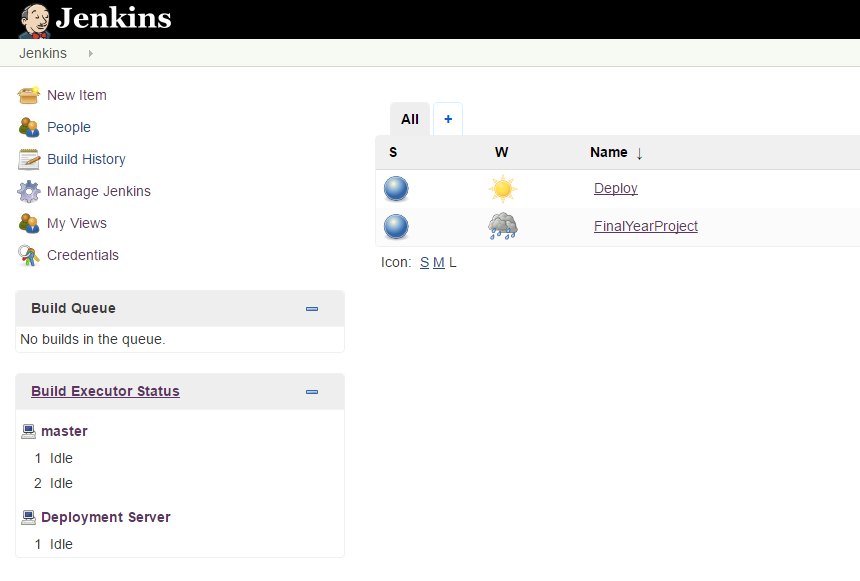
**Contents**

* Jenkins
* Install Jenkins
* Why Jenkins
* Setup
* Creating a Job (Simple Build & Test)
* Creating a Slave Machine (To do work on deployment server)
* Creating a Job (Deploy app when another job passed)

**Jenkins**

<https://jenkins.io/>

“The leading open source automation server, Jenkins provides hundreds of plugins to support building, deploying and automating any project.”



**Install Jenkins:**

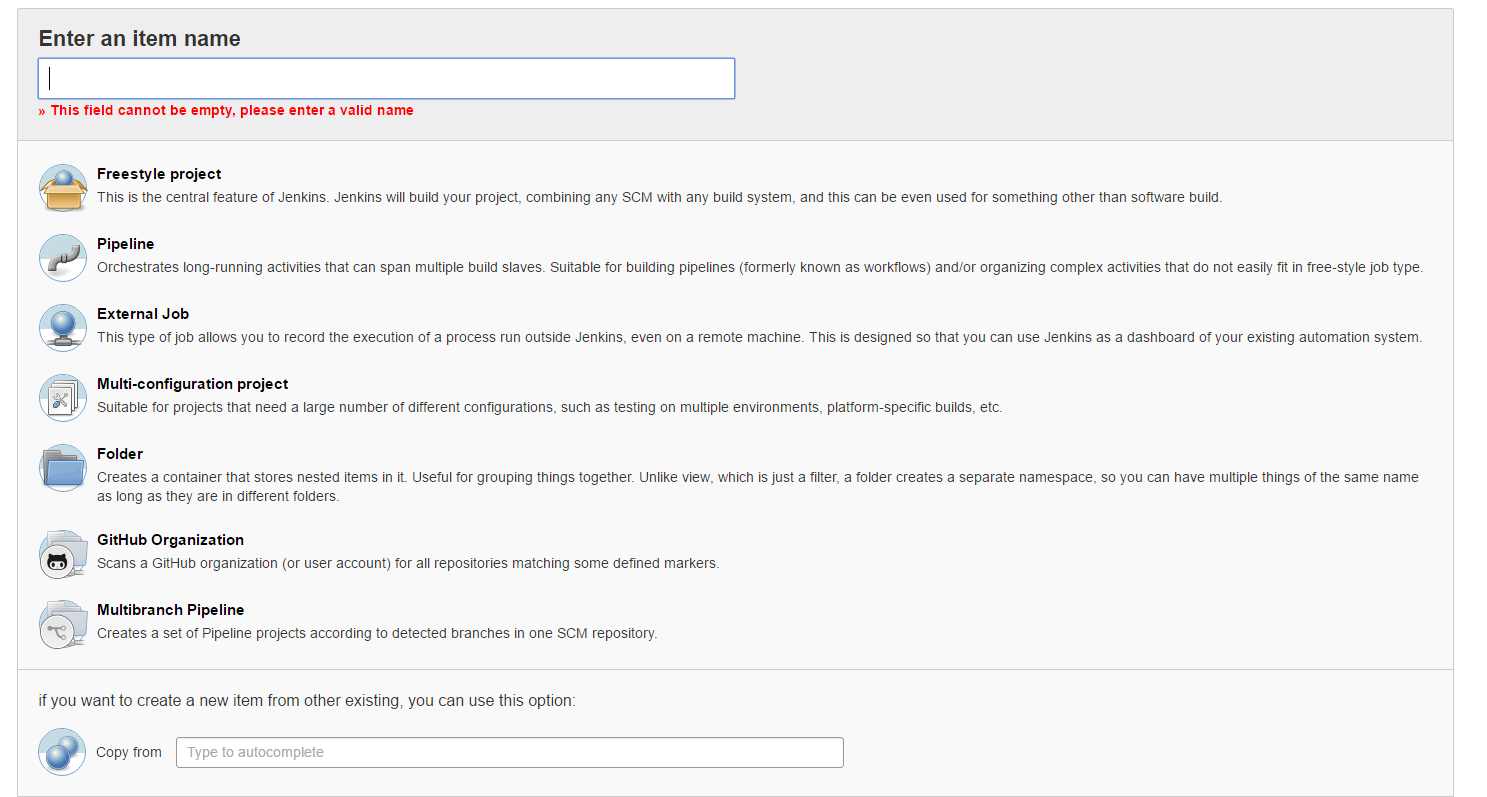
<https://wiki.jenkins-ci.org/display/JENKINS/Installing+Jenkins+on+Ubuntu>

**Why Jenkins?**

When creating small, large or enterprise apps and projects, you will need a centralized build platform, that will take control of all the various builds, tests, and deployment.

Jenkins fits the bill here; it is a continuous integration/ continuous delivery server. Which is extensible with numerous plugins. And can be distributed if needed, Jenkins is easy to install and configure.

Here is an example of the various tasks/jobs Jenkins can preform



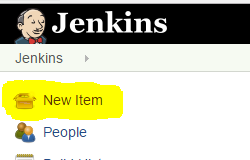
**Setup**

This tutorial will show how to set up a simple build and test task which can be followed by a deployment job.

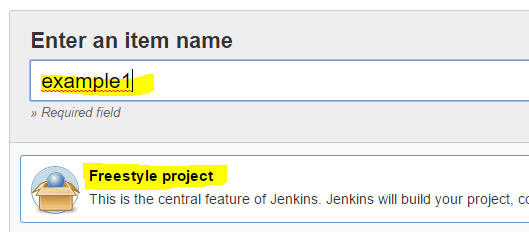
Useful: <http://www.vogella.com/tutorials/Jenkins/article.html>

**Creating a Job (Simple Build & Test)**

Select the “New Item” button to create a new job.

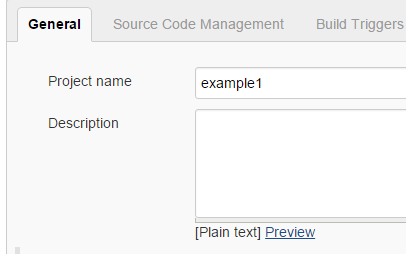


Then select the job you wish to use, in this case we are going to do a freestyle project, give it a name and select ok.



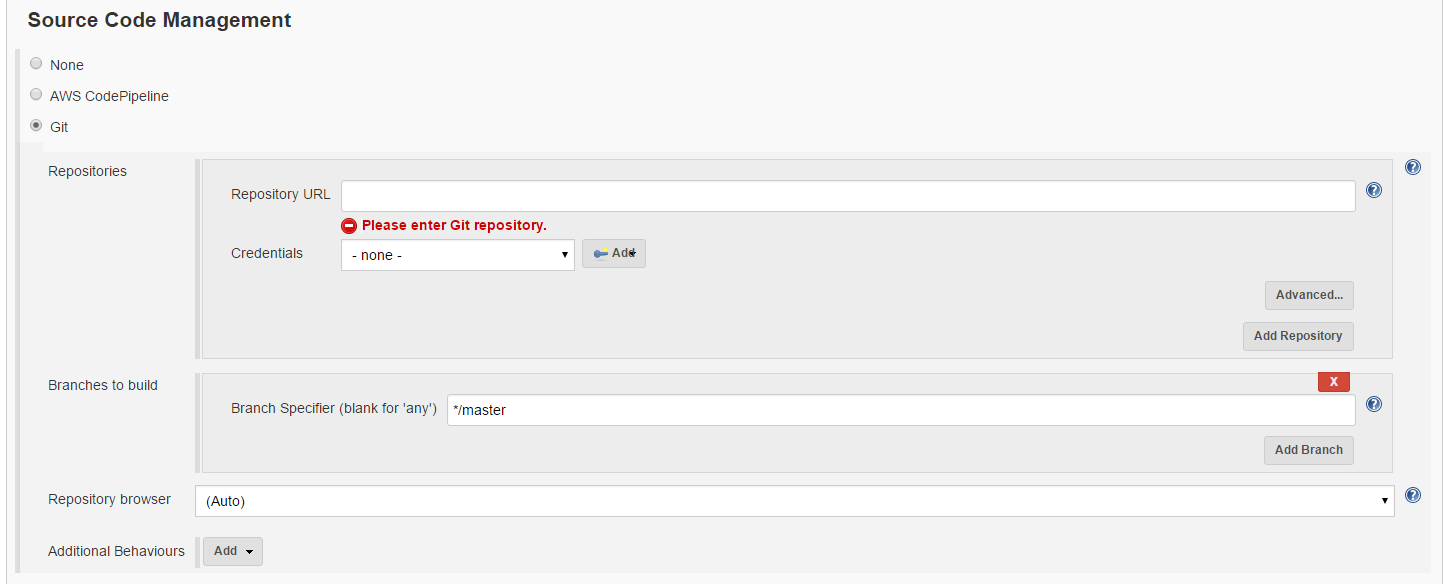
This will bring us to the configuration screen for this project.

Enter a description if you wish, as this may help other users understand this job easier.



Next we would like this job to pull from Git, scroll down to “Source Code Management” and select Git (if option is not available, you may need to install the plugin)

Enter the URL for the repository that you wish to use, credentials if necessary and what branch to use.



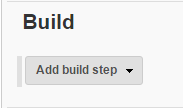
Next we want to setup our build environment for the app we are cloning from Git. Since we are creating a Node app, we must ensure that we have the Jenkins NodeJS plugin installed so we can select the “Provide Node & Npm bin/ folder to PATH” option.

Plugin: <https://wiki.jenkins-ci.org/display/JENKINS/NodeJS+Plugin>





Now we can get onto the build itself, scroll down to “Build” and select “add build step” and select Execute shell, since we want to issue node commands and grunt commands, i.e. npm install, grunt build etc



Next input your commands into the command text area.

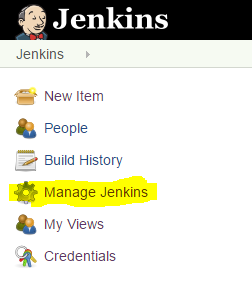


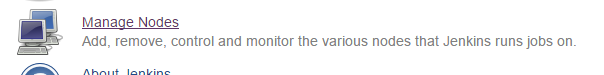
This is it for this simple task, once this task is fired and successful, we can be confident that all the tests pass (if anything fails -> build fails) we can then do other jobs after this one.

**Creating a Slave Machine (To do work on deployment server)**

First we must create a slave machine and point it towards our deployment server

-> Manage Jenkins -> Manage Nodes

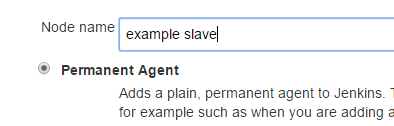




Then select “New Node”

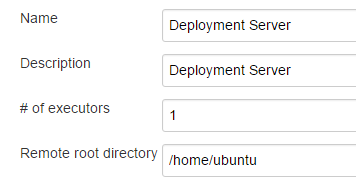


Give the slave a name and select permanent Agent



Fill in the description as this will make it easier to distinguish between nodes later

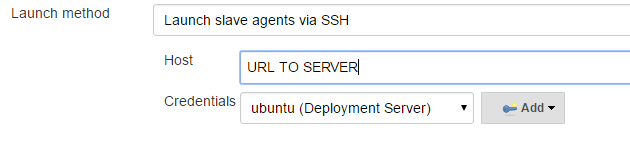
Also the root directory you wish to use



Next is the part where we will label this slave, so other jobs and tasks can select this slave to run its job on



The final step for our slave machine now is to select the launch method, provide the Host details and credentials.



Once we have all this complete it will show up on the home page under “Build Executor Status” pane.



We can now do work on this machine.

**Creating a Job (Deploy app when another job passed)**

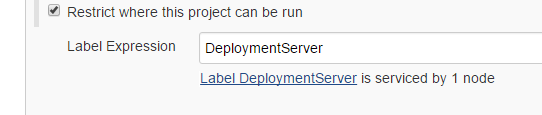
Again we are going to create another Freestyle project, the aim of this task will be to issue commands on our deployment server to pull the latest from Git rebuild and restart the node server (forever instance)

Forever: Run scrips continuously

<https://www.npmjs.com/package/forever>

Then as usual input the description

Then next we want to tell this job to use our slave machine we have created, tick the box that says “Restrict where this project can be run”

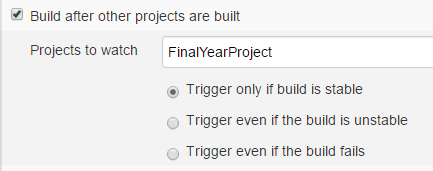


This will ensure that this job is run on our deployment machine

Next we want this build to fire when the other build has successfully ran, scroll down to “Build Triggers”



And tick the box beside “Build this after other projects are built”



Since this is a node app we need to check the box to “provide Node & npm bin/folder to Path”



Finally, under “Build” select “execute shell” and input your commands. Here I am going to checkout master branch, and then pull it to ensure we have a clean working version. Then do the build steps.

