Jenkins

* Installation

Run Jenkins is 9090

* Setup email
* Setup node
* Install mvn plugin
* Run maven for java
* Create pipeline using groovy.
* General pipeline method.
* Migrating Jenkins jobs, plugins from one Jenkins server to another.

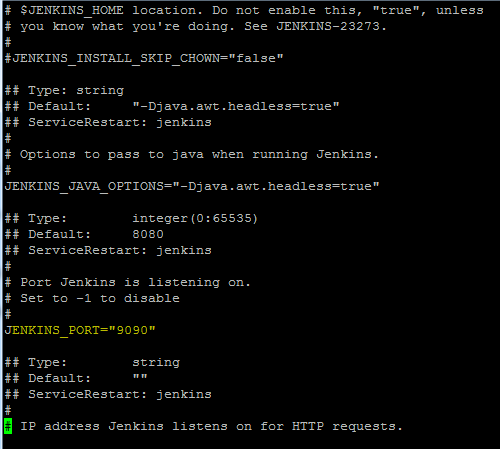
**Installation**

Run the below commands.

* yum install java.
* sudo wget -O /etc/yum.repos.d/jenkins.repo <http://pkg.jenkins-ci.org/redhat-stable/jenkins.repo>
* sudo rpm --import [https://jenkins-ci.org/redhat/jenkins-ci.org.key](http://pkg.jenkins-ci.org/redhat/jenkins-ci.org.key)
* sudo yum install Jenkins

in rhel if we want to run Jenkins on a different port

edit the file - /etc/ sysconfig/Jenkins

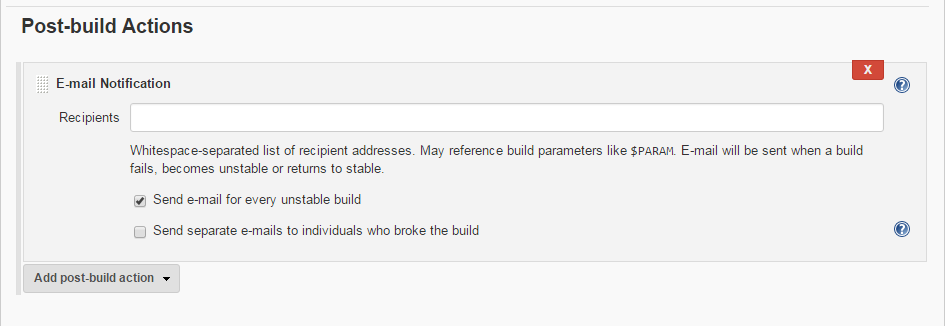


**Configuring email in Jenkins:**

Type 1: when the build fails.

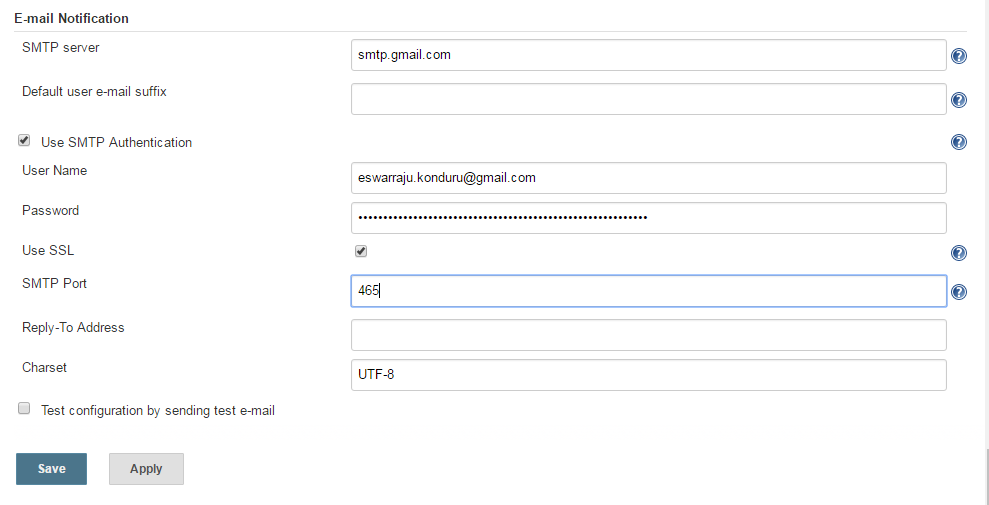
Install the plugin “email notification”

This configuration is done at the job level, in the post build action enter the email id.

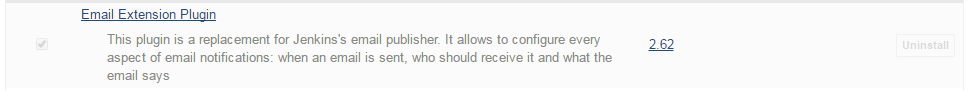


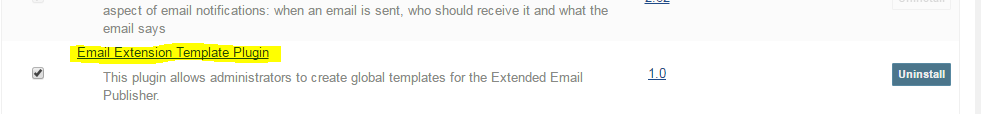
Goto manage Jenkins -> configure system

Then add email id in the “email notification“ section.

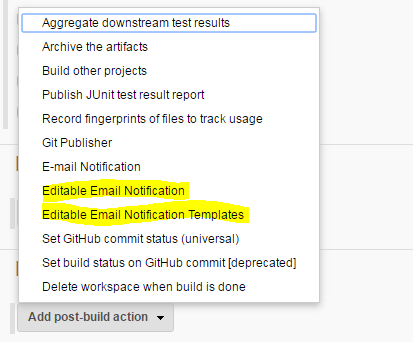


Type 2 – We can trigger automated email on various job result like (fail, success etc), this is achieved with the help of plugin “Email extension plugin”

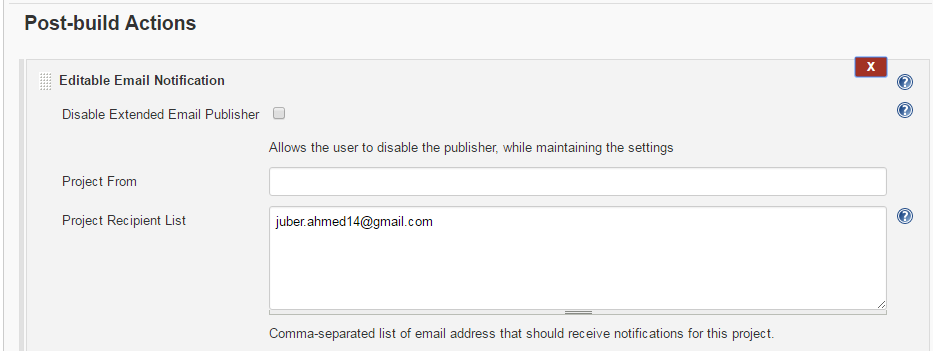




Once plugin is installed will get the options under post build action as shown below

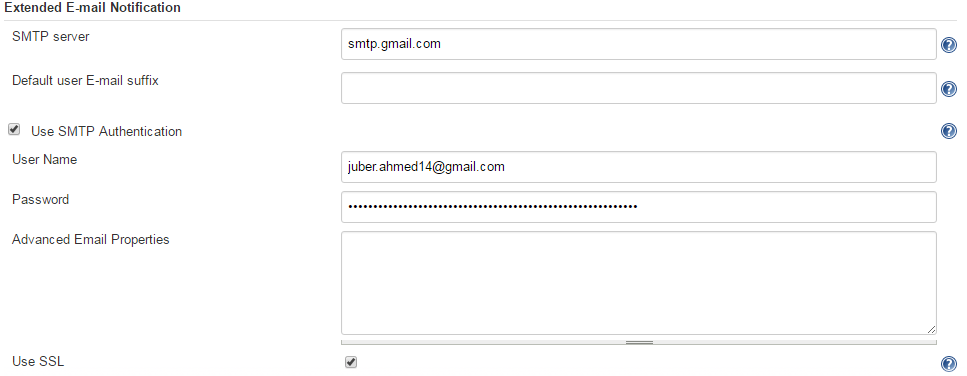


Select “editable Email Notification”, add email address, rest can be left default.



Next, goto “manage Jenkins” -> “configure system”

Configure the “extended E-mail notification” tab, click on “advanced” and enter smpt and email id details as shown below.



Come down to select “default Triggers” in the same section to select when email should be sent,

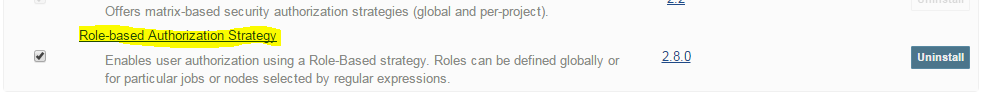


**Configure Jenkins to give users project level access.**

Step 1 - Create a new user.

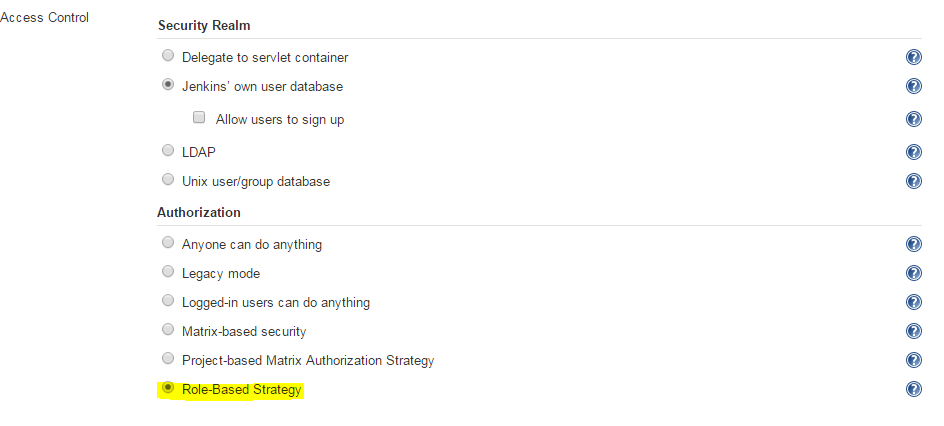
Manage Jenkins -> Manage users – Create user

Step 2 - we need to install the plugin – “role based authorization strategy”



Step 3 – go to manage Jenkins – global configuration settings

Select “role based strategy”

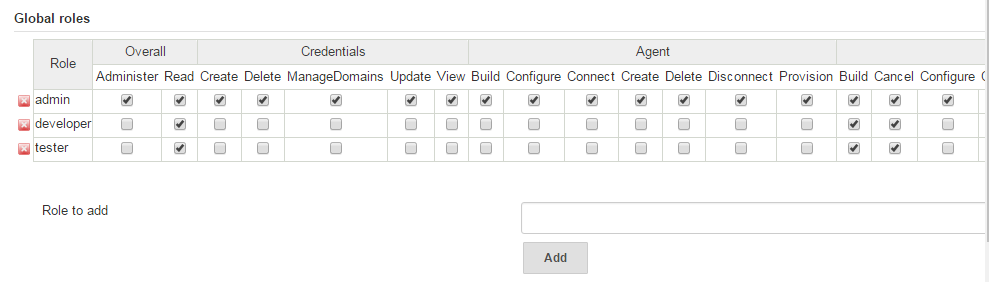


Step 4 – Plugin installation should give you “mange and assign roles” under the manage Jenkins.

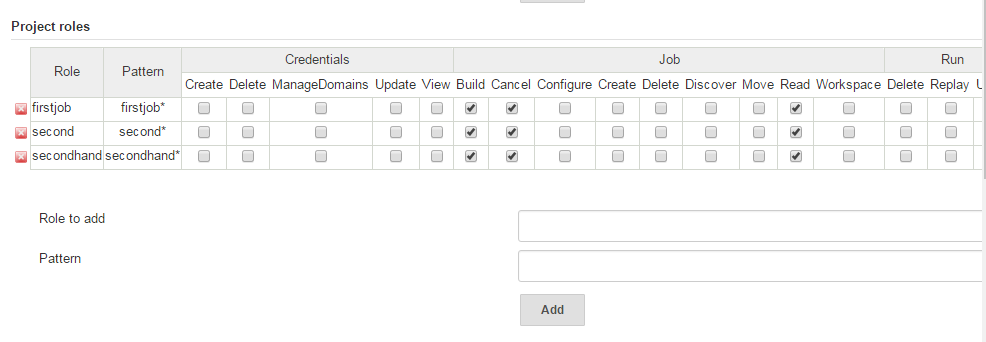


Step 5- Select “manage and assign roles” -> “manage roles”

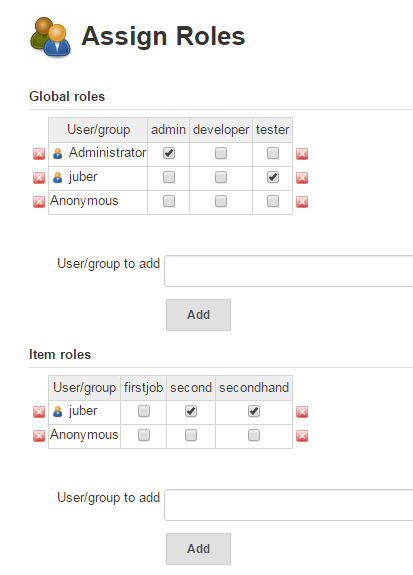
Create roles example developer, tester etc. and give access rights , this role would be assigned to a user later.



Step 6 – in the same section add project as below and give rights.



Step 7 – Go to assign roles and assign roles accordingly.



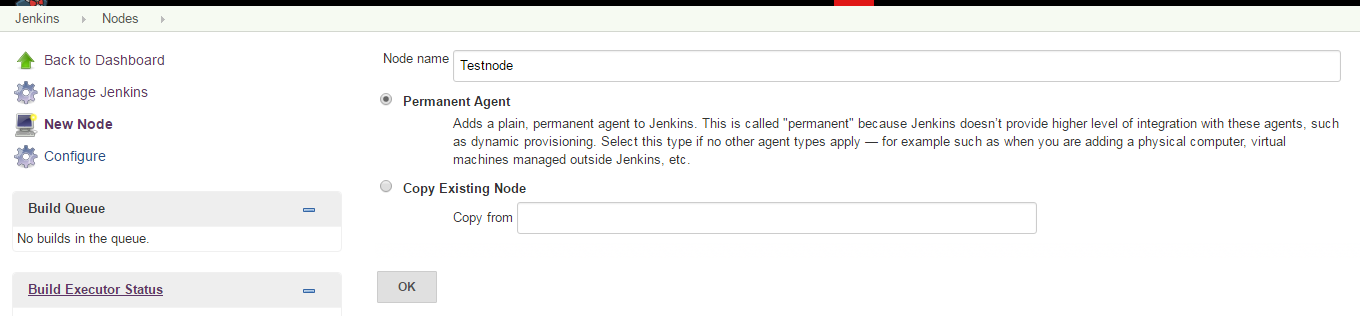
Now login as the user and you will see only the job that the role for that user has.

**Working with Nodes in jenkins**

Task - Create a new node, Commands will run from Jenkins dashboard in the node.

Goto “manage Jenkins” -> Mange nodes

Give a name to the new node as shown below



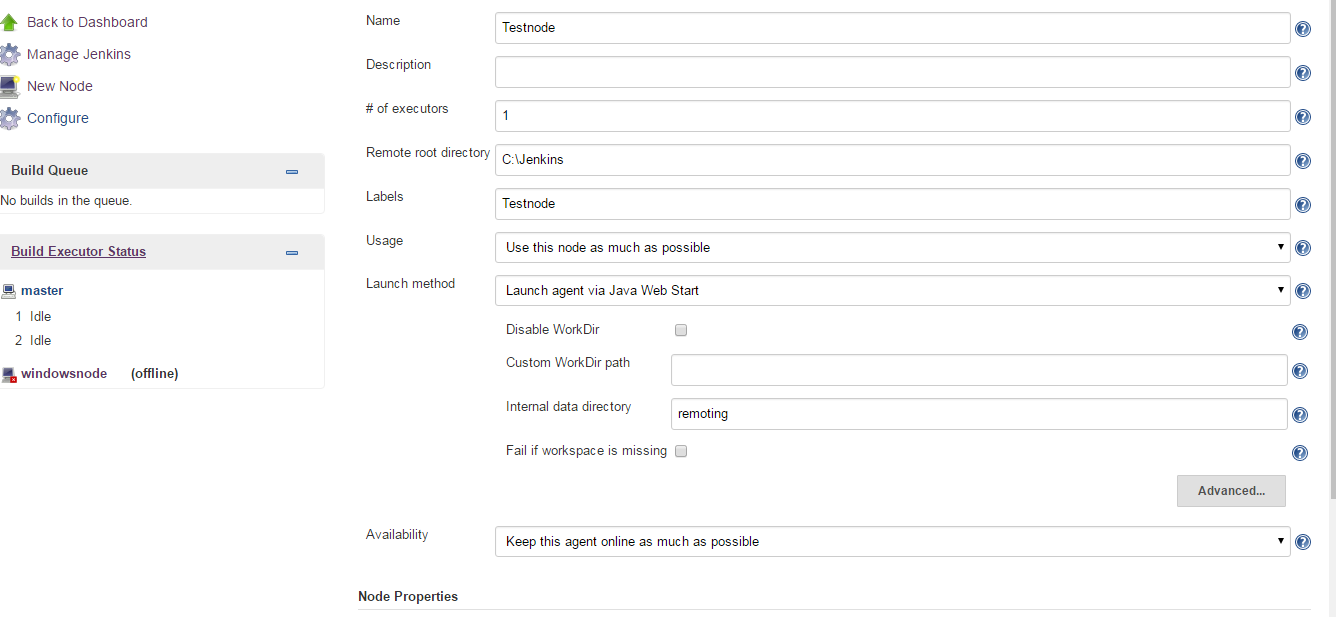
Press ok, and enter details

Name - <any name>

Remote root directory = <work space location in node machine>

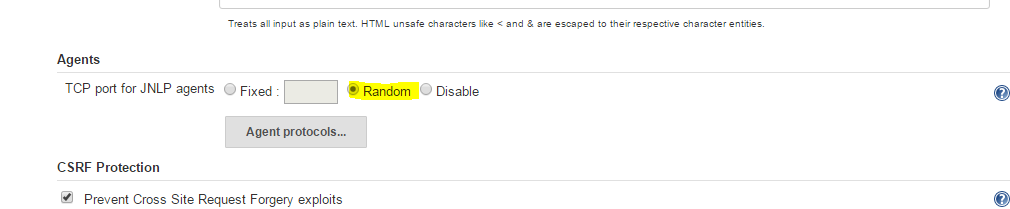
Labels = <name of the node>

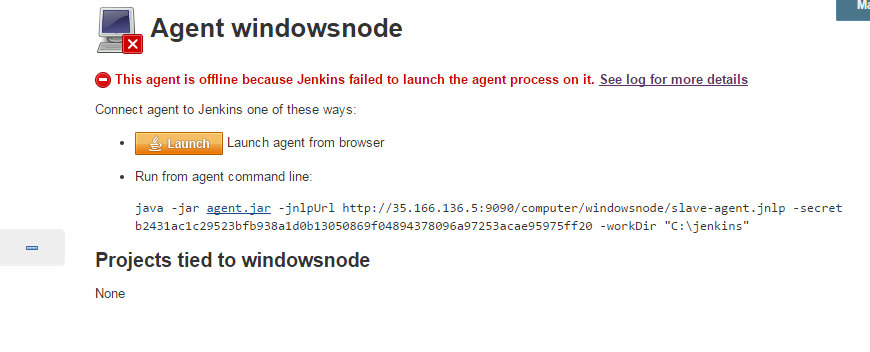
Launch method=”Launch agent via java web start”



Goto “manage Jenkins” -> “configure global security” agents tab

Check the random button as shown below



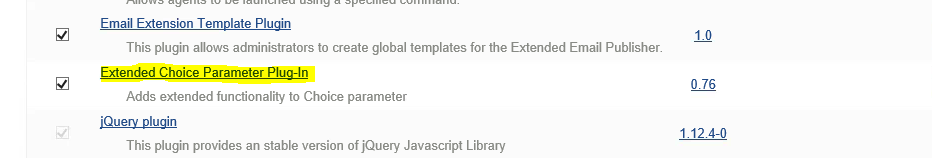


**To configure the job with Choice parameter:**

We have to install plugin named “Extended Choice Parameter Plugin” using below steps.

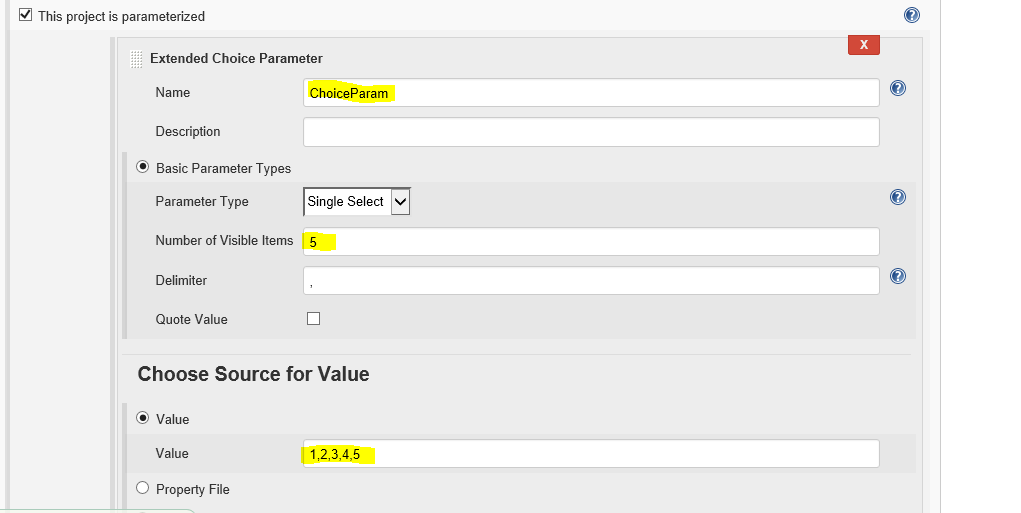
Go to “Manage Jenkins” -> Manage Plugins -> Select above plugin and install

Once installed, it will look like below in Plugin section.

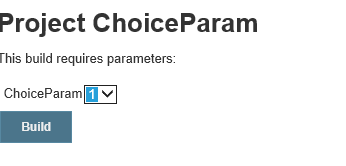


Now create the Job and go to configure section.

And please add the below details to Build with parameters as per your choice.



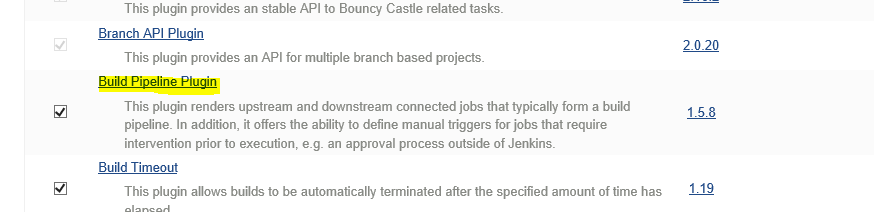
Once done this and select Build option you will be prompted for to select the Choice Param value to Build the job.



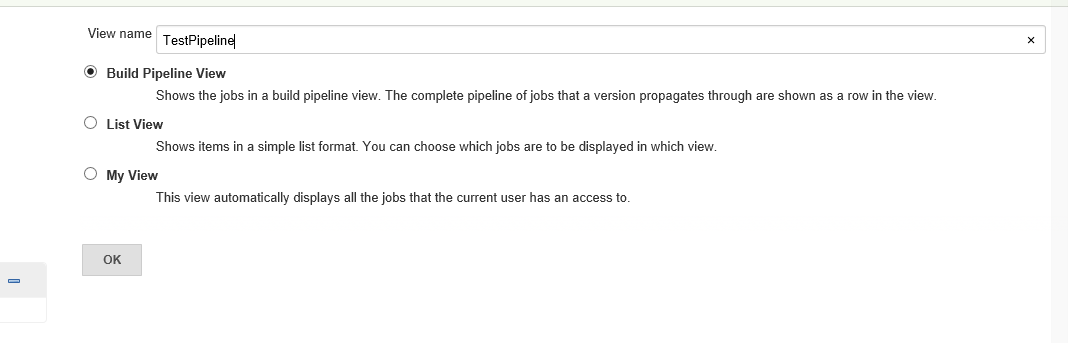
**Code Delivery pipeline.**

We need to Install Build Pipeline plugin to create the pipeline for the Jenkins jobs.

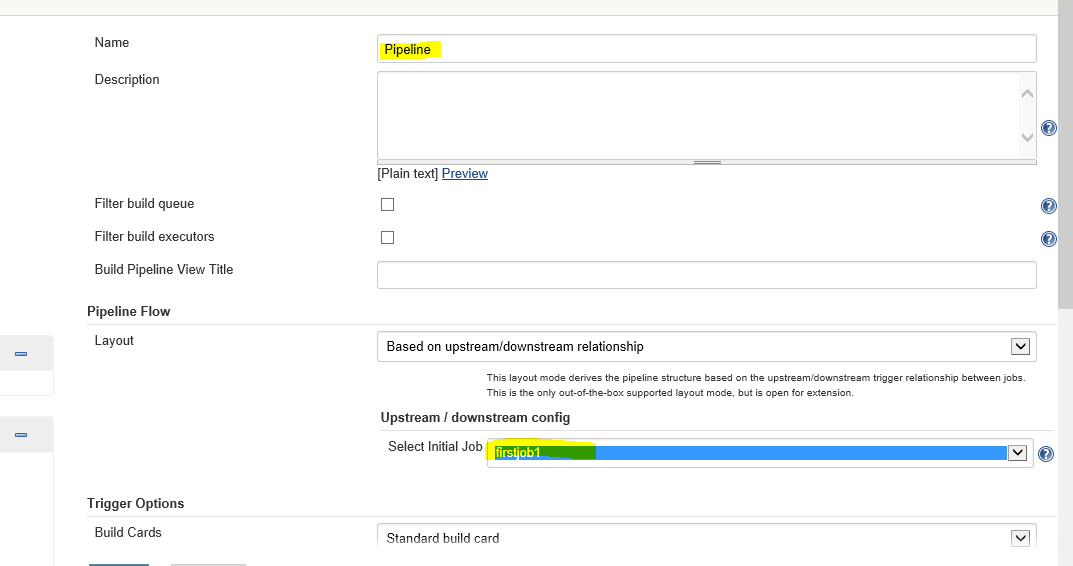
Go to Manage Jenkins -> Manage Plugins -> Select the Build Pipeline -> Install



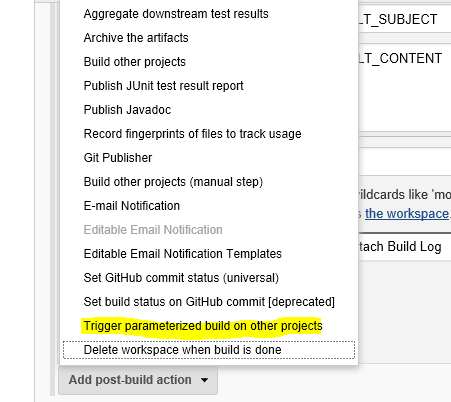
Now go to Jenkins dashboard and select + symbol on top the jobs section and you can view the below.



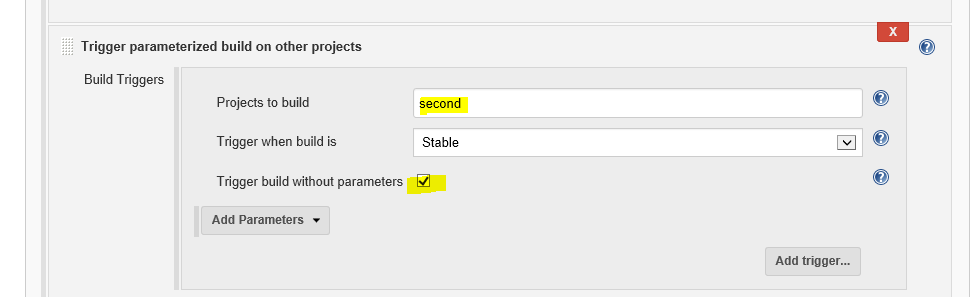
Now select the below options to select the initial job to trigger in pipeline as below.



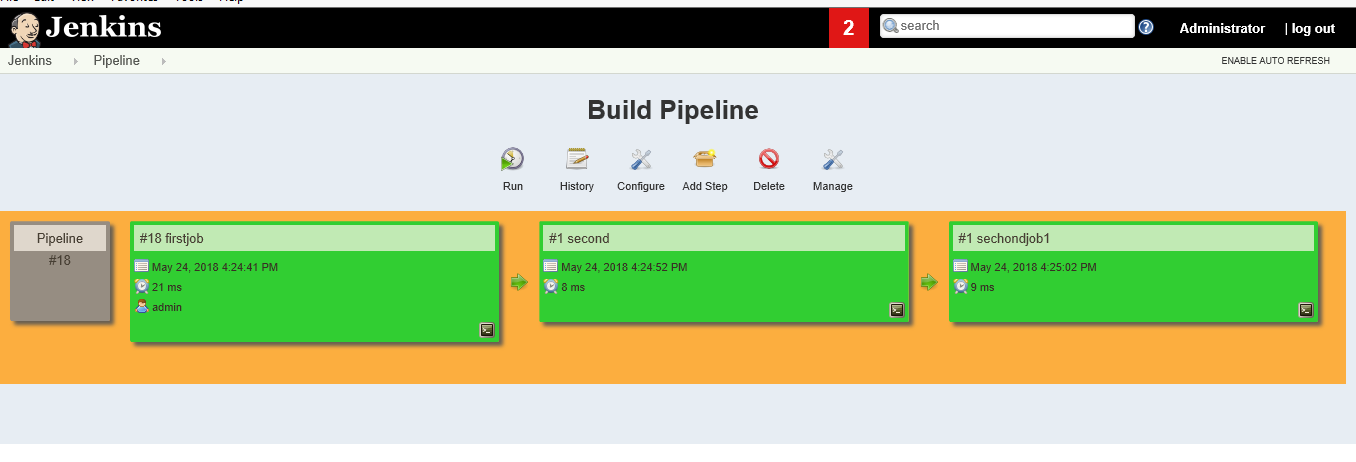
Now go to the firstjob1 (initial job) and select configure for add post build as “Trigger parameterized build on other projects” action as below.



Now select the next job which you want to add in the pipeline as below.

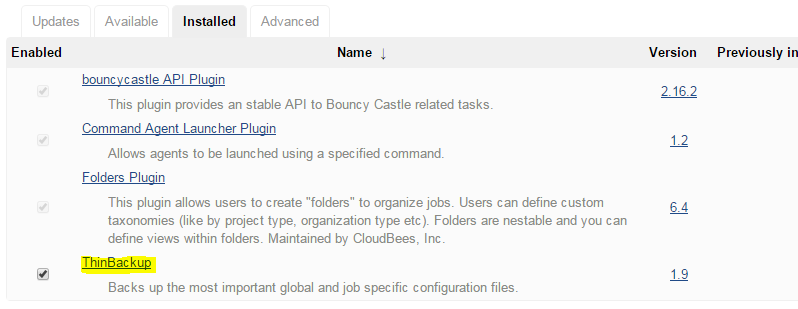


Similarly you can add any number of jobs in to the single pipeline in Upstream/Downstream. And now once done this you can see the actual pipeline as below.

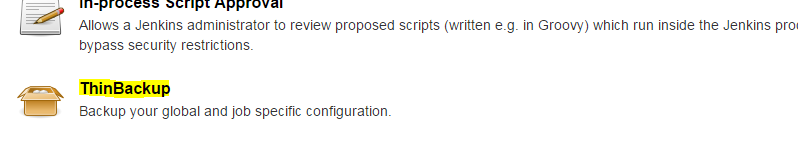


**Migrate Jobs, plugins etc from one Jenkins server to another.**

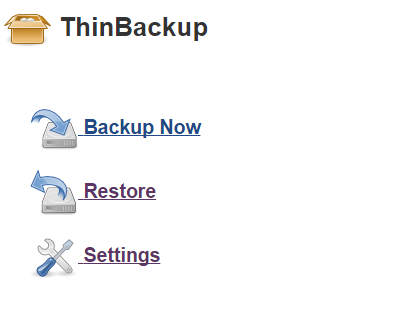
Install thinbackup plugin should be present in both the Jenkins server i.e. the source server and destination server.



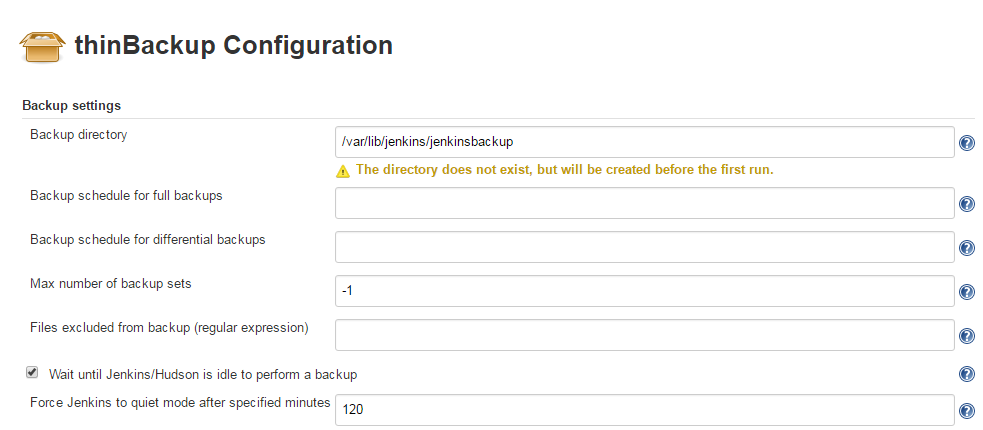
Once installed you will find under manage Jenkins as below.

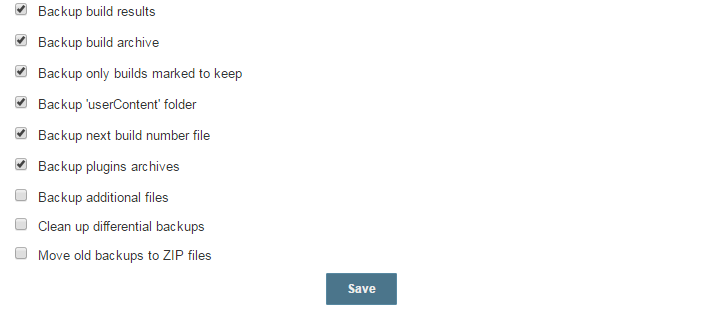


Click on the think backup shown above to get the below screen.



Click on settings

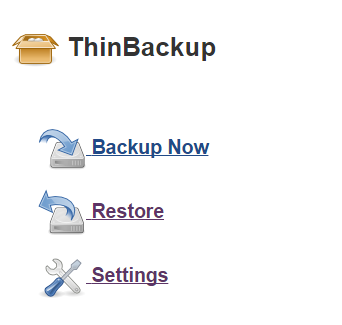




Enter the directory where the backup is to be created in “backup directory” box and select all other options which are appropriate.

Click “Save”

Then go back and select “backup now”



Now to restore we need to have the thin backup plugin on both server as mentioned earlier.

Copy the created backup folder in the destination Jenkins server using scp.

NOTE – We need to have full permission on the backupfolder created in the source server and destination.

Now in the destination Jenkins server Dashboard go to manage Jenkins -> thin backup

Go to settings and give the path of the location where the folder was coped.

Next, click on Restore -> select the restore from -> “restore”.

**Installing maven in jenkins server and building java code.**

First install maven:

step 1 – cd /usr/local

step 2- wget <http://www-eu.apache.org/dist/maven/maven-3/3.5.3/binaries/apache-maven-3.5.3-bin.tar.gz>

step 3- tar xzf apache-maven-3.5.3-bin.tar.gz

step 4 - ln -s apache-maven-3.5.3 maven

step 5 - sudo vi /etc/profile.d/maven.sh and add the below

export M2\_HOME=/usr/local/maven

export PATH=${M2\_HOME}/bin:${PATH}

step 6 - source /etc/profile.d/maven.sh

step 7 - rm -f /usr/local/apache-maven-3.5.3-bin.tar.gz

<https://tecadmin.net/install-apache-maven-on-centos/>

refer the link for above.

# Customizing instance headers

Below you can find a list of steps, which allow customizing headers of both  
CloudBees Jenkins Enterprise and CloudBees Jenkins Operations Center.  
These steps apply to the versions 1.625.x and 1.642.x, steps for other versions  
may differ.

1. Install [Simple Theme Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Simple+Theme+Plugin)
2. Go to the JENKINS\_HOME/userContent directory
3. Create the layout directory
4. Create the style.css file, copy the contents provided below.
5. Customize the “content” text
6. Put the logo of your instance to JENKINS\_HOME/userContent/layout/logo.png. The logo should have 40px height.
7. Go to the global configuration of your CJP instance and find the Theme section there
8. In URL of Theme CSS specify the following path: http://example.com/userContent/layout/logo.png

Example of Simple Theme Plugin configuration:

style.css code:

/\* Custom style for CloudBees Jenkins Platform \*/

.logo img {

content:url("http://example.com/userContent/layout/logo.png");

}

.logo span {

display: none;

}

.logo:after {

content: 'Jenkins. My instance (powered by CJE)';

font-weight: bold;

font-size: 20px;

position: relative;

color: black;

right: -370px;

top: -28px;

display: block;

color: white;

background-image: url("http://example.com/userContent/layout/logo.png");

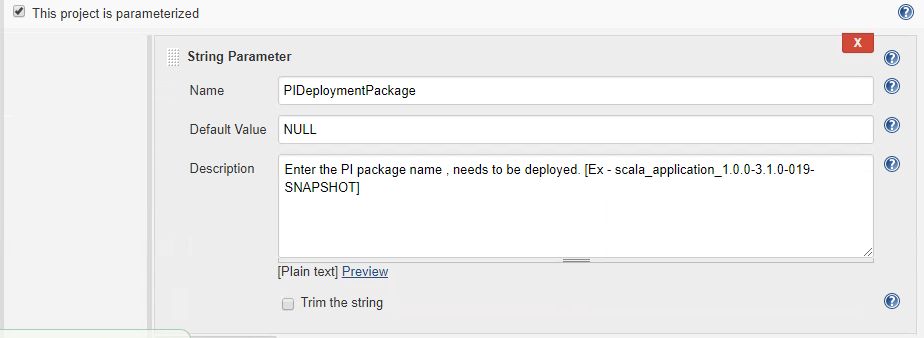
background-position-x:right;

background-position-y:top;

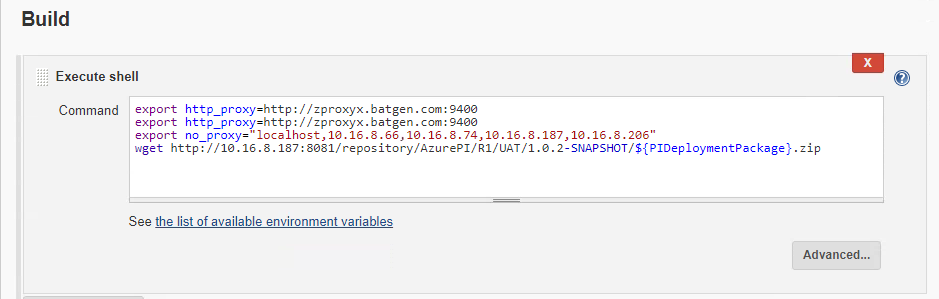
}

**Pass parameter from Jenkins to a script.**

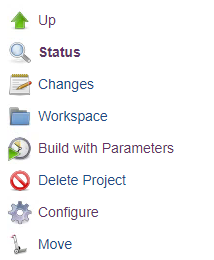
Go to job -> “configure" -> “this project is parameterized.”

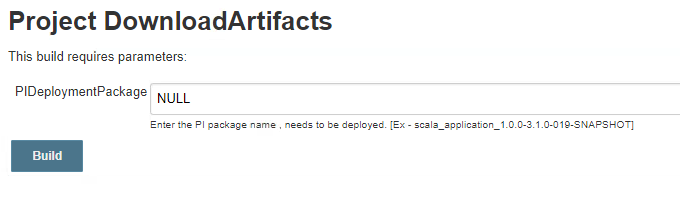


Define the parameter in the script as below.



While building the project we have to pass the parameter which it will substitute in the script.





Pass the value as above.

**Use case**: nexus has the zip files, we need to download the zip files into the Jenkins, and copy to MS build machine for deployment.

Nexus file name should be passed therefore we are passing a parameter so that we can specify the file name as parameter while running the job.

Furthermore, we can assign the same parameter to a variable in post build action if in case the same file name is required in the subsequent jobs.