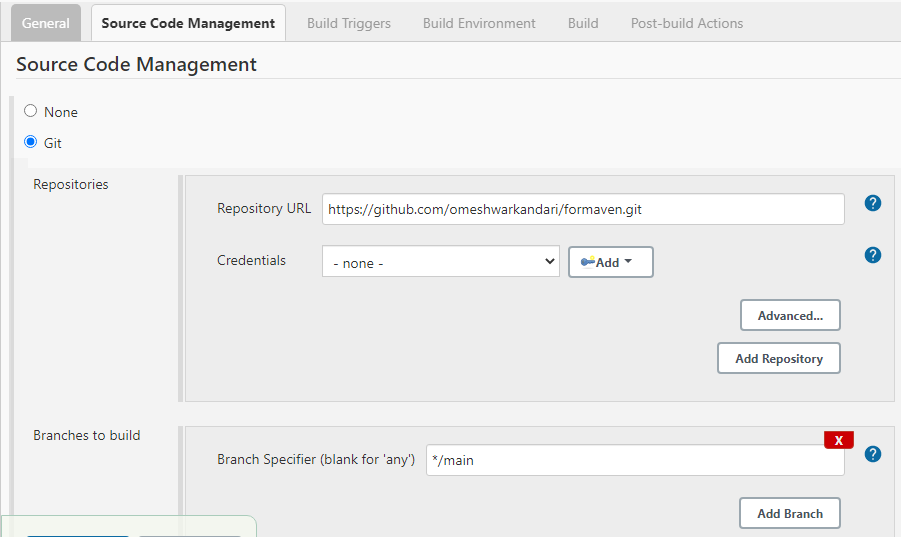
**Project 2: Build MVN Artifact in the Salve Node & Deploy MVN Artifact in the Master Node**

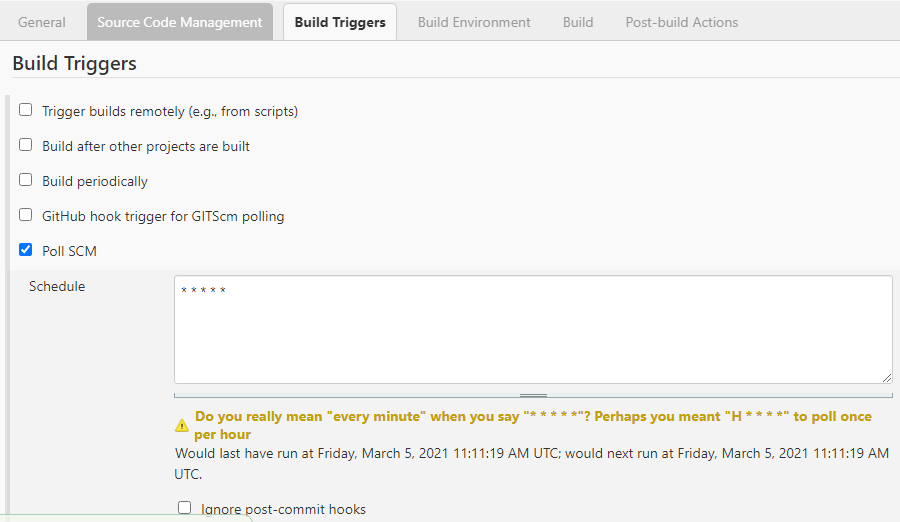
Artifact the war file is built In Slave Node and deployed in the Tomcat Container in the Master Node

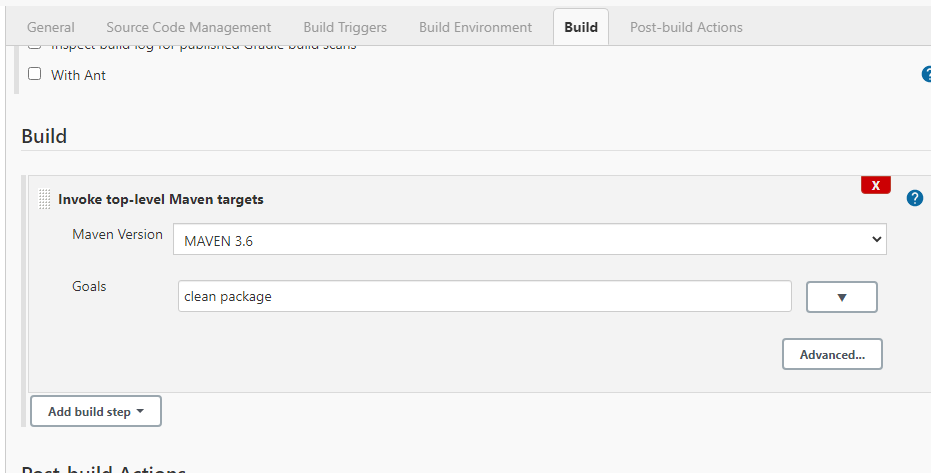


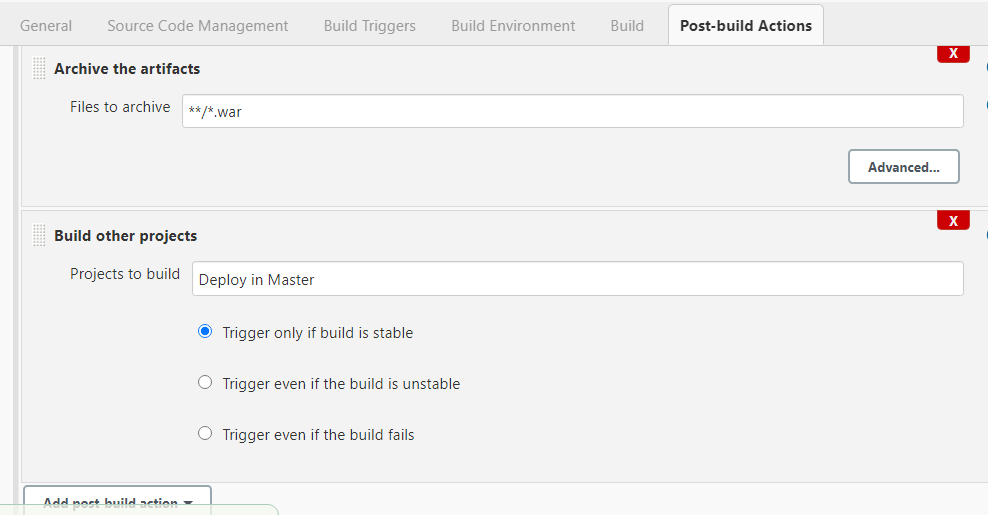
Build Project: MVN Build In Slave Node



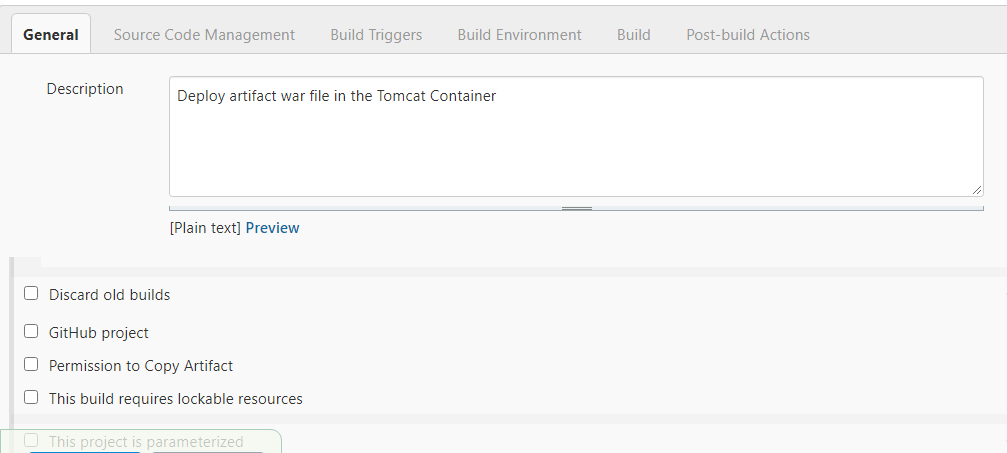




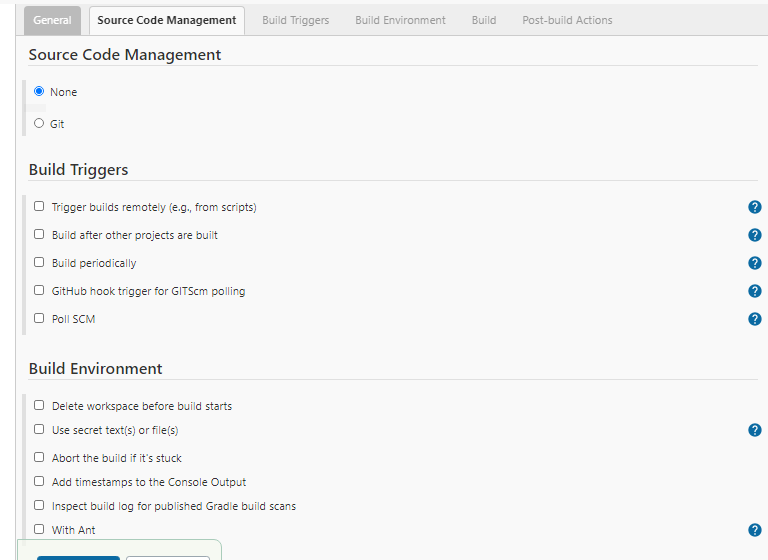


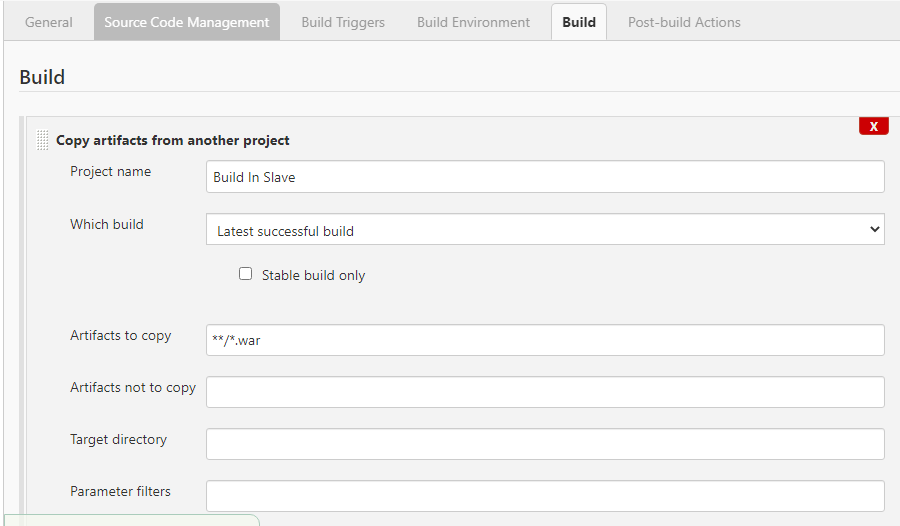


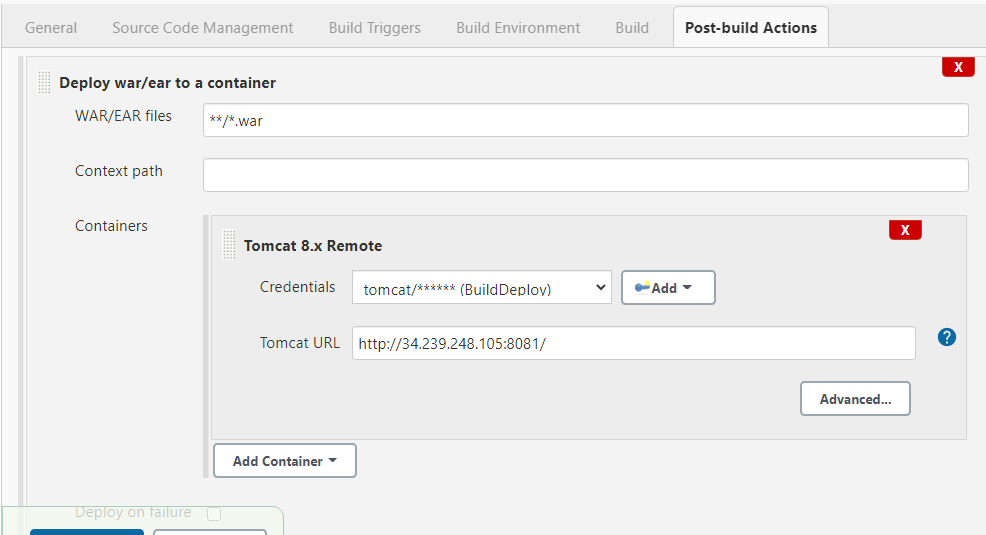
Deploy Project: MVN Deploy In Master Node

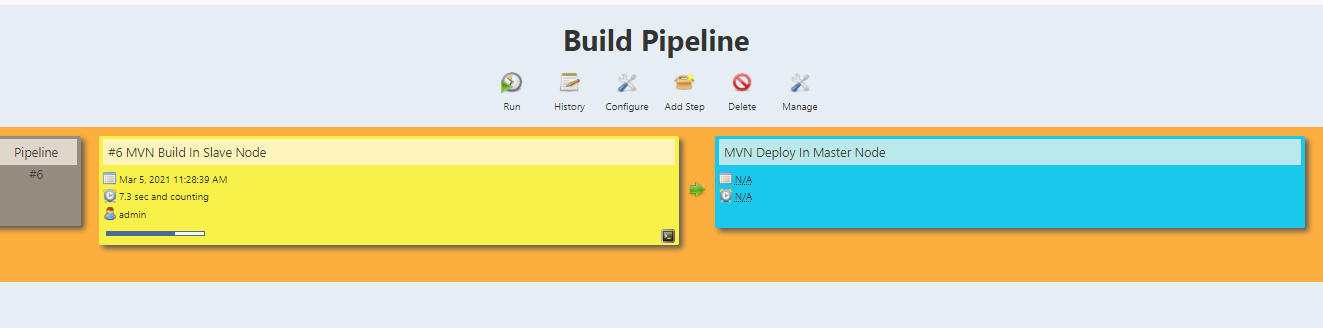


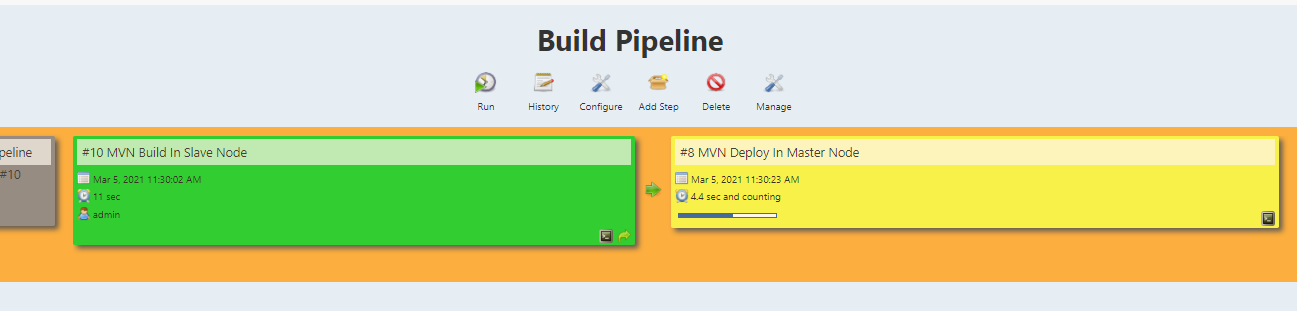
Nothing in SCM/BT/BE:

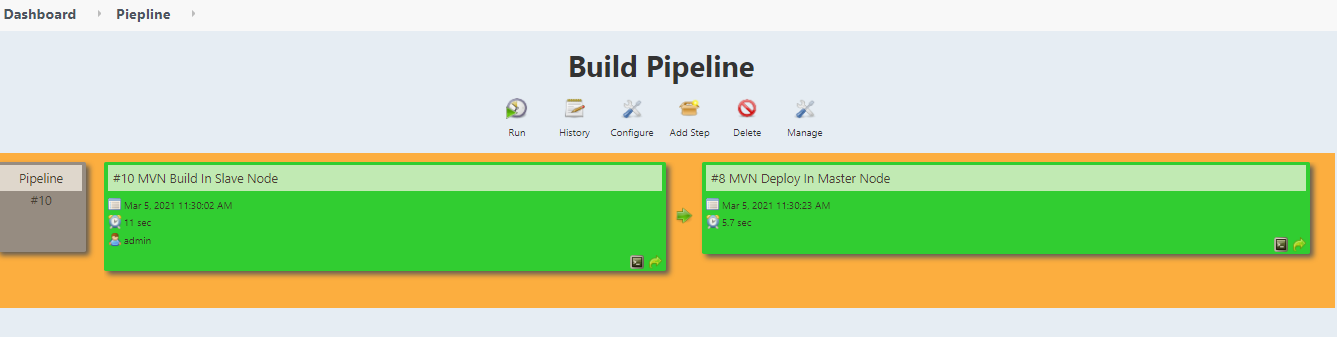












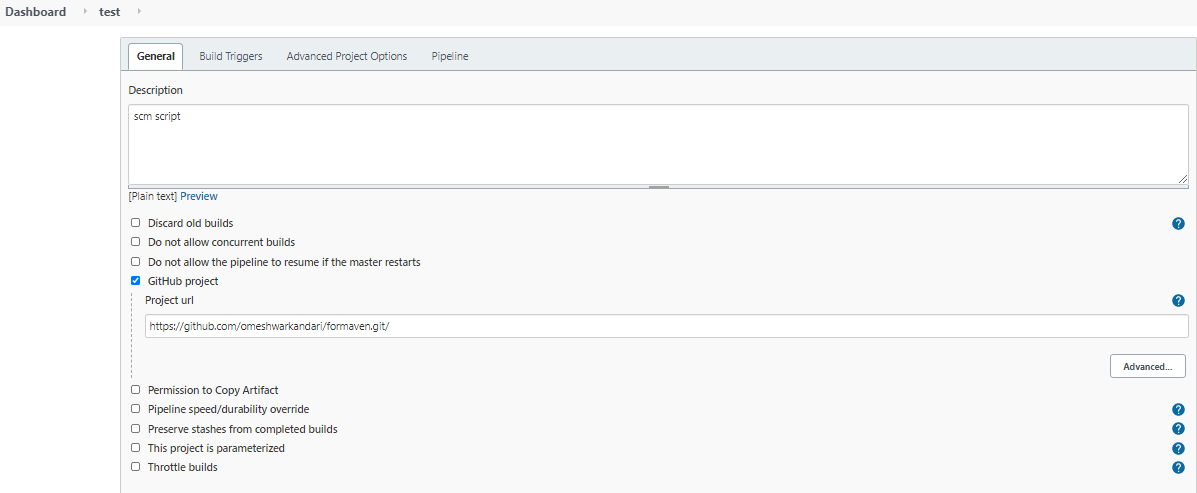
**SCM Script for Jenkins Pipeline**

Step1: Create a script and name it “Jenkinsfile “.

Script type: Groovy (Build/Deploy files) and Declarative (demo file)

Groovy script uses node {} syntax while declarative uses pipleline {} syntax.

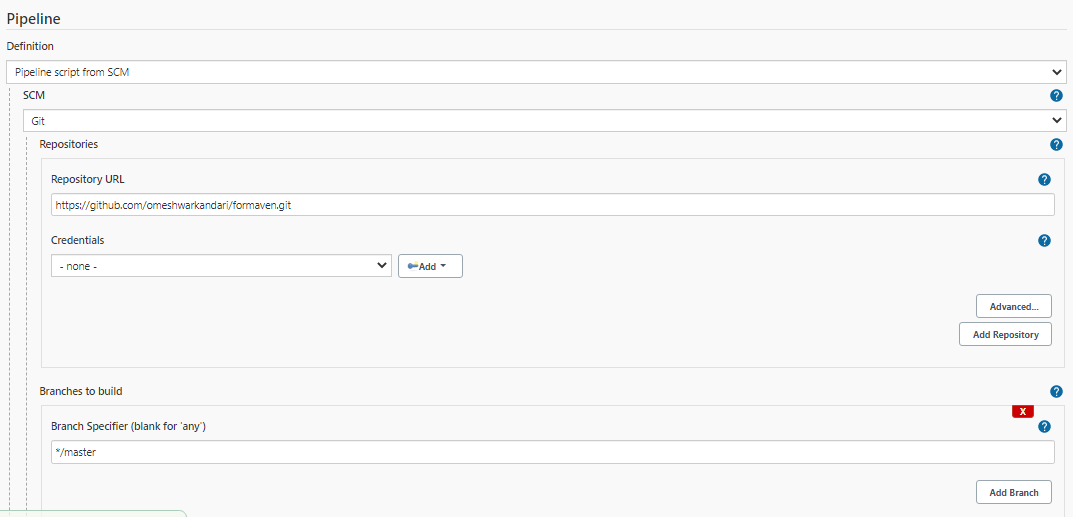
Step2: Create a Pipeline e.g. “test” by selecting “pipeline” project and ok.

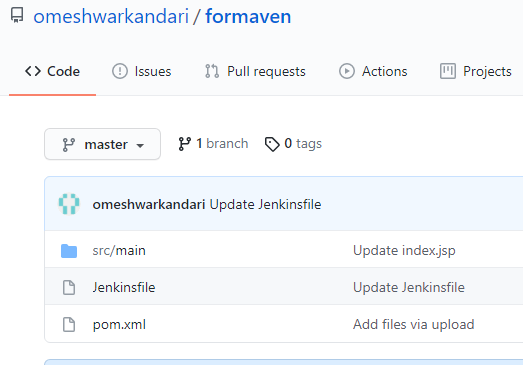


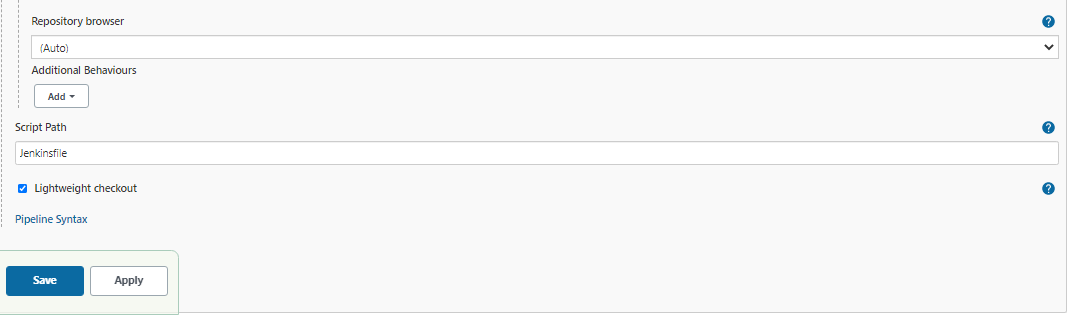


Pipleline Definition Types: we can use script in tow ways based on the selection of pipeline definition.

1. Pipeline script from scm: We have selected it and this says that create a script named “Jenkinsfile” and save it in the preferably root of the SCM the repository for the source code.
2. Pipeline script: Here the script is written in line inside the Pipeline definition built-in template.







**Step3: Create the script**

Node {

Stage ('SCM Checkout') {

git 'https://github.com/omeshwarkandari/formaven.git'

}

Stage ('Build') {

def mvnHome = tool name: 'maven3.6', type: 'maven'

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

}

Stage ('Deploy') {

sshagent (['Test-Deploy']) {

sh "scp -o StrictHostKeyChecking=no target/\*.war [ubuntu@172.31.52.3:~/apache-](mailto:ubuntu@172.31.52.3:~/apache-) tomcat-9.0.45/webapps"

}

}

}

Pipeline runs stage by stage as defined in the script.

Useful plug-in:  [Pipeline Maven Integration Plugin](https://plugins.jenkins.io/pipeline-maven), [Maven Integration plugin](https://plugins.jenkins.io/maven-plugin" \t "_blank) , [Pipeline](https://plugins.jenkins.io/workflow-aggregator)[SSH Agent Plugin](https://plugins.jenkins.io/ssh-agent)

SCM stage: Jenkins will look for the source code commit in the repo and trigger the build process.

Build Stage: It will build a clean Maven package and the Build Stage script follows the sample script generated through the Pipeline Syntax based on the Project Type.

e.g. we are ruining the Maven build so we followed the below process to generate a script under tools.

Go to Pipeline syntax page and Generate Pipeline Script for Maven 3.6

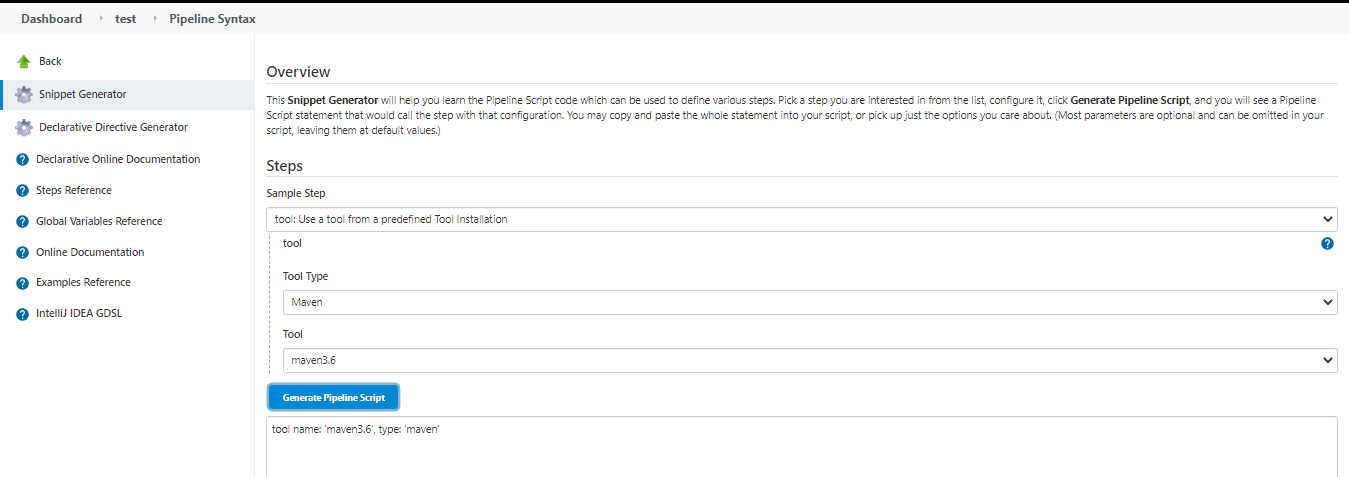
**tool name: 'maven3.6', type: 'maven'**

**def mvnHome = tool name: 'maven3.6', type: 'maven'**

(We have added a definition for declaring the tool as mvnHome)

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

Sh is used for shell script.



**Deploy Stage:**

Stage ('Deploy') {

sshagent (['Test-Deploy']) {

sh "scp -o StrictHostKeyChecking=no target/\*.war [ubuntu@172.31.52.3:~/apache-](mailto:ubuntu@172.31.52.3:~/apache-) tomcat-9.0.45/webapps"

}

}

**SSH Agent script:**

sshagent ( [ ‘Test-Deploy ‘ ] ) {

// some block

}

Select the SSG agent and add the agent by selecting Jenkins and configuring ssh credentials to ssh into the remote container e.g. Tomcat in this case.

ID and Description can be anything.

User: the use who has access to login to remote server e.g. “Ubuntu” in case of Ubuntu EC2 and “ec2-user” in case of Amazon/CentOS.

Private Key: The pem file used for this instance.

Generate Pipeline Script which will follow the ID info e.g. Test-Deploy will generate

sshagent(['Test-Deploy']) {

// some block

}

sh "scp -o StrictHostKeyChecking=no target/\*.war [ubuntu@172.31.52.3: ~/apache-](mailto:ubuntu@172.31.52.3:%20~/apache-) tomcat-9.0.45/webapps"

**scp command script:**

scp <src\_file> username@IP: <dest\_path>

**scp**: It follows the sample script to add “-o StrictHostKeyChecking=no” to disable the host key strict check.

node {

sshagent (credentials: ['deploy-dev']) {

sh 'ssh -o StrictHostKeyChecking=no -l cloudbees 192.168.1.106 uname -a'

}}

**scr\_file:**

It follows the target folder in the workspace where build is taking place “test”

$ cd /var/lib/jenkins/workspace/test

Run a command tree from target directory to get the root file path where build artifact “HelloWorld-1.war” is saved and since there is no clarity in this case so source file is taken as “target/\*.war” because “/var/lib/jenkins/workspace/test” is auto archieved.

ubuntu@ip-172-31-62-132:/var/lib/jenkins/workspace/test/target$ tree.

├── HelloWorld-1

│   ├── META-INF

│   ├── WEB-INF

│   │   ├── classes

│   │   │   ├── app

│   │   │   │   └── Application.class

│   │   │   ├── application.properties

│   │   │   └── public

│   │   │   └── error

│   │   │   ├── 404.html

│   │   │   └── 500.html

│   │   └── lib

│   │   ├── classmate-1.3.3.jar

│   │   ├── commons-lang3-3.0.jar

│   │   ├── hibernate-validator-5.3.4.Final.jar

│   │   ├── jackson-annotations-2.8.0.jar

│   │   ├── jackson-core-2.8.6.jar

│   │   ├── jackson-databind-2.8.6.jar

│   │   ├── jboss-logging-3.3.0.Final.jar

│   │   ├── jcl-over-slf4j-1.7.22.jar

│   │   ├── jul-to-slf4j-1.7.22.jar

│   │   ├── log4j-over-slf4j-1.7.22.jar

│   │   ├── logback-classic-1.1.9.jar

│   │   ├── logback-core-1.1.9.jar

│   │   ├── slf4j-api-1.7.22.jar

│   │   ├── snakeyaml-1.17.jar

│   │   ├── spring-aop-4.3.6.RELEASE.jar

│   │   ├── spring-beans-4.3.6.RELEASE.jar

│   │   ├── spring-boot-1.5.1.RELEASE.jar

│   │   ├── spring-boot-autoconfigure-1.5.1.RELEASE.jar

│   │   ├── spring-boot-starter-1.5.1.RELEASE.jar

│   │   ├── spring-boot-starter-logging-1.5.1.RELEASE.jar

│   │   ├── spring-boot-starter-web-1.5.1.RELEASE.jar

│   │   ├── spring-context-4.3.6.RELEASE.jar

│   │   ├── spring-core-4.3.6.RELEASE.jar

│   │   ├── spring-expression-4.3.6.RELEASE.jar

│   │   ├── spring-web-4.3.6.RELEASE.jar

│   │   ├── spring-webmvc-4.3.6.RELEASE.jar

│   │   └── validation-api-1.1.0.Final.jar

│   ├── assets

│   │   ├── css

│   │   │   ├── bootstrap.css

│   │   │   └── font-awesome.css

│   │   └── vendor

│   │   └── jquery-2.1.0.min.js

│   ├── cloudcomputing.png

│   ├── home.jsp

│   ├── index.jsp

│   └── login\_controller.jsp

├── HelloWorld-1.war

├── HelloWorld-1.war.original

├── classes

│   ├── app

│   │   └── Application.class

│   ├── application.properties

│   └── public

│   └── error

│   ├── 404.html

│   └── 500.html

├── generated-sources

│   └── annotations

├── maven-archiver

│   └── pom.properties

└── maven-status

└── maven-compiler-plugin

└── compile

└── default-compile

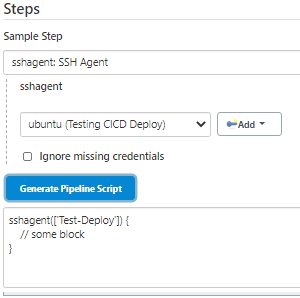
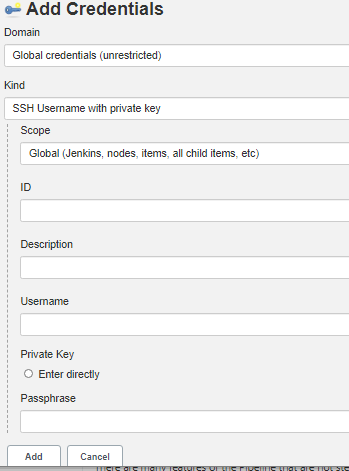
├── createdFiles.lst

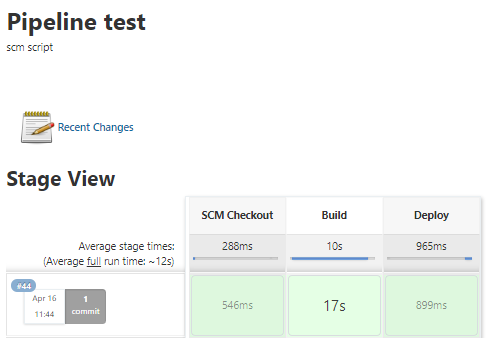
└── inputFiles.lst

**username@IP:** ubuntu@internal ip address of the remote Tomcat server

**Dest\_path:** It’s the root webapps folder where war file is deployed.

“[~ /apache-](mailto:ubuntu@172.31.52.3:%20~/apache-) tomcat-9.0.45/webapps” added ~ to provide access on home directory.



**Output for the Scuccessful Test:**

Started by an SCM change

Obtained Jenkinsfile from git https://github.com/omeshwarkandari/formaven.git

Running in Durability level: MAX\_SURVIVABILITY

[Pipeline] Start of Pipeline

[Pipeline] node

Running on Jenkins in /var/lib/jenkins/workspace/test

[Pipeline] {

[Pipeline] stage

[Pipeline] { (SCM Checkout)

[Pipeline] git

The recommended git tool is: git

No credentials specified

> git rev-parse --is-inside-work-tree # timeout=10

Fetching changes from the remote Git repository

> git config remote.origin.url https://github.com/omeshwarkandari/formaven.git # timeout=10

Fetching upstream changes from https://github.com/omeshwarkandari/formaven.git

> git --version # timeout=10

> git --version # 'git version 2.25.1'

> git fetch --tags --force --progress -- https://github.com/omeshwarkandari/formaven.git +refs/heads/\*:refs/remotes/origin/\* # timeout=10

> git rev-parse refs/remotes/origin/master^{commit} # timeout=10

Checking out Revision cb340843832ef1fdf548aa34ab74bb58e1d5a7a1 (refs/remotes/origin/master)

> git config core.sparsecheckout # timeout=10

> git checkout -f cb340843832ef1fdf548aa34ab74bb58e1d5a7a1 # timeout=10

> git branch -a -v --no-abbrev # timeout=10

> git branch -D master # timeout=10

> git checkout -b master cb340843832ef1fdf548aa34ab74bb58e1d5a7a1 # timeout=10

Commit message: "Test1"

> git rev-list --no-walk e82057bedd4d38b624552492bce8869cb8f6d3cc # timeout=10

[Pipeline] }

[Pipeline] // stage

[Pipeline] stage

[Pipeline] { (Build)

[Pipeline] tool

[Pipeline] sh

+ /home/ubuntu/apache-maven-3.6.3/bin/mvn clean package

[INFO] Scanning for projects...

[INFO]

[INFO] -------------------< org.springframework:HelloWorld >-------------------

[INFO] Building HelloWorld 1

[INFO] --------------------------------[ war ]---------------------------------

[INFO]

[INFO] --- maven-clean-plugin:2.6.1:clean (default-clean) @ HelloWorld ---

[INFO] Deleting /var/lib/jenkins/workspace/test/target

[INFO]

[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ HelloWorld ---

[INFO] Using 'UTF-8' encoding to copy filtered resources.

[INFO] Copying 1 resource

[INFO] Copying 2 resources

[INFO]

[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ HelloWorld ---

[INFO] Changes detected - recompiling the module!

[INFO] Compiling 1 source file to /var/lib/jenkins/workspace/test/target/classes

[INFO]

[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ HelloWorld ---

[INFO] Using 'UTF-8' encoding to copy filtered resources.

[INFO] skip non existing resourceDirectory /var/lib/jenkins/workspace/test/src/test/resources

[INFO]

[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ HelloWorld ---

[INFO] No sources to compile

[INFO]

[INFO] --- maven-surefire-plugin:2.18.1:test (default-test) @ HelloWorld ---

[INFO] No tests to run.

[INFO]

[INFO] --- maven-war-plugin:2.6:war (default-war) @ HelloWorld ---

WARNING: An illegal reflective access operation has occurred

WARNING: Illegal reflective access by com.thoughtworks.xstream.converters.collections.TreeMapConverter (file:/var/lib/jenkins/.m2/repository/com/thoughtworks/xstream/xstream/1.4.4/xstream-1.4.4.jar) to field java.util.TreeMap.comparator

WARNING: Please consider reporting this to the maintainers of com.thoughtworks.xstream.converters.collections.TreeMapConverter

WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations

WARNING: All illegal access operations will be denied in a future release

[INFO] Packaging webapp

[INFO] Assembling webapp [HelloWorld] in [/var/lib/jenkins/workspace/test/target/HelloWorld-1]

[INFO] Processing war project

[INFO] Copying webapp resources [/var/lib/jenkins/workspace/test/src/main/webapp]

[INFO] Webapp assembled in [307 msecs]

[INFO] Building war: /var/lib/jenkins/workspace/test/target/HelloWorld-1.war

[INFO]

[INFO] --- spring-boot-maven-plugin:1.5.1.RELEASE:repackage (default) @ HelloWorld ---

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------

[INFO] Total time: 11.735 s

[INFO] Finished at: 2021-04-16T06:14:26Z

[INFO] ------------------------------------------------------------------------

[Pipeline] }

[Pipeline] // stage

[Pipeline] stage

[Pipeline] { (Deploy)

[Pipeline] sshagent

[ssh-agent] Using credentials ubuntu (Testing CICD Deploy)

[ssh-agent] Looking for ssh-agent implementation...

[ssh-agent] Exec ssh-agent (binary ssh-agent on a remote machine)

$ ssh-agent

SSH\_AUTH\_SOCK=/tmp/ssh-nuGcU1MTYMC0/agent.100647

SSH\_AGENT\_PID=100649

Running ssh-add (command line suppressed)

Identity added: /var/lib/jenkins/workspace/test@tmp/private\_key\_8066735775461345974.key (/var/lib/jenkins/workspace/test@tmp/private\_key\_8066735775461345974.key)

[ssh-agent] Started.

[Pipeline] {

[Pipeline] sh

+ scp -o StrictHostKeyChecking=no target/HelloWorld-1.war ubuntu@172.31.52.3:~/apache-tomcat-9.0.45/webapps

[Pipeline] }

$ ssh-agent -k

unset SSH\_AUTH\_SOCK;

unset SSH\_AGENT\_PID;

echo Agent pid 100649 killed;

[ssh-agent] Stopped.

[Pipeline] // sshagent

[Pipeline] }

[Pipeline] // stage

[Pipeline] }

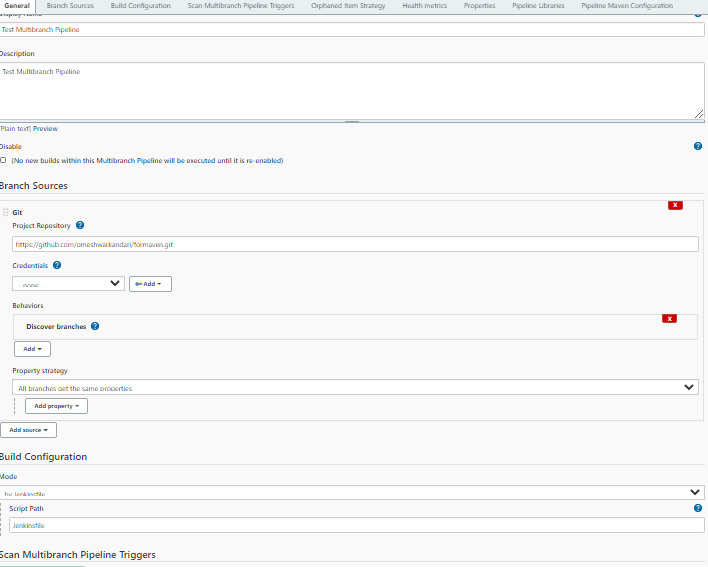
[Pipeline] // node

[Pipeline] End of Pipeline

Finished: SUCCESS

Mutli-branch Pipeline:

It works based on the jenkinsfile availability in a branch which means once we have below plug-in installed and pipeline configured then we can trigger a job from a branch which has jenkinsfile.



Configure Auto Scan Multiple Pipeline Triggers:

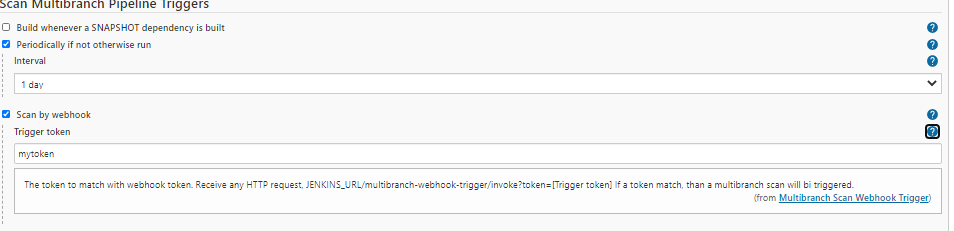
Install the plug-in

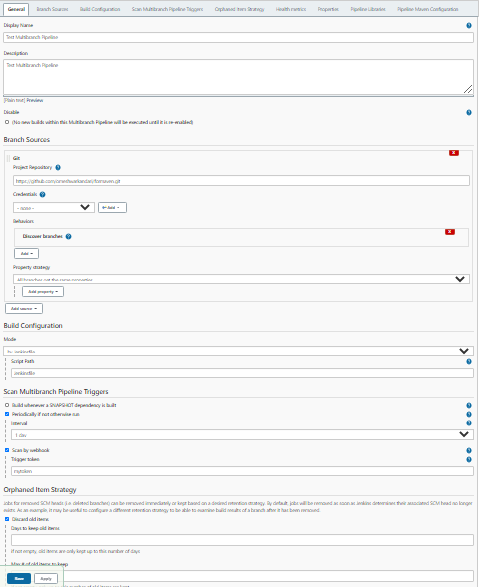


Once plugin installed: we need to configure pipeline as well as Repository

Configure pipeline and provide some name to token e.g. mytoken and add The URL under mytoken in the Git Repo.

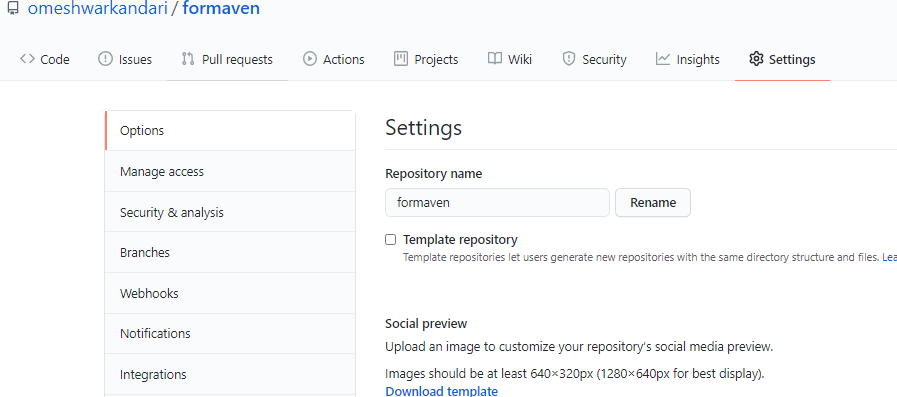
JENKINS\_URL/multibranch-webhook-trigger/invoke?token=[Trigger token]





Git Repo Configuration:

Repo --- Settings ---- Webhooks



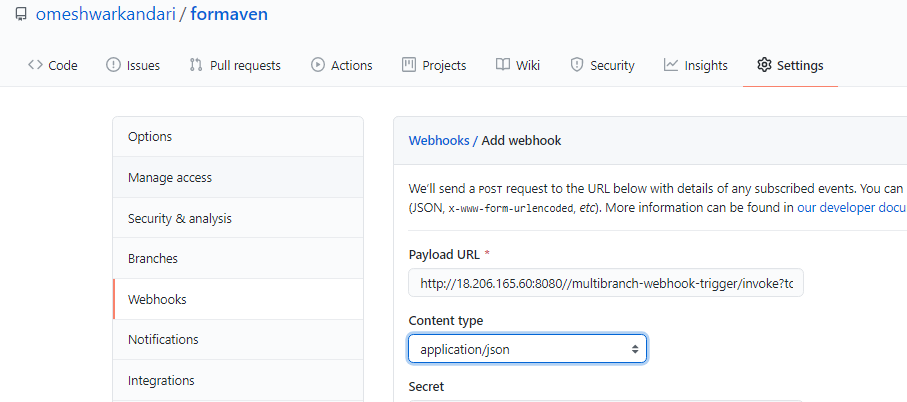
Add Webhook: “JENKINS\_URL/multibranch-webhook-trigger/invoke?token=[Trigger token]”

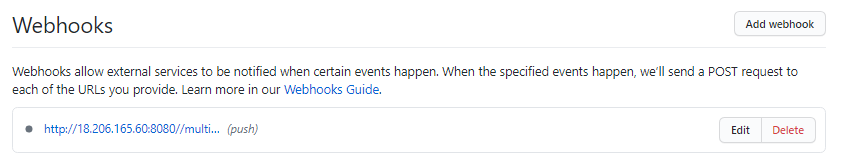
Customize the url by adding Jenkins url e.g. http://18.206.165.60:8080/ and followed by our token “mytoken”

<http://18.206.165.60:8080//multibranch-webhook-trigger/invoke?token=mytoken>

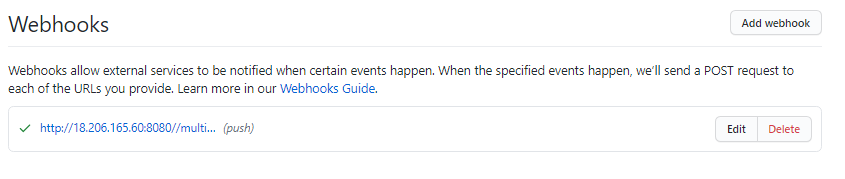
Change Content type as “application/json”

Finally add the webhook.





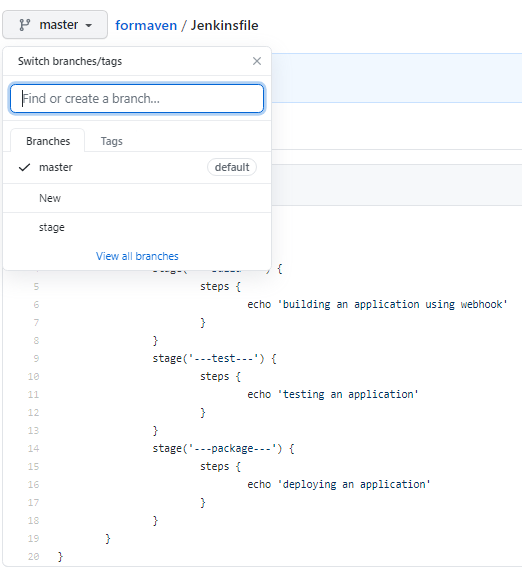
Refresh the Repo will enable the webhook.

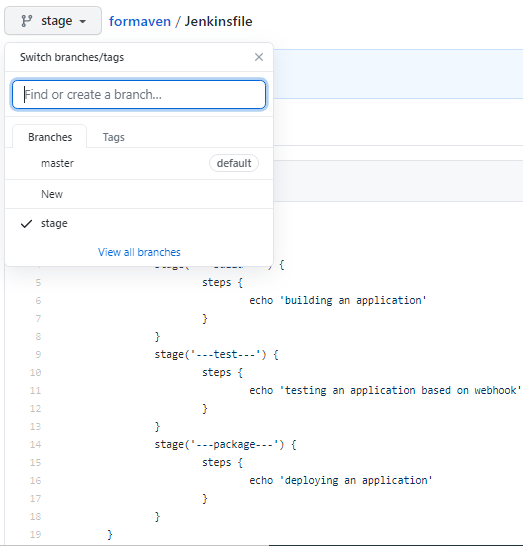
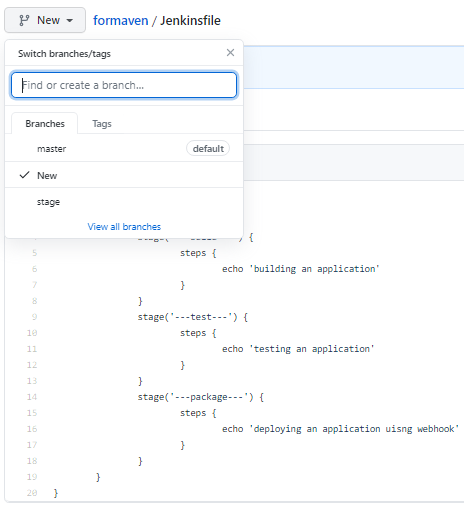


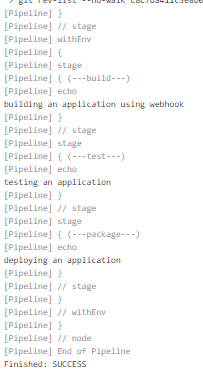
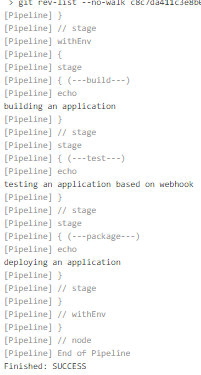
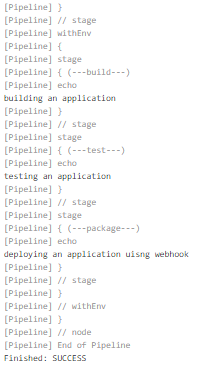
Now we can see the pipeline changes based on Repo selection – Master or any other branch.

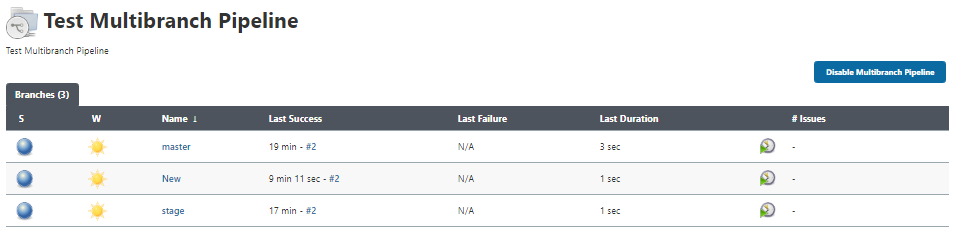
e.g. I first updated jenkinsfile under Master branch for “ build” so it changed only the Master Branch Pipeline, then changed Stage branch for “test” and finally changed New Branch .

SCM Update:

Master Repo updated for the Build Script.



If we remove jenkinsfile from any branch then the pipeline is auto deleted.

e.g. lets remove jenkinsfile from the New Branch.

$ git checkout New

$ git branch

\* New

master

stage

$ ls

Githelp Info Installation 'Jenkins Master-slave' Jenkinsfile README.md formaven pom.xml src

$ rm -rf Jenkinsfile

$ ls

Githelp Info Installation 'Jenkins Master-slave' README.md formaven pom.xml src

$ git commit -am "remode Jenkinsfile from New branch"

$ git push

Enumerating objects: 3, done.

Counting objects: 100% (3/3), done.

Delta compression using up to 8 threads

Compressing objects: 100% (2/2), done.

Writing objects: 100% (2/2), 244 bytes | 244.00 KiB/s, done.

Total 2 (delta 1), reused 0 (delta 0)

remote: Resolving deltas: 100% (1/1), completed with 1 local object.

To https://github.com/omeshwarkandari/formaven.git

bcfc029..c0ed434 New -> New

We can observe that New branch is auto deleted from the multi-branch pipeline.

