SUBVERSION

How to create repository in shortcut way:

[vagrant@localhost ~]$ pwd

/home/vagrant

[vagrant@localhost ~]$ cd subversion-1.9.4-1/subversion/bin/

[vagrant@localhost bin]$ sh svnadmin create /home/vagrant/demo

[vagrant@localhost bin]$ vi /home/vagrant/subversion-1.9.4-1/apache2/conf/httpd.conf

[vagrant@localhost bin]$ cd ../..

[vagrant@localhost subversion-1.9.4-1]$ sh ctlscript.sh restart

How to create folder in the repository:

* Create one folder in desktop ex: SVN practice.
* Open that command prompt inside the svn practice folder.
* Run the command

C:\Users\Hare Rama\Desktop\svn practice>svn co <http://192.168.33.10:8080/subversion/>

* Once it downloads all repository I will get the subversion folder.then go to the subversion folder using the command cd subversion.
* Run the command svn mkdir foldername.
* After that run the cmd svn ci –m “msg”.

C:\Users\Hare Rama\Desktop\svn practice>svn co http://192.168.33.10:8080/subversion/

Checked out revision 4.

C:\Users\Hare Rama\Desktop\svn practice>cd subversion

C:\Users\Hare Rama\Desktop\svn practice\subversion>svn mkdir java

A java 🡪Adding

C:\Users\Hare Rama\Desktop\svn practice\subversion>svn ci -m "folder"

Adding java

Committed revision 5.

C:\Users\Hare Rama\Desktop\svn practice\subversion>svn delete java

D java 🡪deleting

C:\Users\Hare Rama\Desktop\svn practice\subversion>svn ci -m "del"

Deleting java

Committed revision 6.

How to create file in repository :

C:\Users\Hare Rama\Desktop\svn practice\subversion>svn st

? rama.txt 🡪indicates manually added file

C:\Users\Hare Rama\Desktop\svn practice\subversion>svn add rama.txt

A rama.txt 🡪 adding

C:\Users\Hare Rama\Desktop\svn practice\subversion>svn ci -m "txt'

Adding rama.txt

Transmitting file data .

Committed revision 7.

C:\Users\Hare Rama\Desktop\svn practice\subversion>svn st

M rama.txt 🡪modified

C:\Users\Hare Rama\Desktop\svn practice\subversion>svn ci -m "modified"

Sending rama.txt

Transmitting file data .

Committed revision 8.

C:\Users\Hare Rama\Desktop\svn practice\subversion>svn delete rama.txt

D rama.txt 🡪 deleting

C:\Users\Hare Rama\Desktop\svn practice\subversion>svn ci -m "del"

Deleting rama.txt

Committed revision 9.

How to copy the code from one folder to another folder:

C:\Users\Hare Rama\Desktop\svn practice\subversion>svn mkdir test1 test2

A test1

A test2

C:\Users\Hare Rama\Desktop\svn practice\subversion>svn ci -m "folder"

Adding test1

Adding test2

Committed revision 12.

C:\Users\Hare Rama\Desktop\svn practice\subversion>svn st

? test1\java.txt.txtl

C:\Users\Hare Rama\Desktop\svn practice\subversion>cd test1

C:\Users\Hare Rama\Desktop\svn practice\subversion\test1>svn add java.txt.txt

A java.txt.txt

C:\Users\Hare Rama\Desktop\svn practice\subversion\test1>svn ci -m "text"

Adding java.txt.txt

Transmitting file data .

Committed revision 13.

C:\Users\Hare Rama\Desktop\svn practice\subversion\test1>cd ..

C:\Users\Hare Rama\Desktop\svn practice\subversion>Svn copy http://192.168.33.10:8080/subversion/test1/java.txt.txt http://192.168.33.10:8080/subversion/test2/ -m "message"

Committed revision 14.

C:\Users\Hare Rama\Desktop\svn practice\subversion>

Svn merging concept:

* create a new repository suppose ‘merge1’

[vagrant@localhost ~]$ cd subversion-1.9.4-1/subversion/bin/

[vagrant@localhost bin]$ sh svnadmin create /home/vagrant/merge1

[vagrant@localhost bin]$ cd ../..

[vagrant@localhost subversion-1.9.4-1]$ vi apache2/conf/httpd.conf

[vagrant@localhost ~]$ cd subversion-1.9.4-1/

[vagrant@localhost subversion-1.9.4-1]$ sh ctlscript.sh restart

* create one folder in desktop ‘old dev’ and checkout the respository.

C:\Users\Hare Rama\Desktop\old dev>svn co http://192.168.33.10:8080/subversion/

Checked out revision 0.

* After that create three folders in repository suppose

🡪 Trunk

🡪 Branch

🡪 Tag

C:\Users\Hare Rama\Desktop\old dev\subversion>svn mkdir trunk branch tag

A trunk

A branch

A tag

C:\Users\Hare Rama\Desktop\old dev\subversion>svn ci -m "folder from old dev"

Adding branch

Adding tag

Adding trunk

* Goto inside the trunk folder and create two folders name it as
* Audio
* video

C:\Users\Hare Rama\Desktop\old dev\subversion>cd trunk

C:\Users\Hare Rama\Desktop\old dev\subversion\trunk>svn mkdir audio video

A audio

A video

C:\Users\Hare Rama\Desktop\old dev\subversion\trunk>svn ci -m "inner folders"

Adding audio

Adding video

* copy the content of ‘trunk’ folder to ‘branch’ folder inside sub folder ‘r1’.

C:\Users\Hare Rama\Desktop\old dev\subversion>svn copy http://192.168.33.10:8080/subversion/trunk/ http://192.168.33.10:8080/subversion/branch/r1 -m "copying"

* create new folder in desktop suppose ‘new dev’ and checkout the ‘r1’ folder.

C:\Users\Hare Rama\Desktop\new dev\subversion>svn co

http://192.168.33.10:8080/subversion/branch/r1/

A r1\audio

A r1\video

* if I open now ‘r1’ folder I will get audio and video folders.
* No I need to create two more folders in ‘r1’ folder as

🡪 Pdf

🡪 image

C:\Users\Hare Rama\Desktop\new dev\subversion>cd r1

C:\Users\Hare Rama\Desktop\new dev\subversion\r1>svn mkdir pdf image

A pdf

A image

C:\Users\Hare Rama\Desktop\new dev\subversion\r1>svn ci -m "new folders"

Adding image

Adding pdf

* Now my requirement is combine the ‘r1’ folder code to ‘trunk’ folder. For that requirement I will go to trunk folder and run the command svn update and svn merge r1 url and next svn ci –m “msg”.

C:\Users\Hare Rama\Desktop\old dev\subversion>cd trunk

C:\Users\Hare Rama\Desktop\old dev\subversion\trunk>svn update

Updating '.':

C:\Users\Hare Rama\Desktop\old dev\subversion\trunk>cd ..

C:\Users\Hare Rama\Desktop\old dev\subversion>svn update

Updating '.':

D pdf

D image

A branch\r1\image

A branch\r1\pdf

C:\Users\Hare Rama\Desktop\old dev\subversion>cd trunk

C:\Users\Hare Rama\Desktop\old dev\subversion\trunk>svn merge <http://192.168.33.10:8080/subversion/branch/r1/>

C:\Users\Hare Rama\Desktop\old dev\subversion\trunk>svn ci -m "merged"

Sending .

Adding image

Adding pdf

………………………………………………………………………………………………………………………………………………………………….

Merging concept commands :

1. New Repository creation:
2. Old Dev:

* Svn mkdir trunk branch tag
* Svn ci –m “ss”
* Cd trunk
* Svn mkdir audio video
* Svn ci –m “ss”
* Cd ..
* Svn copy <http://192.168.33.10:8080/subversion/trunk/> <http://192.168.33.10:8080/subversion/branch/R1> -m “ss”

1. New Dev

* Svn co <http://192.168.33.10:8080/subversion/branch/R1/>
* Cd R1
* Svn mkdir pdf image
* Svn ci –m “s”

1. Open Old Dev Folder and open trunk folder open CMD run the command svn update
2. Svn merge <http://192.168.33.10:8080/subversion/branch/R1/>
3. Svn ci –m “ss”.

………………………………………………………………………………………………………………………………………………………………

Branchin strategy or what is meant by baseline or svn workflow:

For example take three folders in repository like

🡪Trunk

🡪Branch

🡪 Tag

Trunk -----> maintaining the original source code of particular version.

Tag --------> taking the backup of particular version.

Branch -----------------------------

Standard branch

Future branch

Hot fix or temporary branch

Bug fix branch

Standard branch bug fix branch

Ex : Dev 1.0 Ex : dev 1.0#1

Dev 1.1 dev 1.1#2

Dev 1.2 dev 1.2#3

Hot fix branch future branch

Ex : hot fix dev 1.0 ex: future dev 1.0

hot fix dev 1.1 future dev 1.1

temporary dev 1.2 future dev 1.2

SVN backup:

For example create two repositories like “merge” and “merge1”.

* In merge respository ….we need to do backup for that command is

Sh svnadmin dump /home/vagrant/merge > /home/vagrant/rama.bkp

[vagrant@localhost bin]$ sh svnadmin dump /home/vagrant/merge > /home/vagrant/rama.bkp

\* Dumped revision 0.

\* Dumped revision 1.

\* Dumped revision 2.

\* Dumped revision 3.

\* Dumped revision 4.

* Now delete merge repository because everything is dumped.
* Now to load the dumped files in next repository like merge1 we have command is

Sh svnadmin load /home/vagrant/merge > /home/vagrant/rama.bkp

[vagrant@localhost bin]$ sh svnadmin load /home/vagrant/merge1 < /home/vagrant/rama.bkp

<<< Started new transaction, based on original revision 1

\* editing path : test ... done.

------- Committed revision 1 >>>

<<< Started new transaction, based on original revision 2

\* editing path : test2 ... done.

------- Committed revision 2 >>>

<<< Started new transaction, based on original revision 3

\* editing path : test3 ... done.

------- Committed revision 3 >>>

<<< Started new transaction, based on original revision 4

\* editing path : test/java (2).txt ... done.

\* editing path : test/java.txt ... done.

------- Committed revision 4 >>>

* Now everything backedup into new repository merge1.

SVN ACCESS concept:

* First create two file and name it as ---

🡪svn\_users

🡪svn\_access

[vagrant@localhost ~]$ touch svn\_users svn\_access

[vagrant@localhost ~]$ ls

bitnami-subversion-1.9.4-1-linux-x64-installer.run hello.sh merge sample.sh subversion-1.9.4-1 svn\_access svn\_users test

* Get into apache/bin

[vagrant@localhost ~]$ cd subversion-1.9.4-1/apache2/bin/

* Create username by using cmd --- sh htpasswd -c /home/vagrant/svn\_users username

[vagrant@localhost bin]$ sh htpasswd -c /home/vagrant/svn\_users hareram

New password:

Re-type new password:

Adding password for user hareram

[vagrant@localhost bin]$ pwd

/home/vagrant/subversion-1.9.4-1/apache2/bin

[vagrant@localhost bin]$ cd ../../..

[vagrant@localhost ~]$ cat svn\_users

hareram:$apr1$TiryLHU5$kTfDN8fMyyainGtaclqwP1

* Get back home location and edit svn access and write the following data in that

[groups]

[subversion:/]

username=rw

(why we are writing it means to give read and write permission for the user to access)

[vagrant@localhost ~]$ vi svn\_access

[vagrant@localhost ~]$ cat < svn\_access

[groups]

[subversion:/]

hareram=rw

* After that saving the content in svn\_access. Now get into httpd.conf and enter the following data init

AuthType Basic

AuthName "Subversion Project1 repository"

AuthUserFile /home/vagrant/svn\_users

Require valid-user

AuthzSVNAccessFile /home/vagrant/svn\_access

[vagrant@localhost ~]$ vi /home/vagrant/subversion-1.9.4-1/apache2/conf/httpd.conf

[vagrant@localhost ~]$ cd subversion-1.9.4-1/

[vagrant@localhost subversion-1.9.4-1]$ sh ctlscript.sh restart

[vagrant@localhost subversion-1.9.4-1]$

* So finally username and passwd is created for the repository now reload the remote server then it ask to enter username and passwd.

SVN Hooks:

* Pre-commit
* Post-commit

ANT

ANT 🡪 Another Neat Tool

RELEASE

DEPLOYMENT

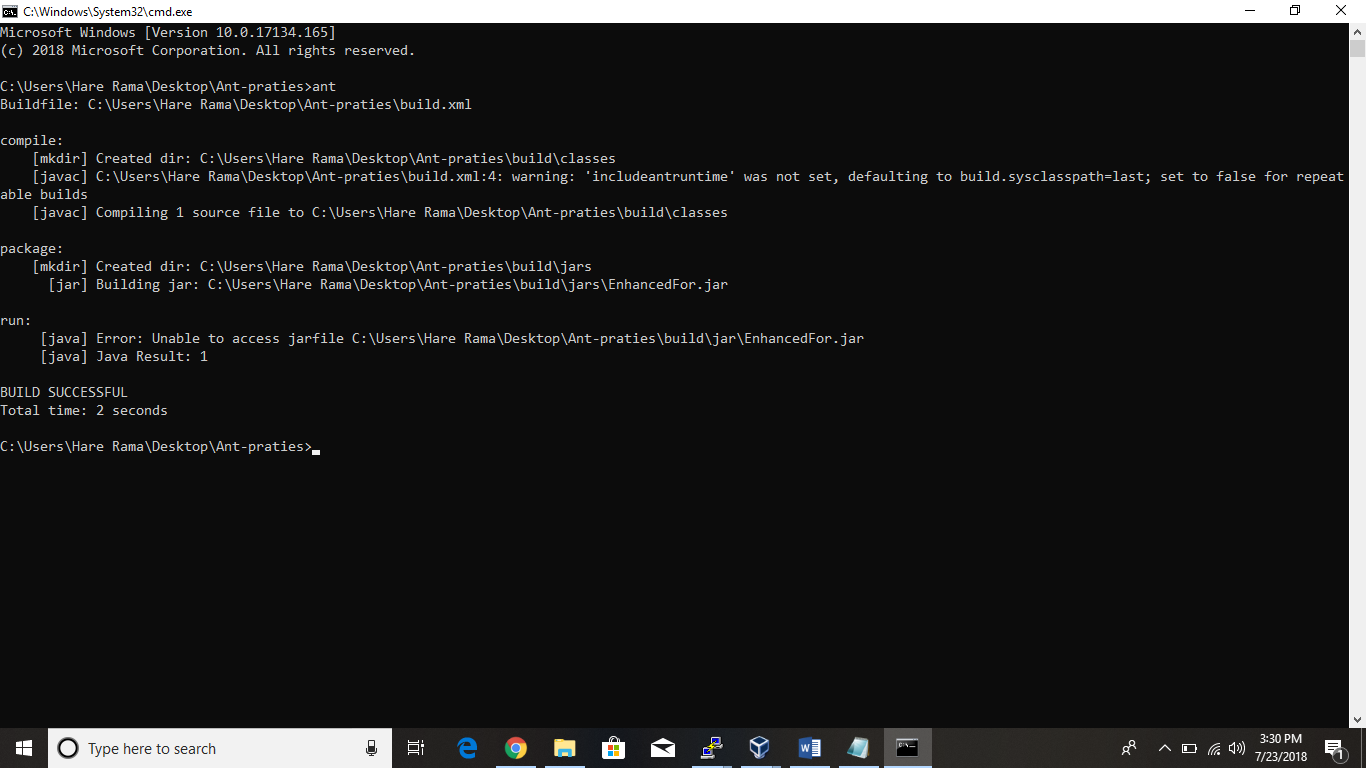
------> ----> ----->

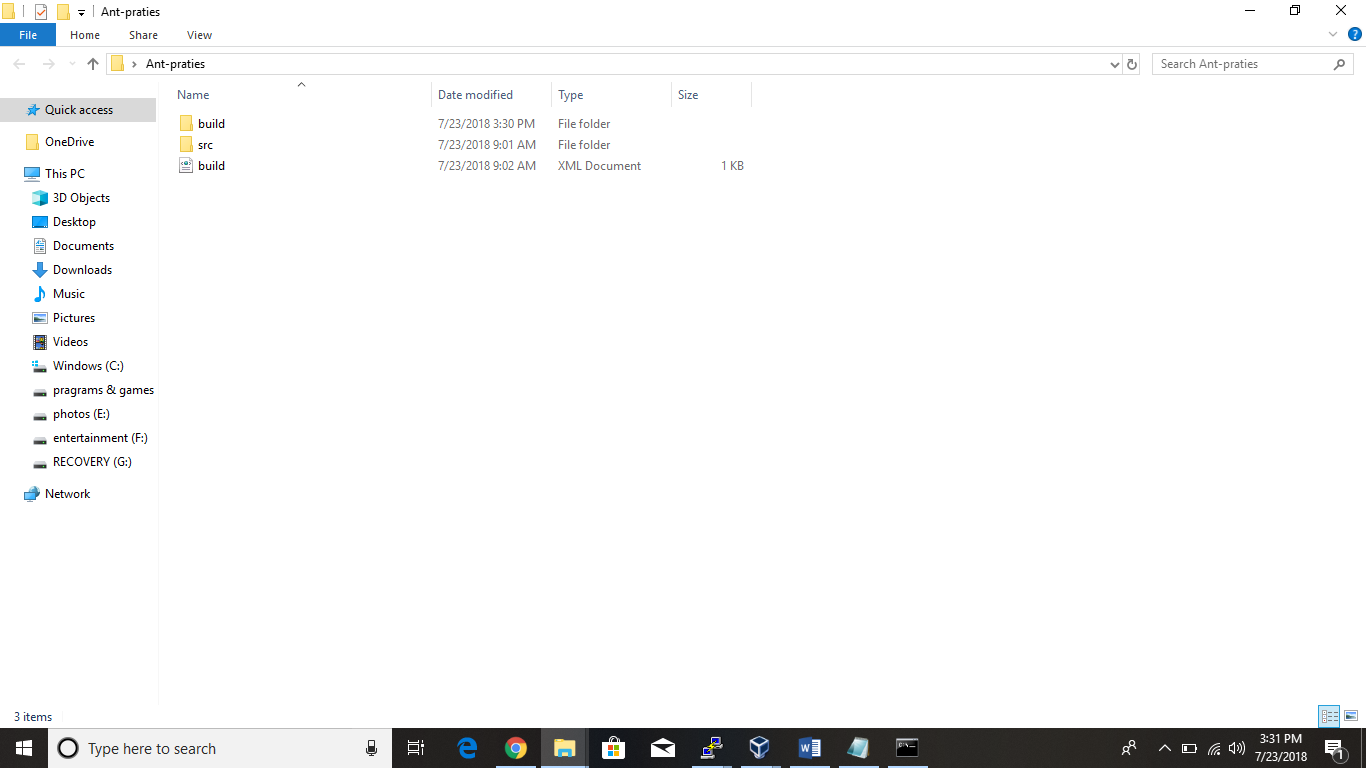
BUILD

In build it will compile, package and run.

Process to build the java file:

* First create folder in desktop and name it ex: ‘ant practice’.
* Inside the folder create one more folder name it as ‘src’.
* Get inside the ‘src’ folder and save the java code by ‘.java’ format.
* Come back to main folder (ant practice) and create one text doc and save it as ‘build.xml’ and it is main to build java file.
* After open the cmd prompt in ant practice folder and enter “ant” cmd and finally it builds.
* After build we will get one folder “build” in that we have “.class files” and “jar” files.





JENKINS

Mainly we go with 4 type of Jenkins versions. Those are

Version- 1.565.1

Version- 1.658

Version- 1.602

Version- 2.134

To install java :

Cmd: sudo yum install java-1.7.0-openjdk –y

How to start the Jenkins in windows:

Cmd: java –jar Jenkins.war

How to change port number:

Actual port number for subversion is 8080 and Jenkins default port number is 8080. So we need to change sub version port number for example :8081.

…………go to below location ………

🡪 cd subversion-1.9.4.1/apache2/conf/httpd.conf

🡪don’t click on shift + G

🡪now just run the command ---- /Listen

🡪next edit the port number 8080 to 8081.

🡪save the file.

🡪now restart the svn.

Master/slave configuration (or) Build machine configuration (or) Node configuration (or) Label configuration:

Steps in Jenkins:

1. Code commit completed
2. Build phase
3. Junit test cases
4. Code quality
5. Release phase
6. Upload package to server
7. Deployment

Slave - To distribute the load.

How to configure slaves:

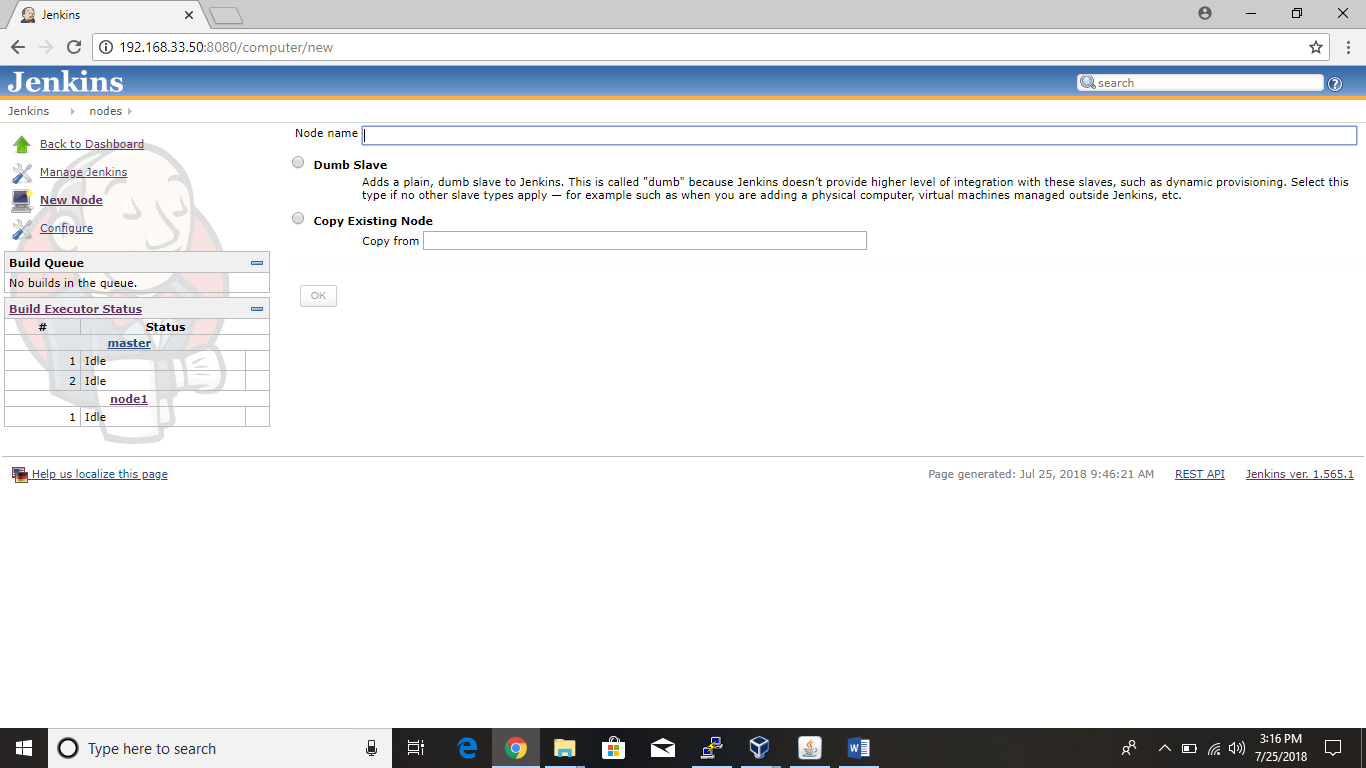
Creating slaves:

Manage Jenkins 🡪 manage node 🡪 new node 🡪

Name : node1

Select :Dump slave

Click ok



After that another page comes to under the details like

Name: node1

Description : ex: for project requirement

Remote fs root :

Create folder in desktop name it as Jenkins logs and copy the path set in remote fs root

Launch method : select java web start

Tool location we need to set

Before setting tool location. First we need add tool names in the configure system tab

Manage Jenkins

Configure system

Click on JDK

Add JDK

Uncheck the option install automatically

Name : JAVA\_HOME

Click on ant

Add Ant

Uncheck the install automatically

Name : Ant\_Home

Save

After configuring everything in configure system come back to creation of node process

Adding tools location to slave

Manage Jenkins

Manage node

Configure

Tool location

Click on Add

Select : JAVA\_HOME

Select : Ant\_Home

Copy the java and ant location from local and set in slave don’t copy the bin location

Save

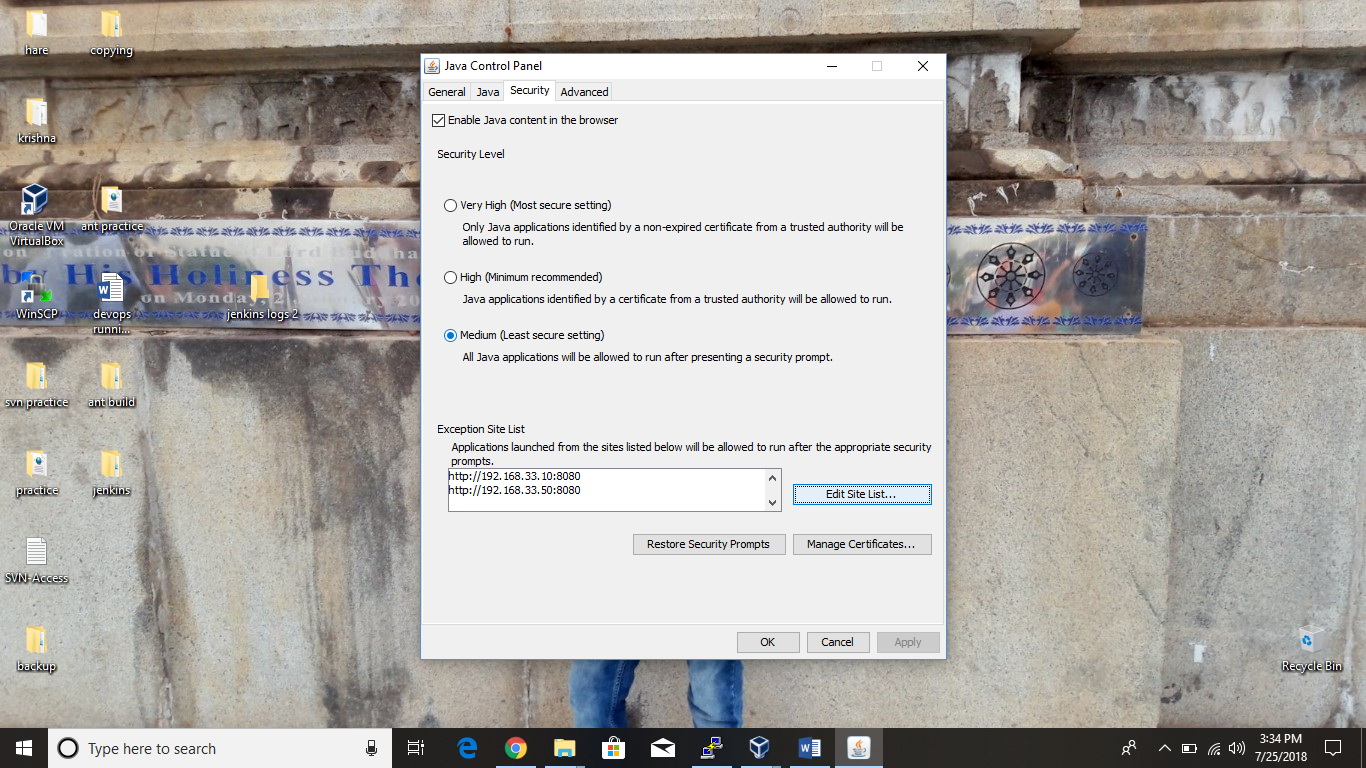
Launch

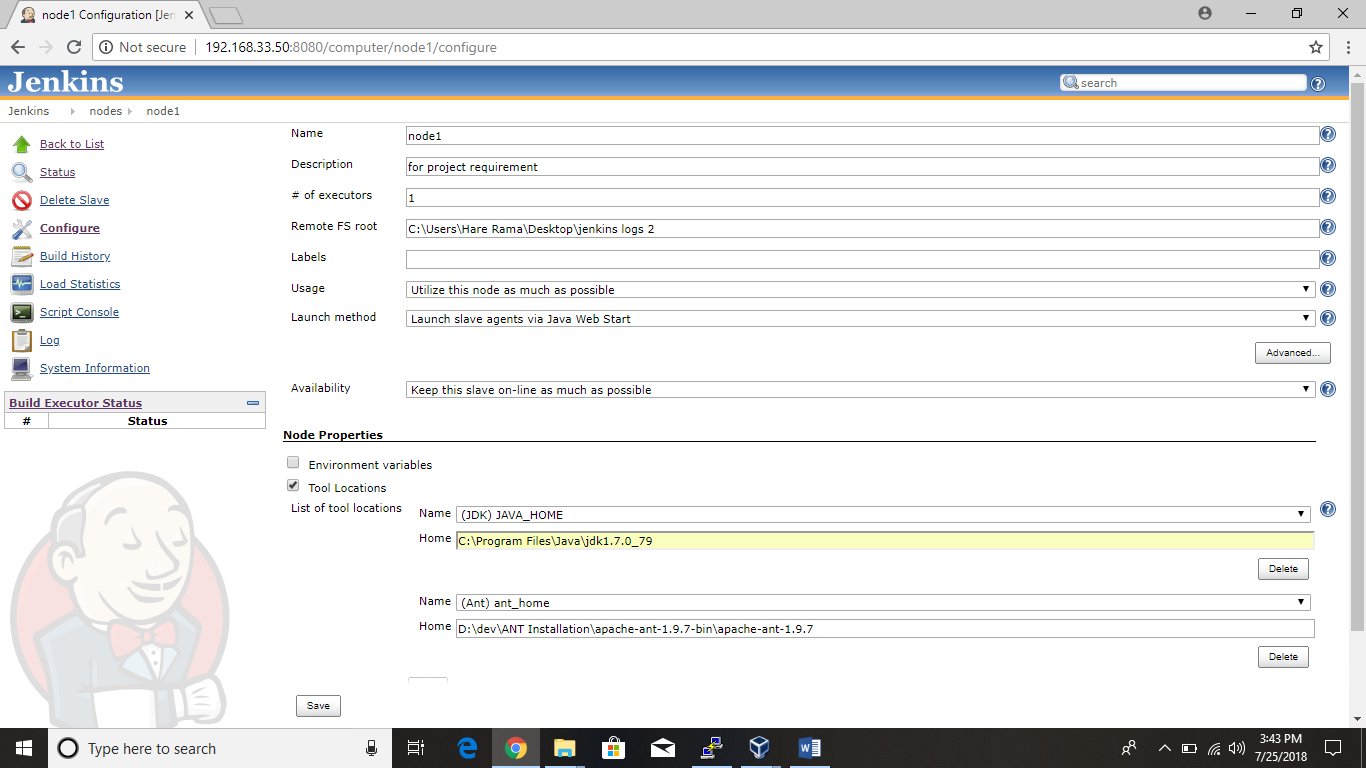
Keep

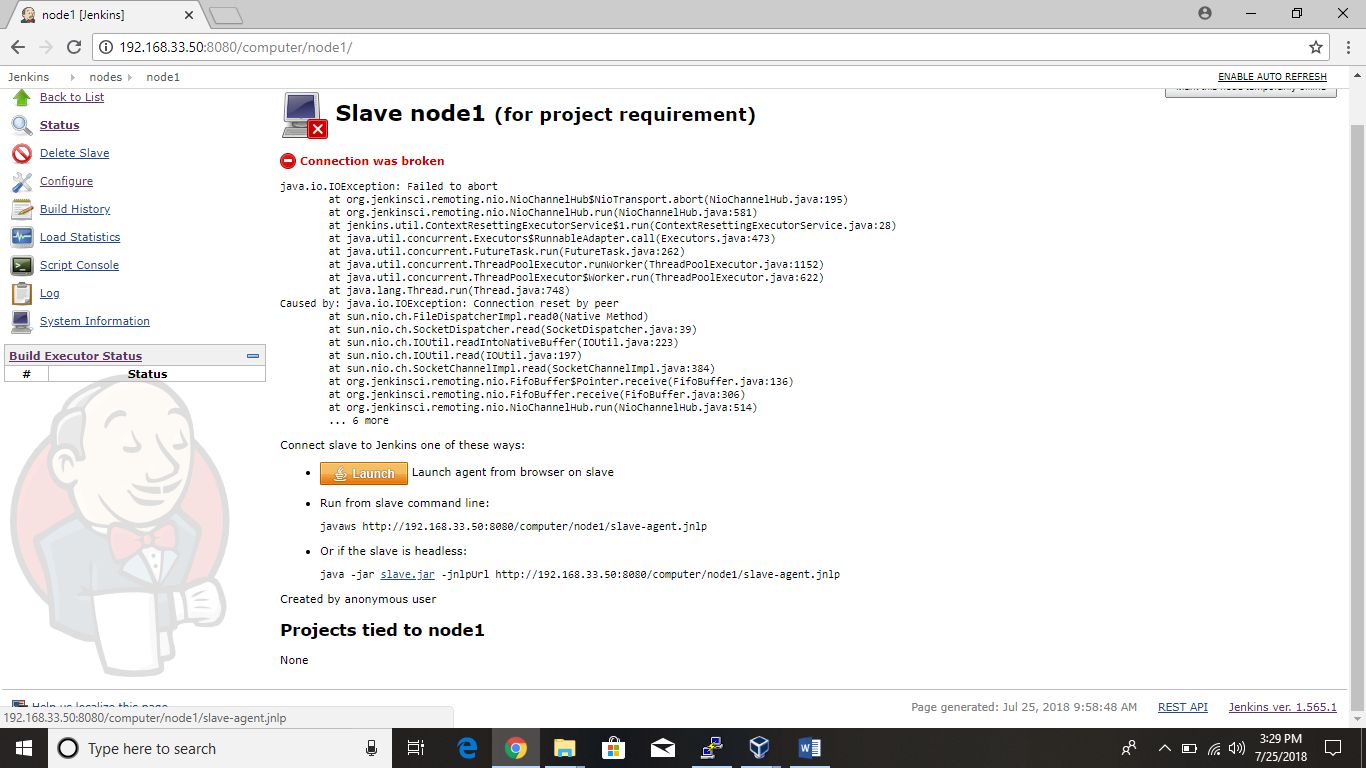
Start.

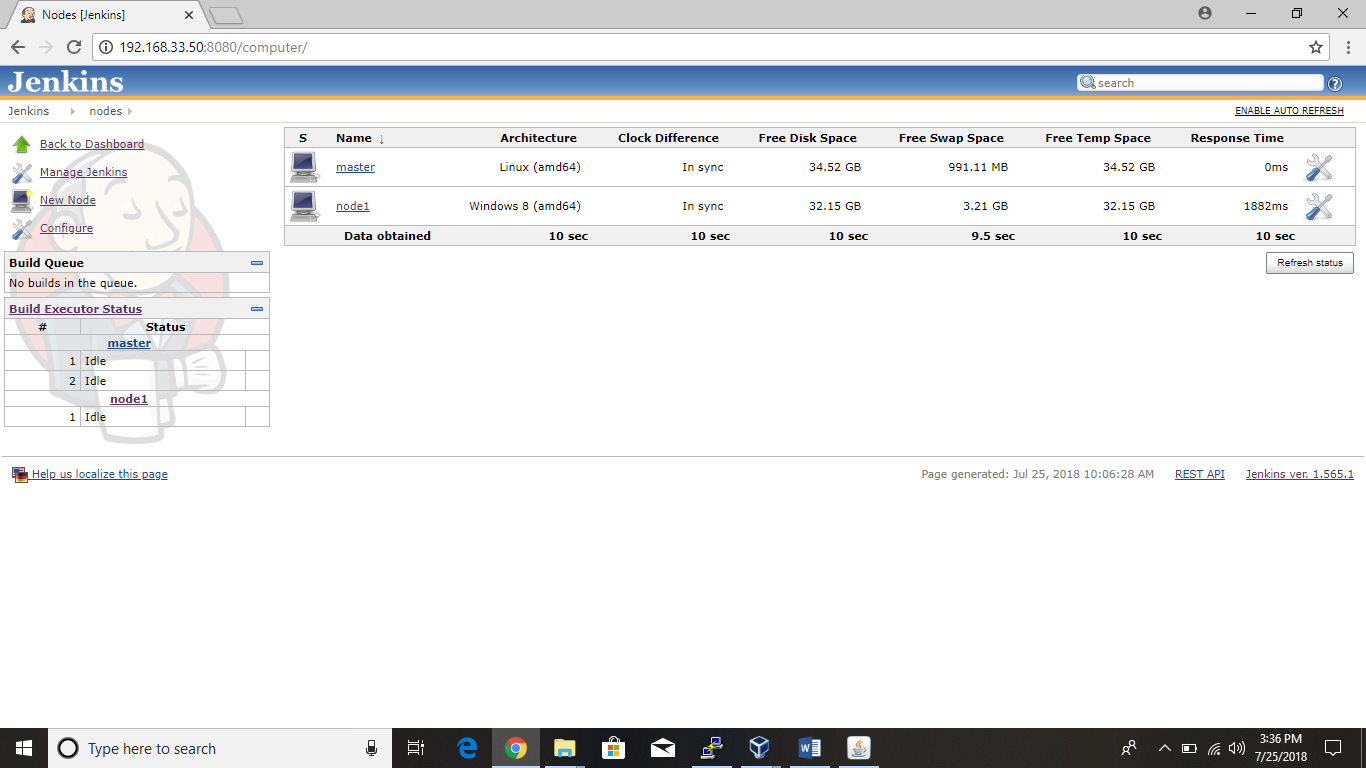
* In this process we have to configure java in our pc . search for java in menu and select “configure java” .
* Next select security option
* Keep the medium secure in security level.
* Next edit site list and add ip address of the Jenkins ex :

http://192.168.33.50:8080









How to create jobs in Jenkins:

* First login into the Jenkins. After login you will see the page



click on new item

Enter the item name :

Here the name should be in the format of

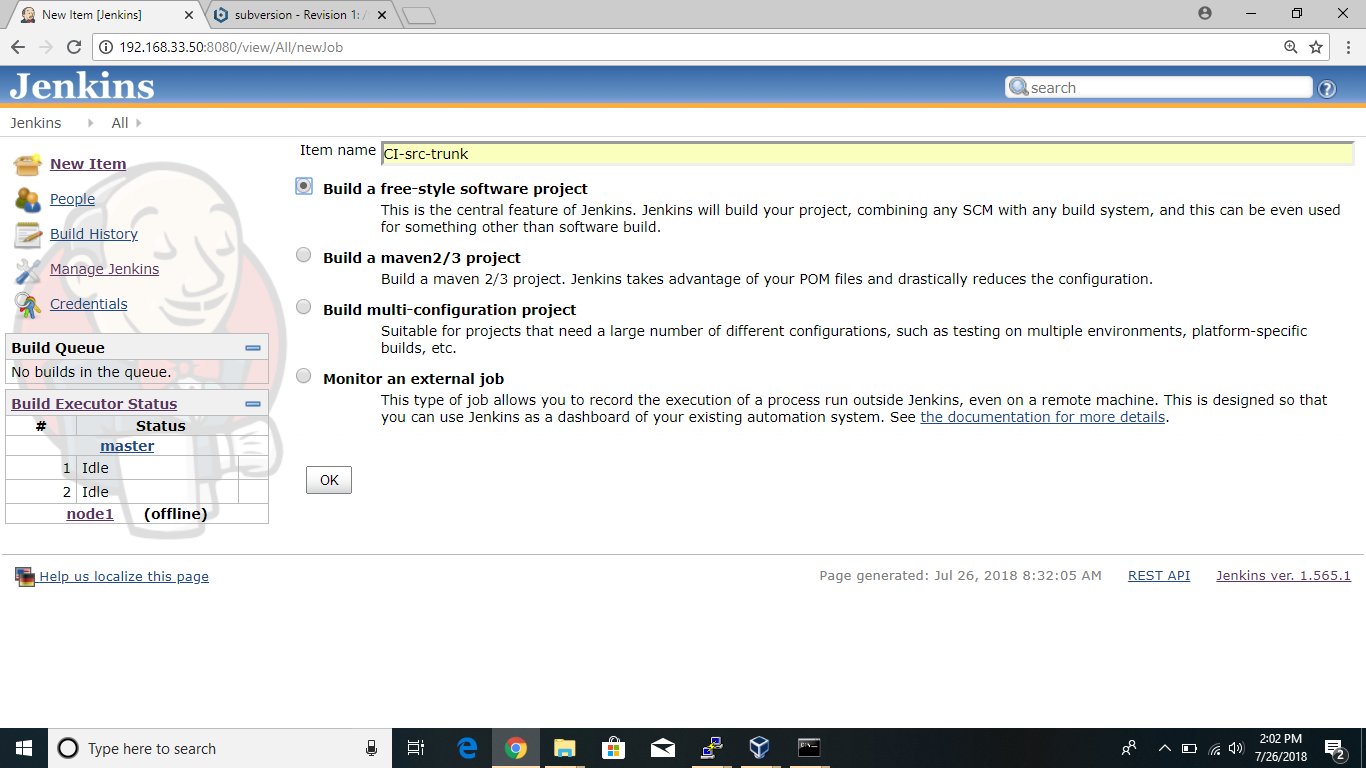
CI-Jobname-folder name

Ex: CI-CITI-Dev1.0

Select : Build a free style software project

(for the fresh project we need select this one)

Click ok



Now clicking ok you have entered into configuration page

In this enable the option restrict where this project can be run

Label expression : select node1

Come down to source code management

Select SUBVERSION

Repository url : provide url of the java code which is in svn repository

Ex : <http://192.168.33.50:8081/subversion/trunk>

Come down to Build

Select invoke ant

Ant version : ant\_home

Here ant\_home automatically because previously we have added ant\_home in manage Jenkins 🡪 configure system.

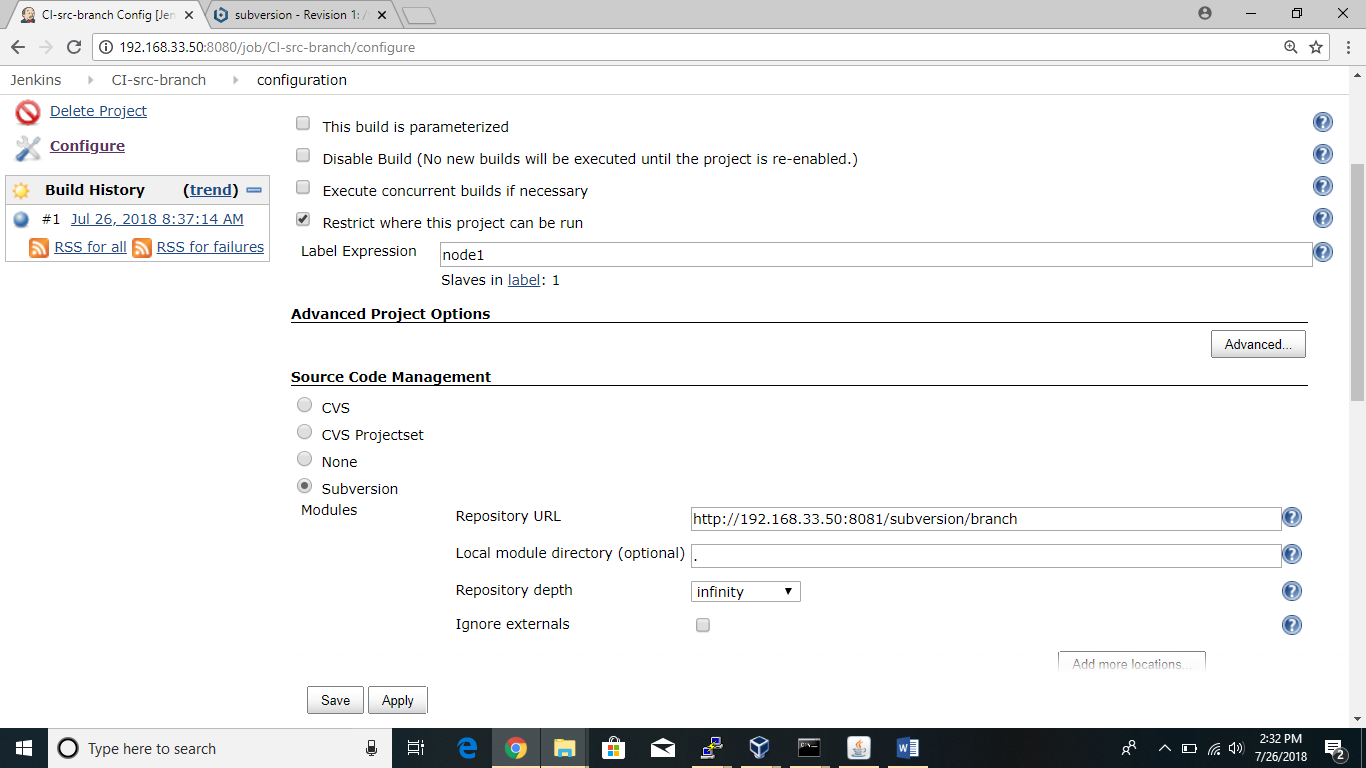
Come down to post-build actions

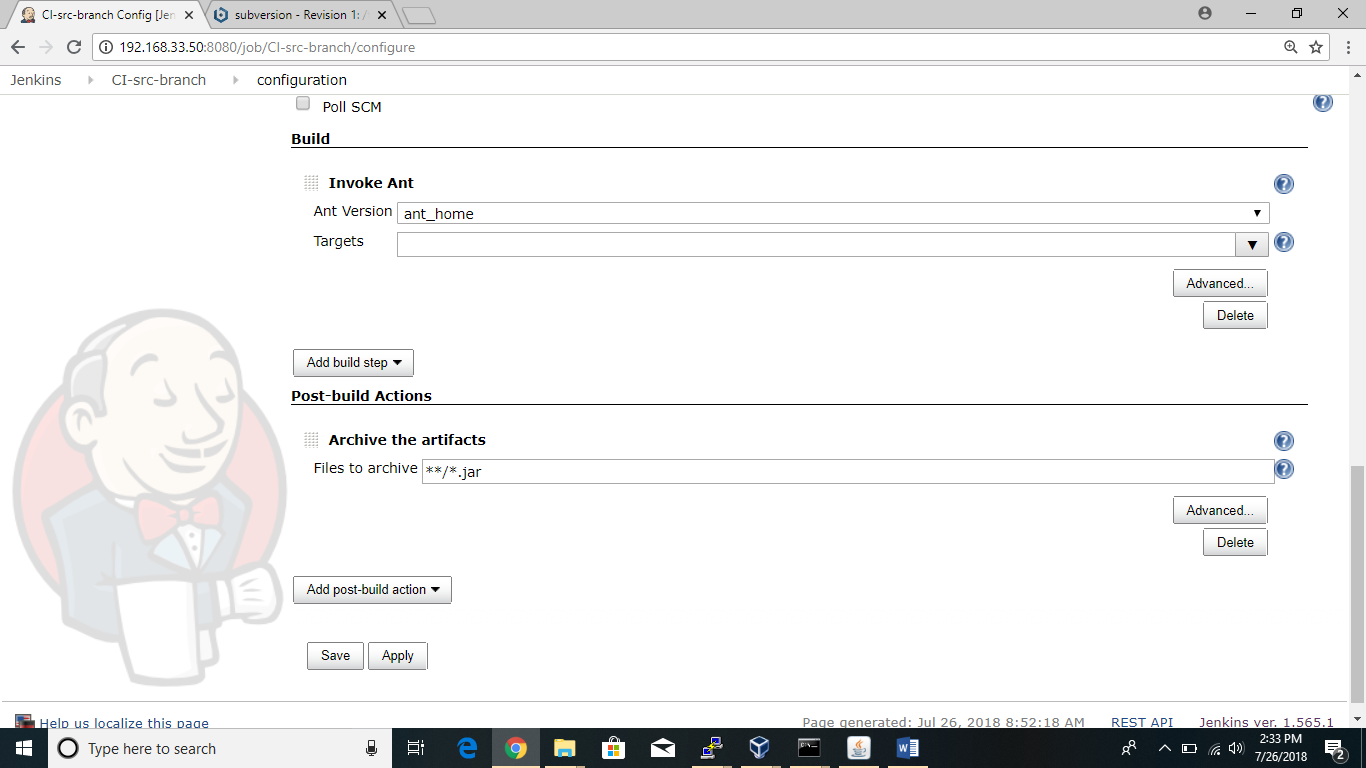
Select archive the artifacts

Files to archive : \*\*/\*

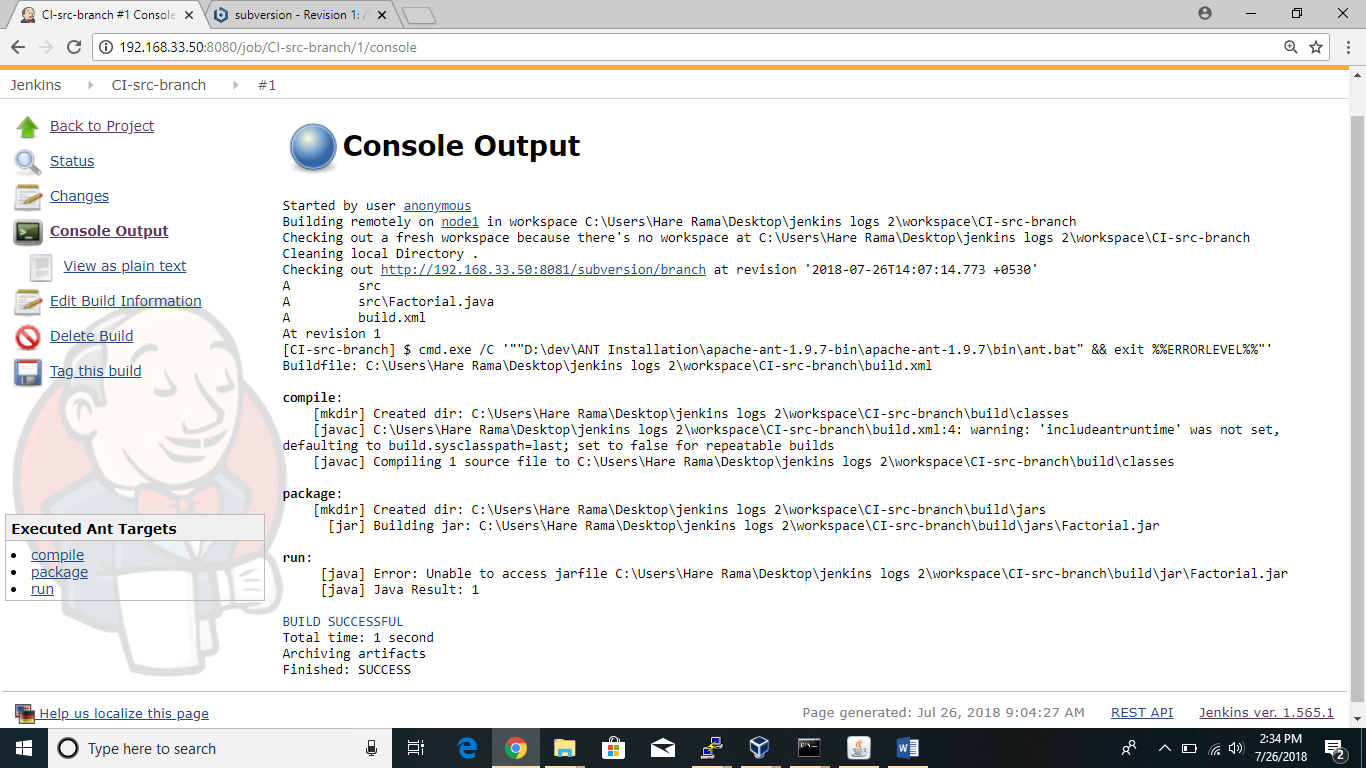
Save

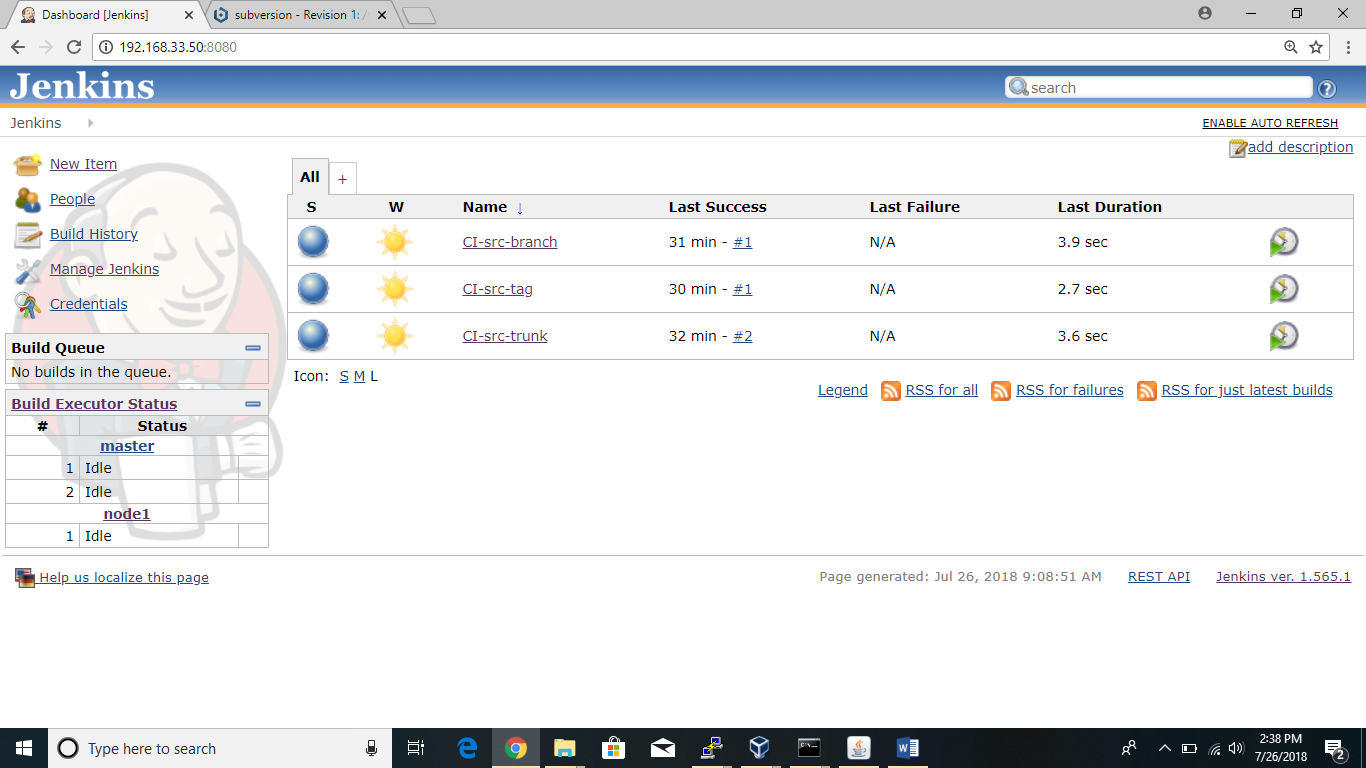
After that select the option BUILD NOW.











Finally build is success that we can see in console output. If any build error we can identify that error in console output. The data like .class files and jar files are stored in workspace and same logs will be stored in local folder. For example in my pc stored in the folder of Jenkins logs.

Imp concepts that we need to remember while creating job:

* Discard old builds
* Poll SCM
* Build periodically
* Quiet period
* Crontab format
* This build is paramterised

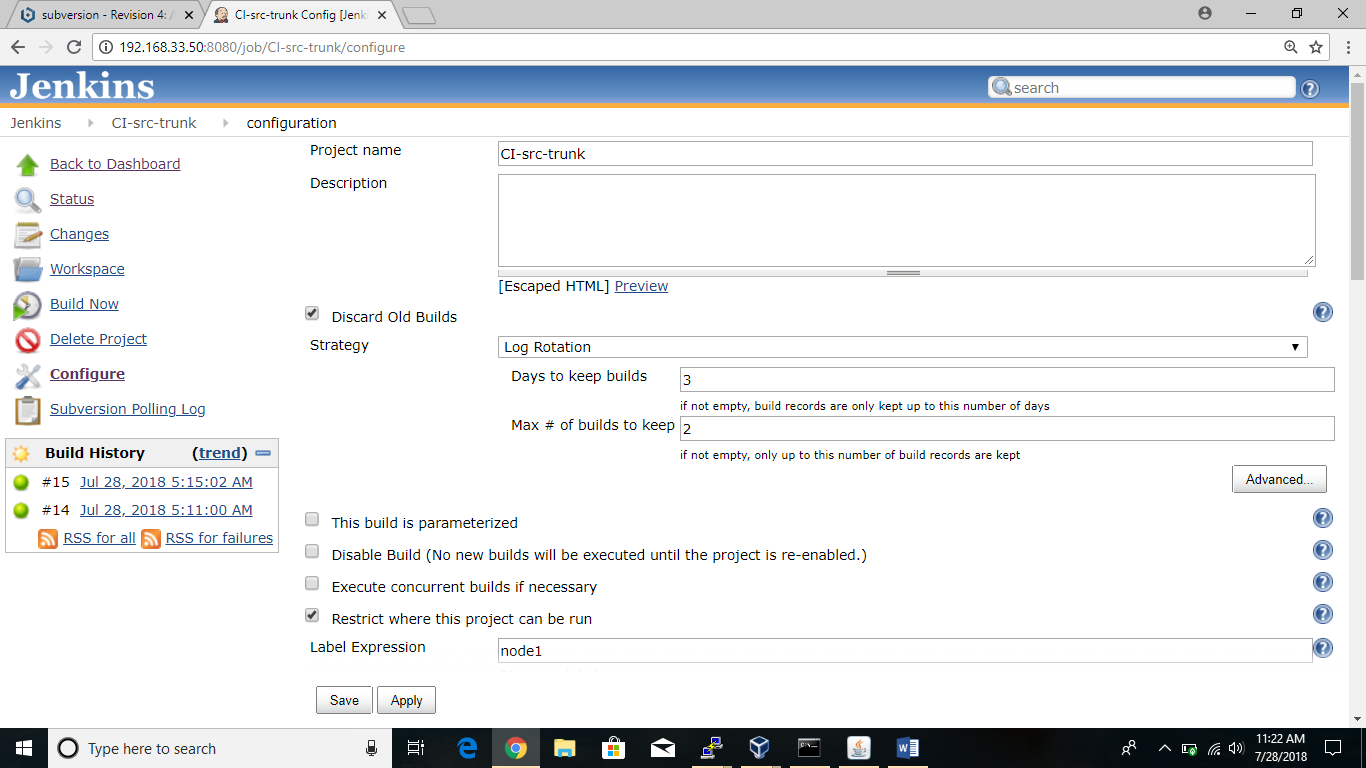
Discard old builds:

* This option is mainly used to keep only recent builds and for certain days.
* For example go to project CI-src-trunk 🡪 configure
* In configuration part enable the option discard old builds.
* In that

Strategy : log rotation

Days to keep builds : enter no of days

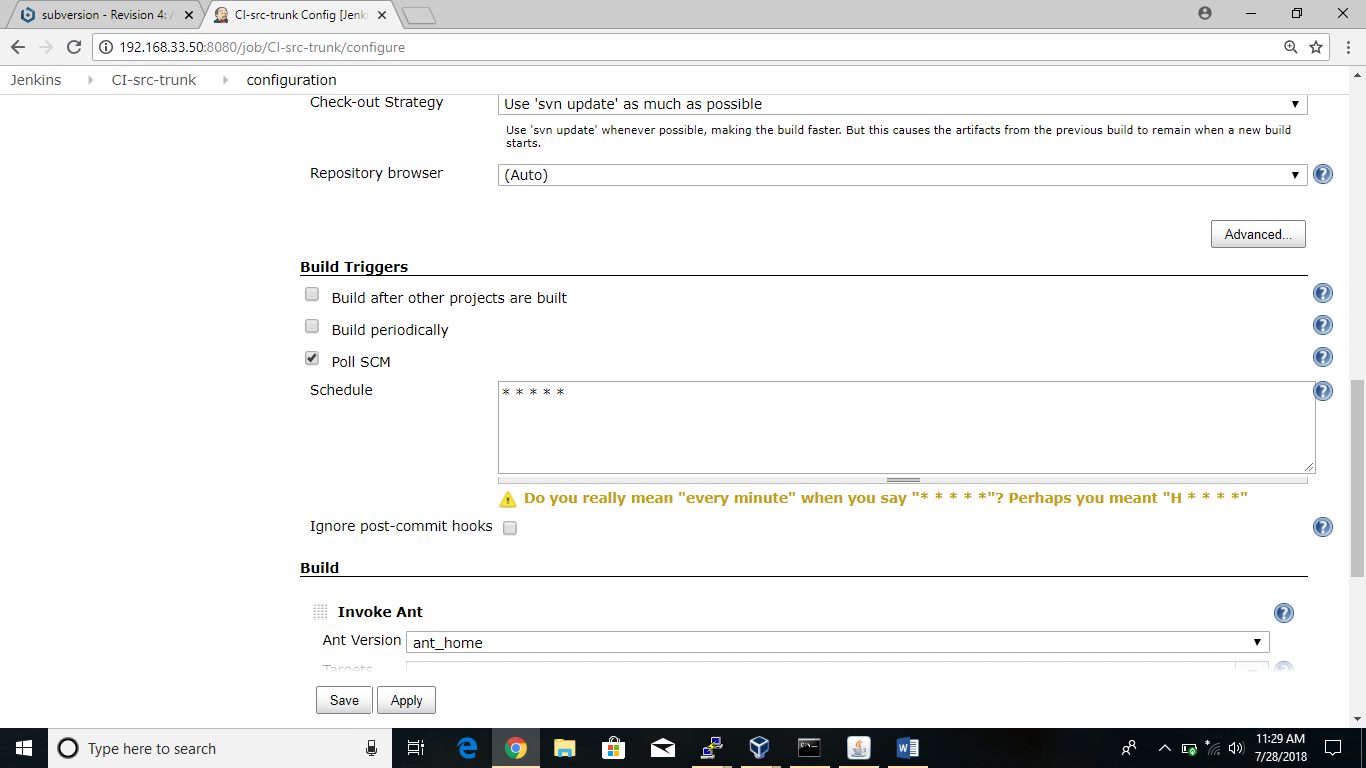
Max builds to keep : enter how many builds u need to keep.



Save the project and start build and there you can observe only recent 2 builds will be displayed.

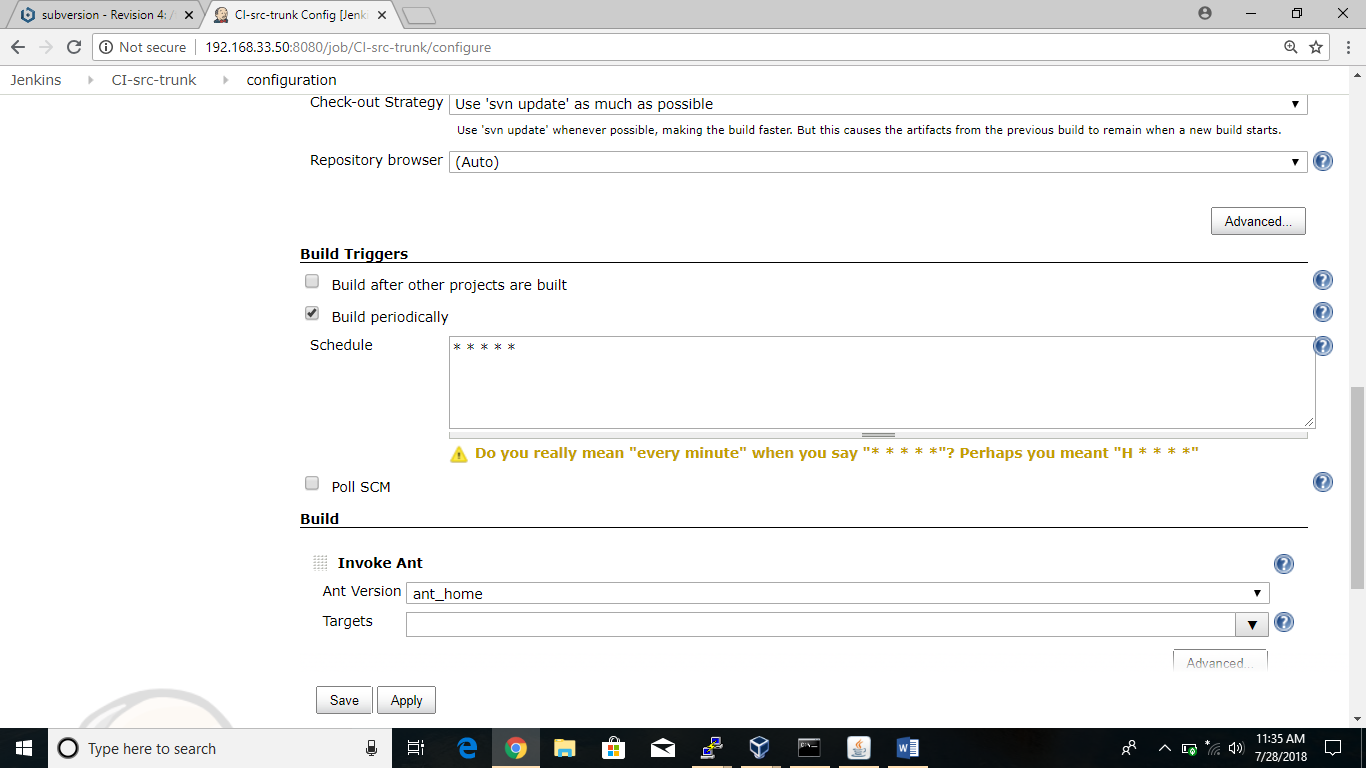
Poll SCM:

* This poll scm is used whenever developer do changes to the code and checkin to the rerpository then automatically detects the changes and starts build in Jenkins.
* First we need to enable the poll scm in project configure 🡪 Build triggers.
* Schedule the time suppose \* \* \* \* \* .so every minute it checks the code and if any changes are done the builds the code.



Build periodically:

* It means arranging the time to build. Based on that time project is builded. Whether code changes done or not it builds for the specific time. That time is given by us.
* First we need to enable the build periodically option in project configure 🡪 Build triggers
* And schedule the time default time is \* \* \* \* \* .that means I arranged time for every minutes. We can arrange time based on requirement.



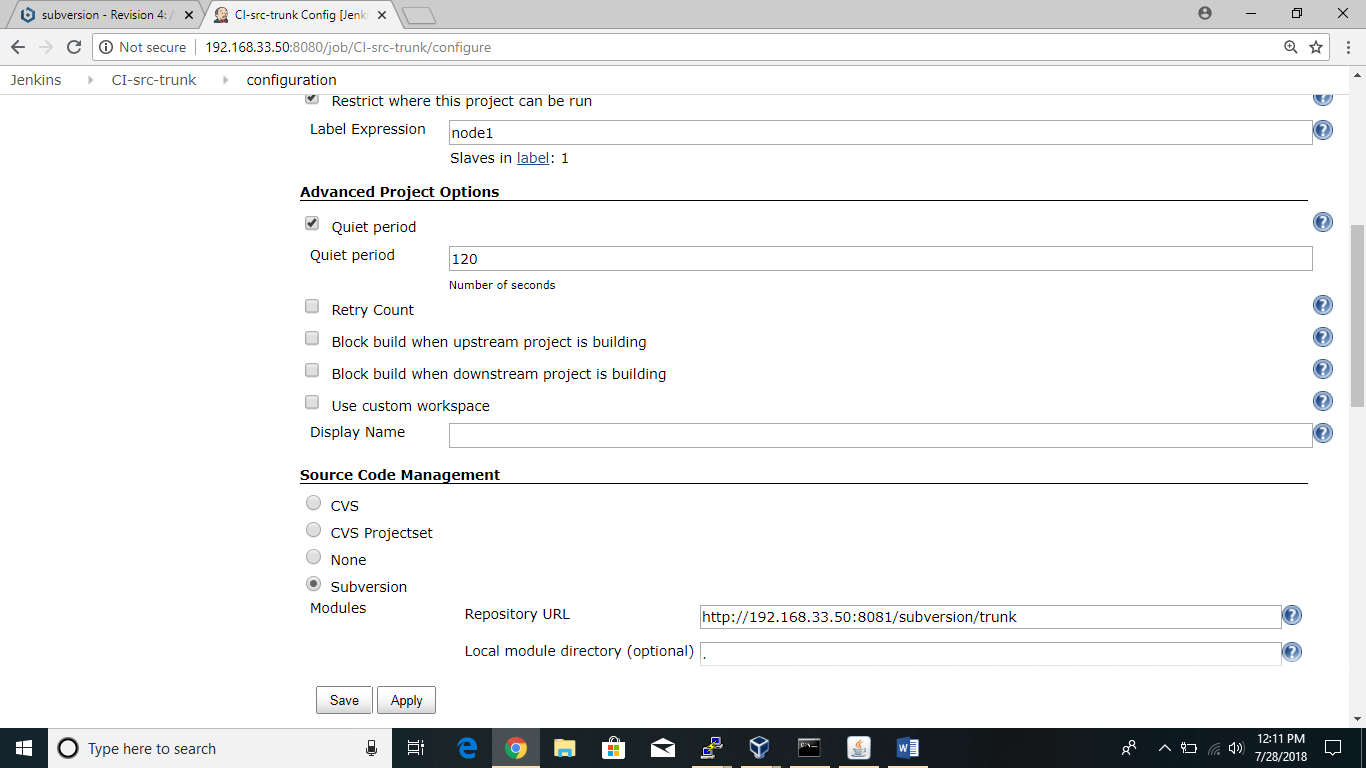
Based on that time Jenkins automatically builds the project.

Quiet period:

* It means build is being hold for a certain seconds. We have to enable only in seconds it won’t accept in minutes.
* For example in poll scm you have arranged time schedule for certain time to take automatic build when code change done but after getting code also we can hold build for some time for doing changes in code. This is possible only through quiet period.

It is available at project 🡪 configure 🡪 advanced project options 🡪

Enable quiet period 🡪 provide the time in sec.





See in the left countdown started . I holded build for 160 sec.

This build is parameterized:

There are different parameters available in this.

String parameter

Name : ex : CI-src-trunk

Default value : ex: CI-src-trunk

Description : <font size= “q” jobname starts with “ci-src”>

Text parameter

Name : ip address

Dedault value : ex:192.168.33.12,192.168.33.13

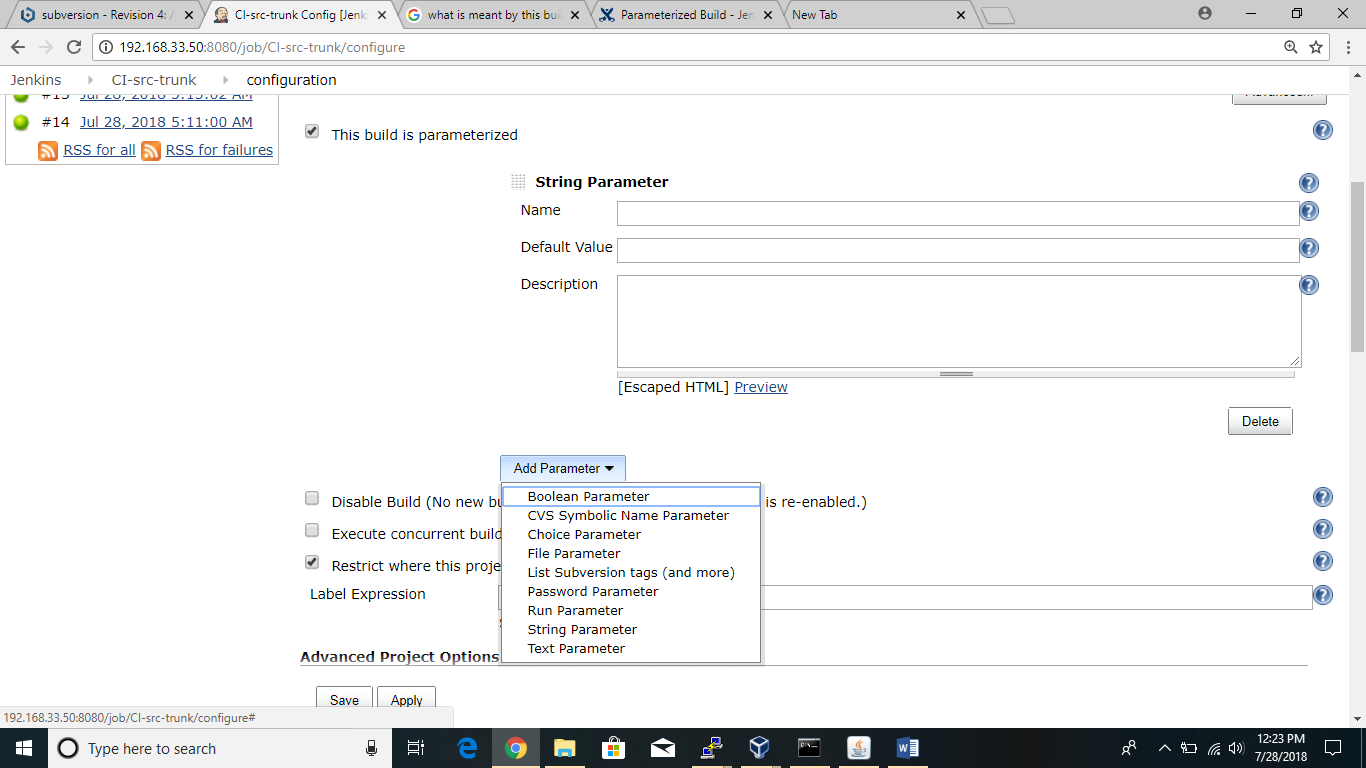
Description : <font size =”2” separated by comma”>

Text parameter

Name : deploy

Default value : yes

Description : <font size =”2” separated by comma”>



Crontab format:

**\* \* \* \* \* \***

| | | | | |

| | | | | +-- Year (range: 1900-3000)

| | | | +---- Day of the Week (range: 1-7, 1 standing for Monday)

| | | +------ Month of the Year (range: 1-12)

| | +-------- Day of the Month (range: 1-31)

| +---------- Hour (range: 0-23)

+------------ Minute (range: 0-59)

* **Minutes** specified as a number from 0 to 59.
* **Hours** specified as numbers from 0 to 23.
* **Days of the month**, specified as numbers from 1 to 31.
* **Months** specified as numbers from 1 to 12.
* **Days of the week**, specified as numbers from 0 to 7, with Sunday represented as either/both 0 and 7.

<Minute> <Hour> <Day\_of\_the\_Month> <Month\_of\_the\_Year> <Day\_of\_the\_Week>

<Minute> <Hour> <Day\_of\_the\_Month> <Month\_of\_the\_Year> <Day\_of\_the\_Week> <Year>

Examples:

\* \* \* \* \* \* Each minute

59 23 31 12 5 \* One minute before the end of year if the last day of the year is Friday

59 23 31 DEC Fri \* Same as above (different notation)

45 17 7 6 \* \* Every year, on June 7th at 17:45

Some of the important plugins that used in Jenkins:

Plugin means changing the functionality of your Jenkins. Is available to enhance the power and usability.

* Green balls
* Git
* Docker
* Rhn push
* Docker build and publisher
* Sonarqube
* Rundeck
* Confluence publisher
* Configuration metrix
* Pipeline

These are important plugins that used in Jenkins.

Plugins can be installed in two ways:

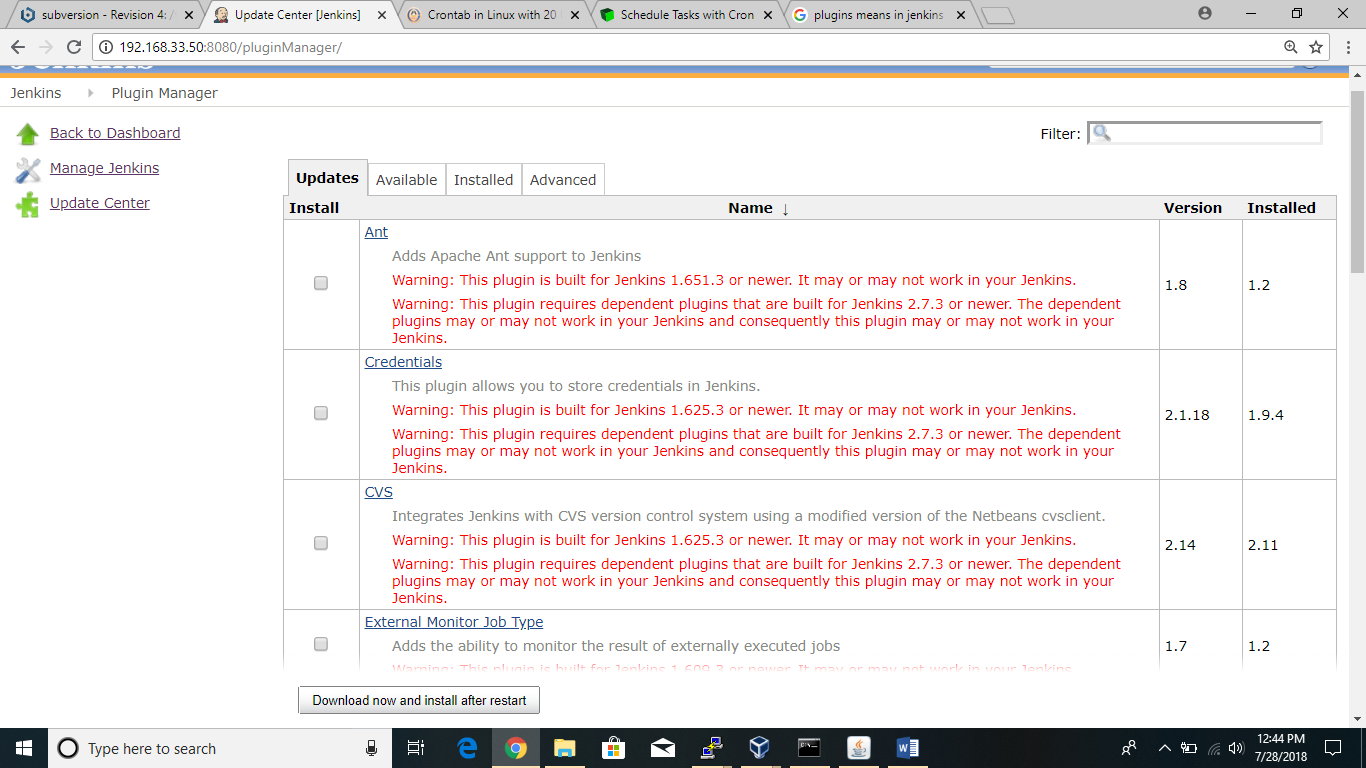
* Manual
* Automation
* For the manual process we need download the plugins from the internet. And the format is in “.hpi file” and upload in Jenkins.

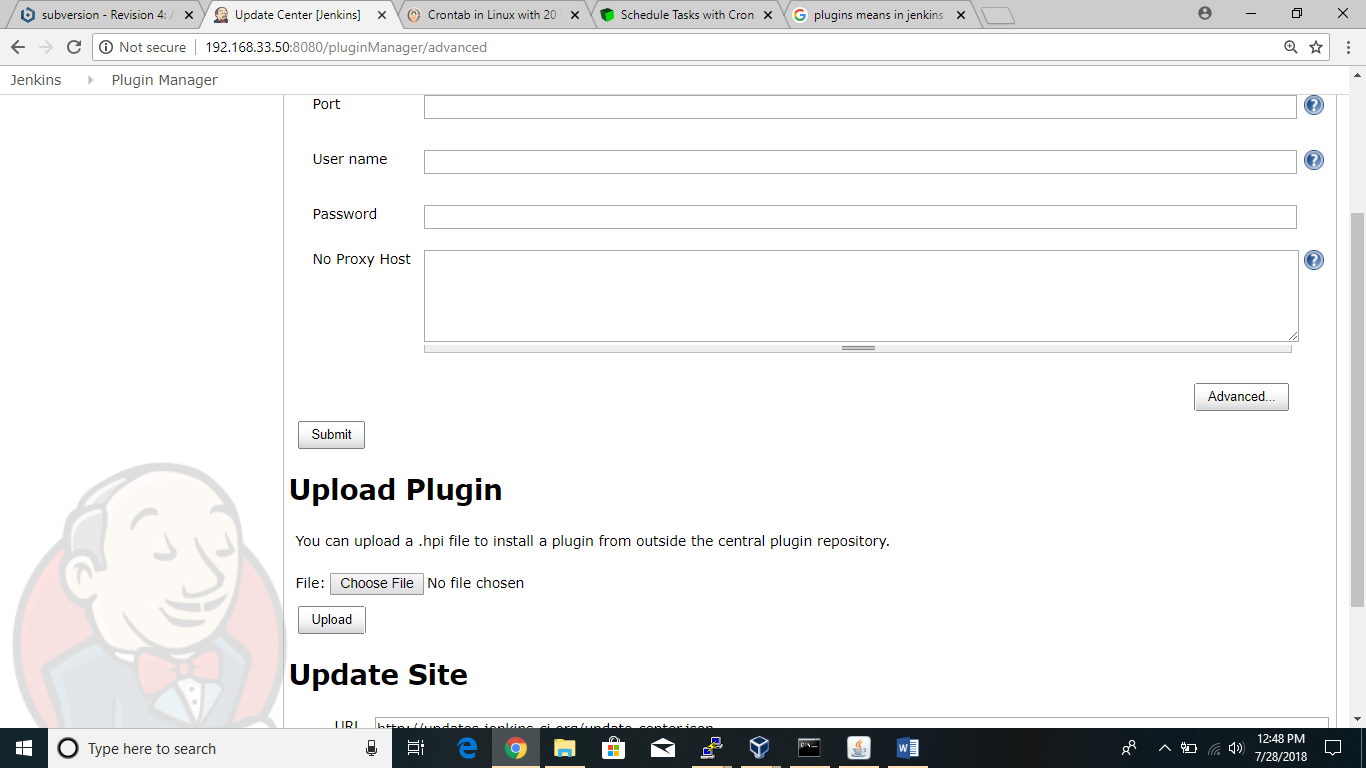
After downloading the “.hpi file” we need upload in Jenkins.

Manage Jenkins 🡪 manage plugins 🡪 advanced

Upload option will be there we need to select and install the plugin.

* For the automation process in Jenkins we have divide into four concepts
* Update 🡪 update the existing plugin.
* Available 🡪 currently available plugins in market.
* Installed 🡪 already installed plugins in your machines.
* Advanced🡪 new plugin installation.





**MAVEN**

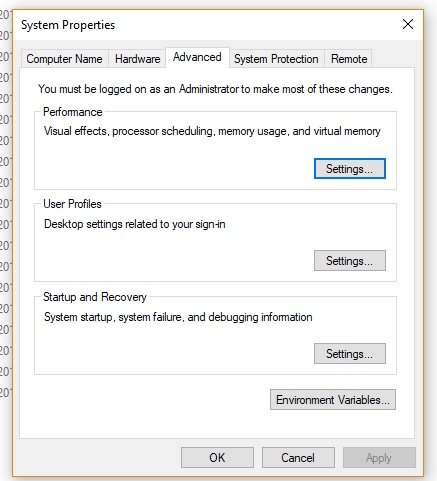
**Maven** is a build automation tool used primarily for Java projects.

Difference between ANT and MAVEN :

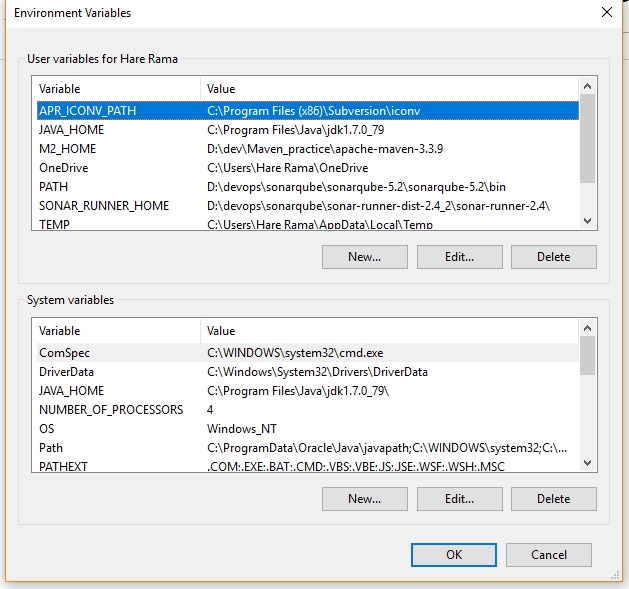
|  |  |
| --- | --- |
| ANT   * ANT is a build tool. Using ANT we can only build the code. * Here we will write the build.xml. * No plugins are required. * Here no repositories. * Here no life cycle. | MAVEN   * We can build the code and we can release the code. * Here we will write the pom.xml. * Maven will compatible for plugins. * Here we have repository. * Here we have the life cycle. |

In maven to build we require pom.xml file but in ANT we use build.xml.

* Before to start the build we need to provide path in environment varibales in our system.
* For that my pc 🡪 right click and select properties🡪advanced system settings.
* Now we will get system properties ---> select environment variables.



In environtmant variables we will have user variables and system varaibales like



* In user variables first we will set the java path. Select New
* Now it ask to enter

**Variable name : JAVA\_HOME**

**Variable value : C:\Program Files\Java\jdk1.7.0\_79**

* Now again select New and now provide path for maven

**Variable name: M2\_HOME**

**Variable value: D:\dev\Maven\_practice\apache-maven-3.3.9**

* Now come to system varaibale. Select new and it ask to enter details again we need to provide java path same as above in user variable.

**Variable name : JAVA\_HOME**

**Variable value : C:\Program Files\Java\jdk1.7.0\_79**

* After setting java path again we need to set maven path in path. Select path in that and enter the maven path link.

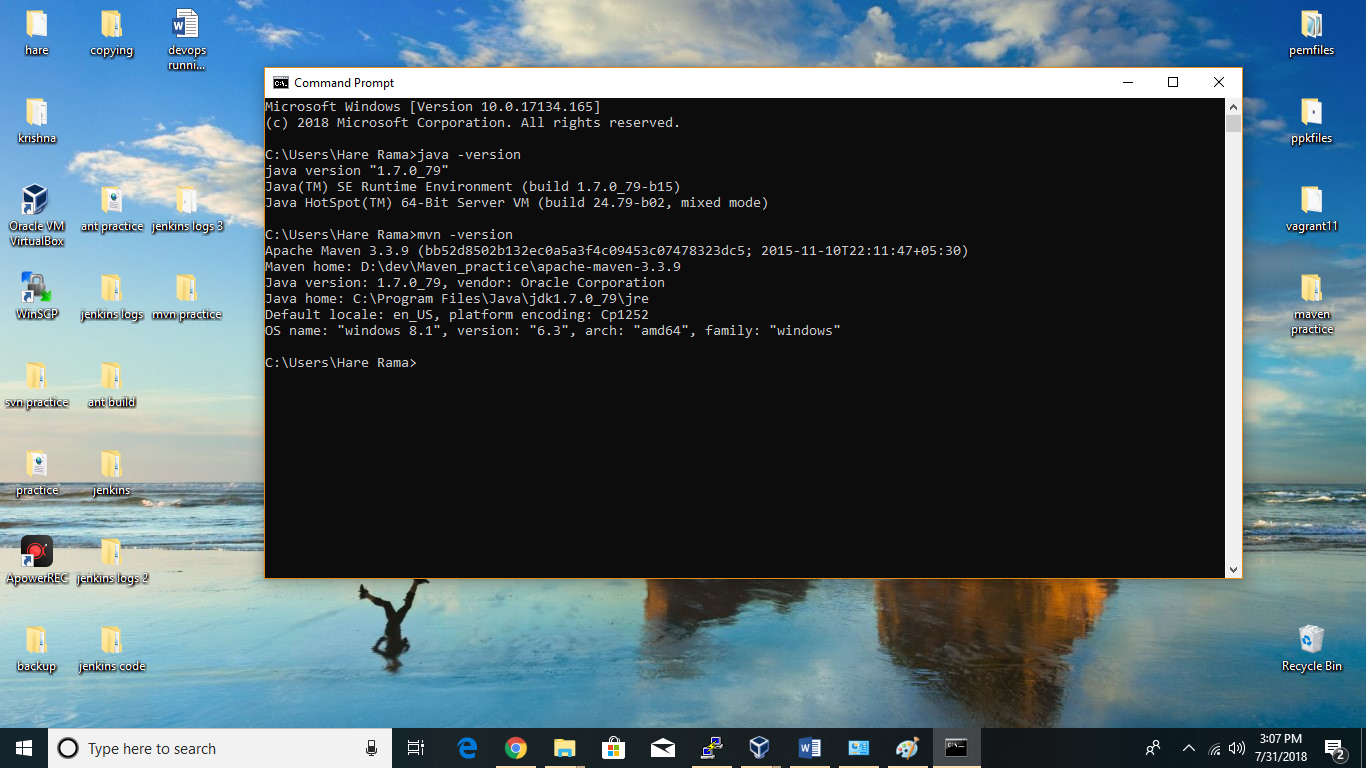
**D:\dev\Maven\_practice\apache-maven-3.3.9\bin**

We arranged path in environment variables. Now we need to know whether it is working or not. Now open the cmd.

Enter the cmd : java –version

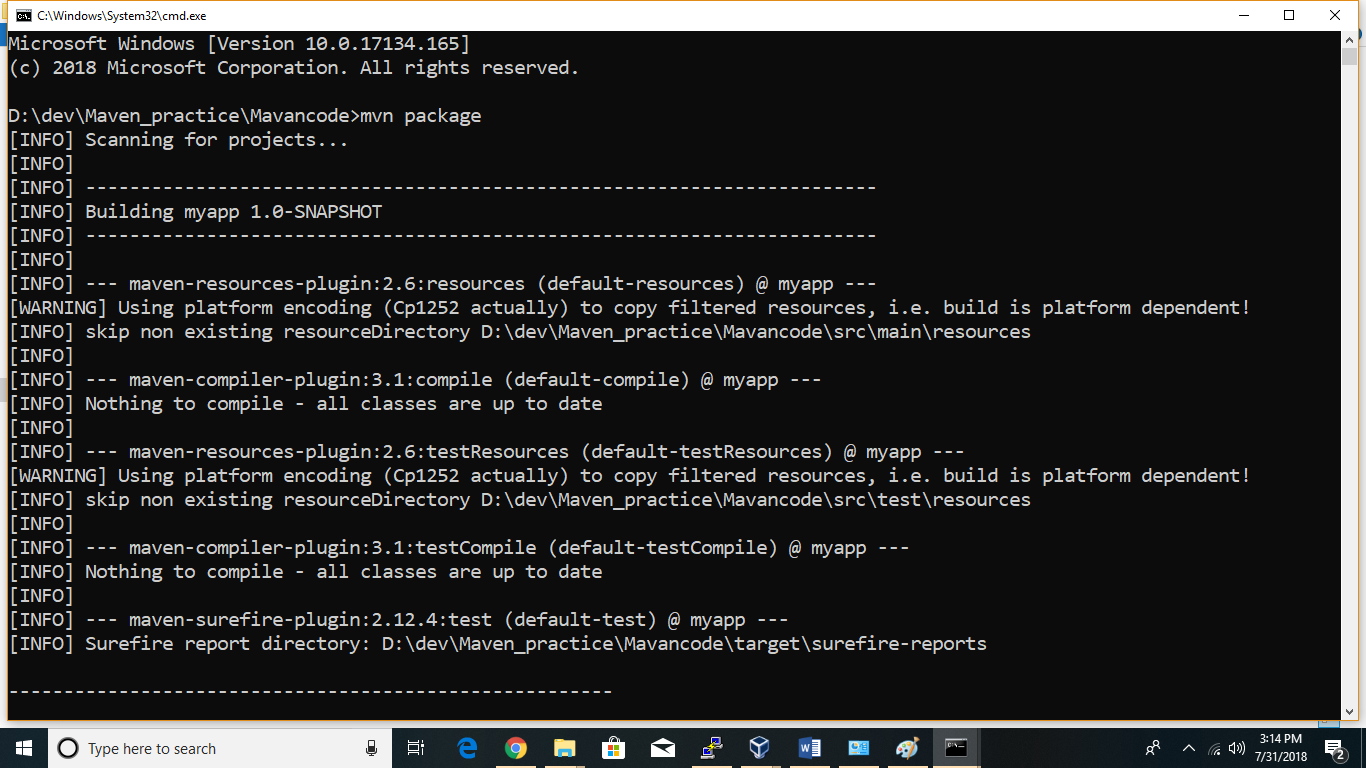
If it displays version details it is ok.

Next enter the cmd for verifying maven : mvn –version.



To build the code manually in windows:

* For that create folder ex : maven practice
* Inside the folder create src folder and one pom.xml file.
* After that we have to give required java code and required dependencies in pom.xml file.
* Open the cmd prompt in that maven practice to build the code.
* Enter the cmd : mvn package
* if build is success we get target file and in that jar files will be presented.



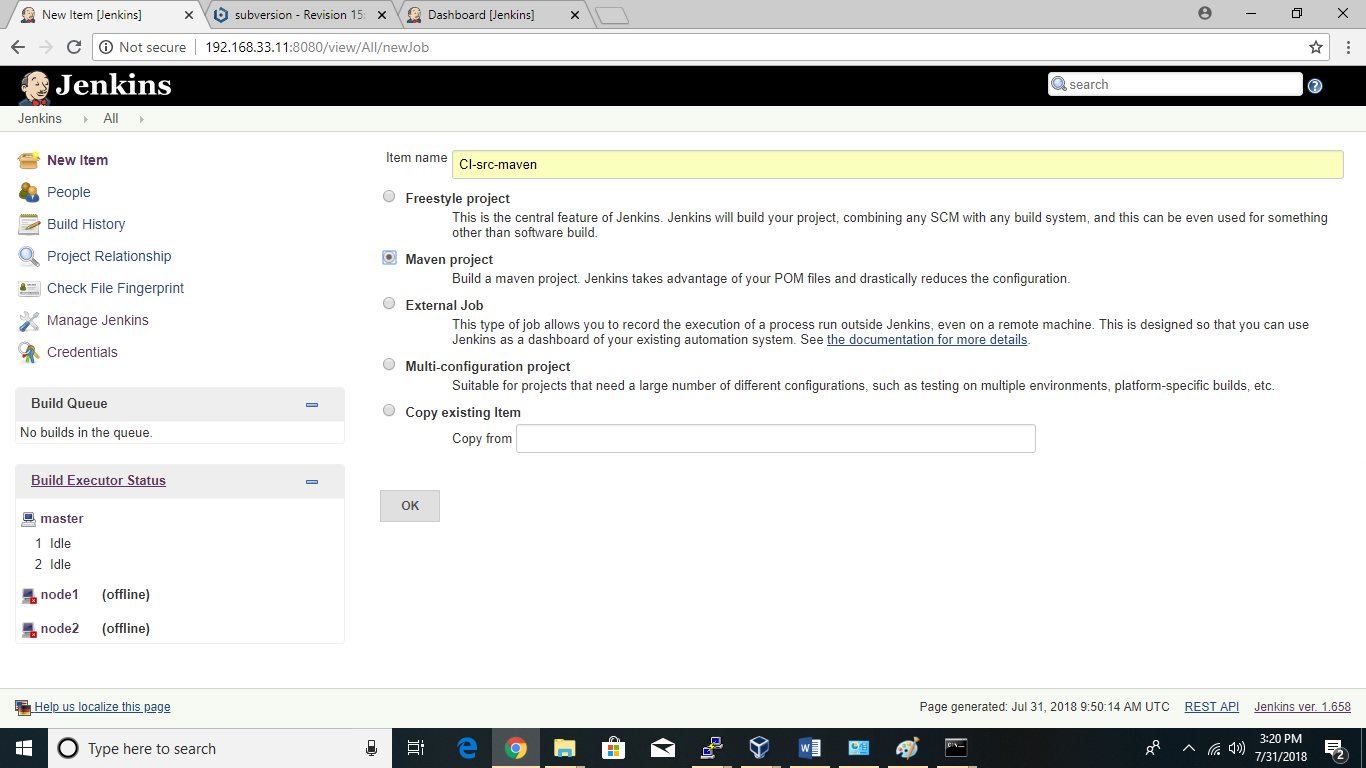
How to build code in Jenkins:

* Open the Jenkins by using the command : java –jar Jenkins.war
* Open the chrome browser and enter the ip 192.168.33.11:8080 as per your ip you enter in the browser.
* Before creating a job. First we need to create node and in configure system we should set jdk with name of JAVA\_HOME and maven as M2\_HOME. (creation is explained in above pages).
* Select New item

Item name : CI-src-maven

Select : maven project

Click ok



Now entering into the configuration part we have to give all details as we done before by using ANT.

First enable the restrict where this project can be run

* Label expression : select node

Now come to source code management

* Select subversion
* Repository url : ex: <http://192.168.33.11:8081/subversion/maven/>
* Now come to pre-steps

Select invoke top level maven targets

* (remember for ANT we will select invole-ant)
* Maven version :M2\_HOME
* Goals: package

Now come to build

Root POM : POM.xml

Goals and options: package

Select post build actions

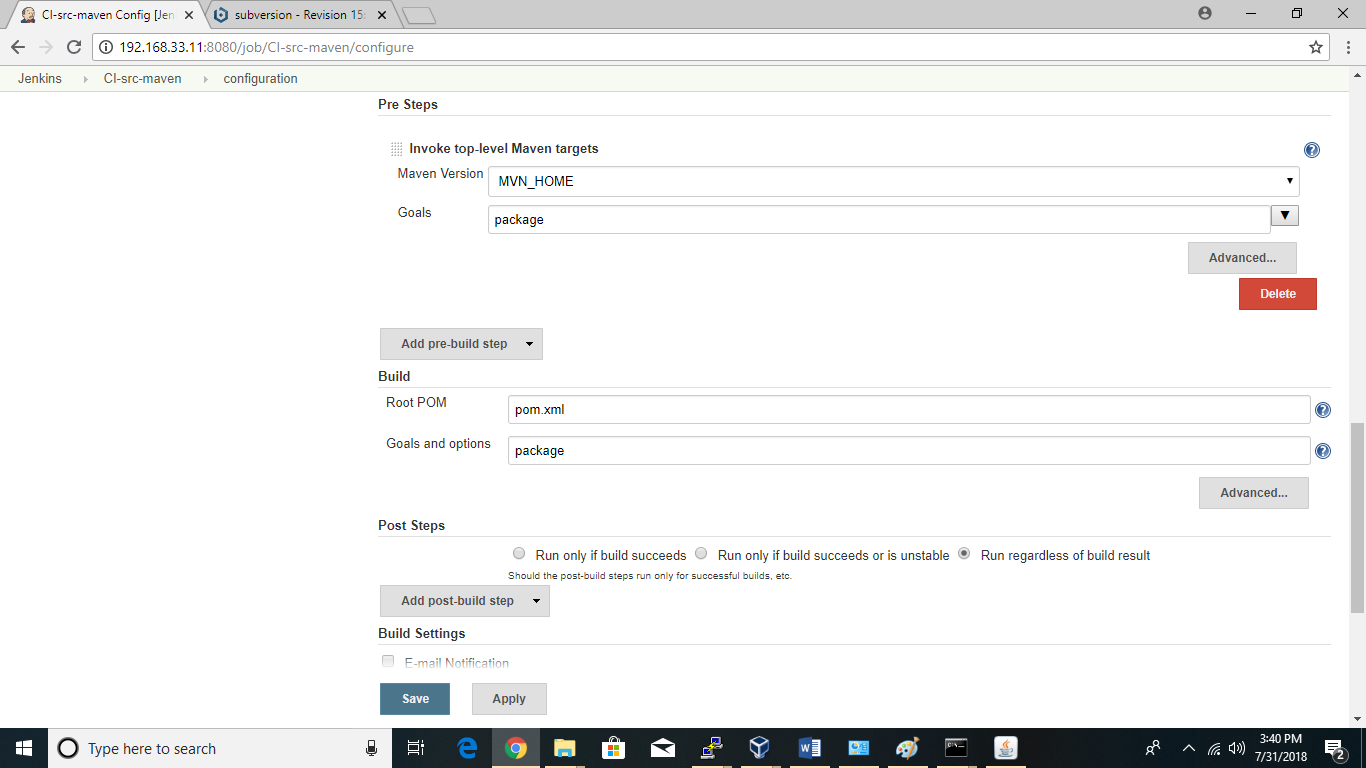
Archive the artifacts

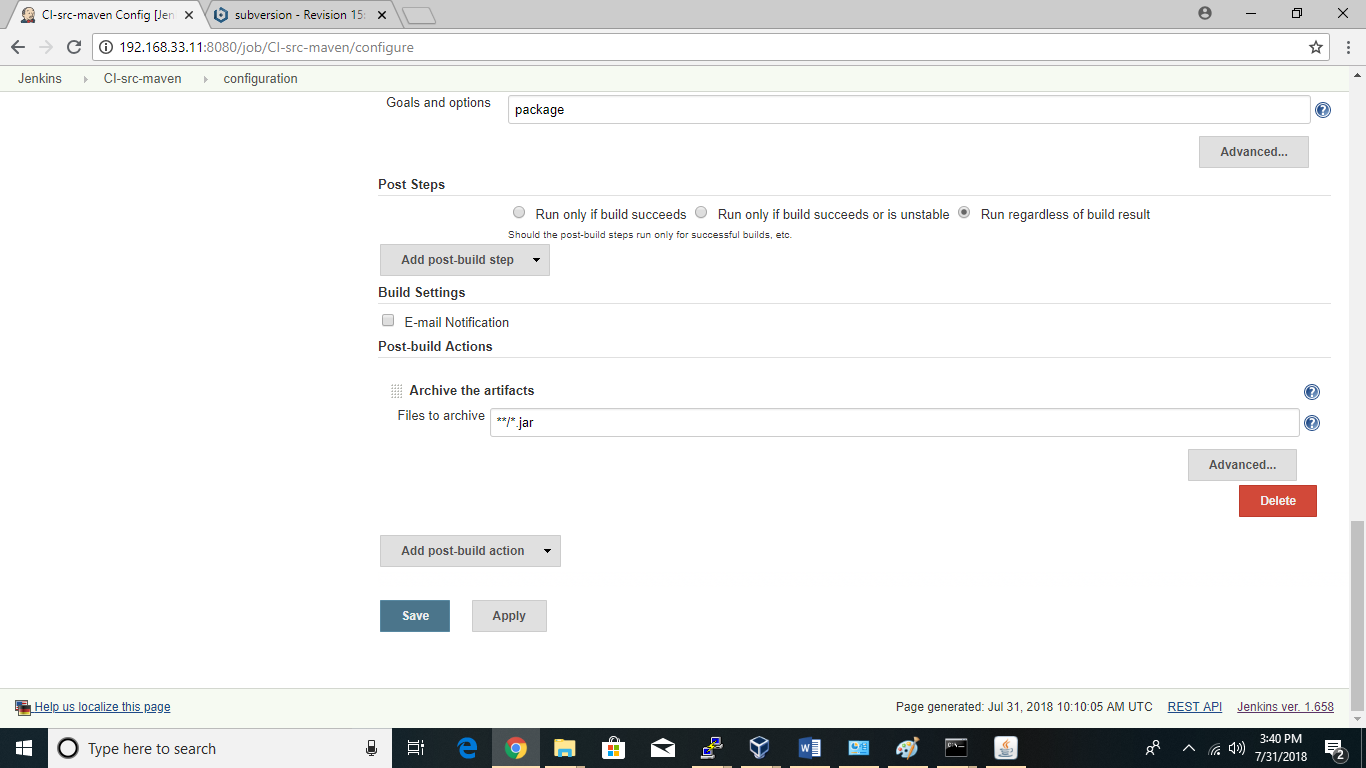
Files to archive: \*\*/\*.jar

Save the job.

Select build now.







Maven repositories:

Maven have three types of repositories:

1. Local repository.
2. Central repository or .m2 folder.
3. Remote repository or backup repository.

Maven life cycle:

Maven life cycle have stages. They are

1. Validate: before it will come to compileation.maven will validate the code.
2. Compile: compile the source code and create .classfiles.
3. Test: before packaging maven will test .classfiles.
4. Package: collection of .classfiles.
5. Verify: verify the package before install.
6. Install: installing package locally.
7. Deploy: installing package remotely.

Pom.xml structure:

it is mainly can describe in 3 ways.

1. G A V parameters

G – group ID

A – artifact ID

V – Version

1. < scm >

192.168.33.11:8080/subversion/trunk

< /scm >

1. < distribution management >

Nexus urls

< /distribution management >