****

Table of Contents

[1. Continuous Integration Overview 3](#_Toc425468830)

[2. What is the Jenkins integration server? 3](#_Toc425468831)

[3. Jenkins VS Other Conventional CI tools 4](#_Toc425468832)

[4. Jenkins home page 5](#_Toc425468833)

[5. Install Jenkins as a Windows service 5](#_Toc425468834)

[6. Introducing Master Slave Setup on Jenkins 12](#_Toc425468835)

[Introduction: 12](#_Toc425468836)

[Prerequisites: 12](#_Toc425468837)

[Creating Node: 12](#_Toc425468838)

[Configuring Node: 14](#_Toc425468839)

[Launch the JNLP agent: 15](#_Toc425468840)

[Launch the node in remote directory: 16](#_Toc425468841)

[Launch the slave as windows service 18](#_Toc425468842)

[7.Main components of Jenkins 20](#_Toc425468843)

[8.Plugins 24](#_Toc425468844)

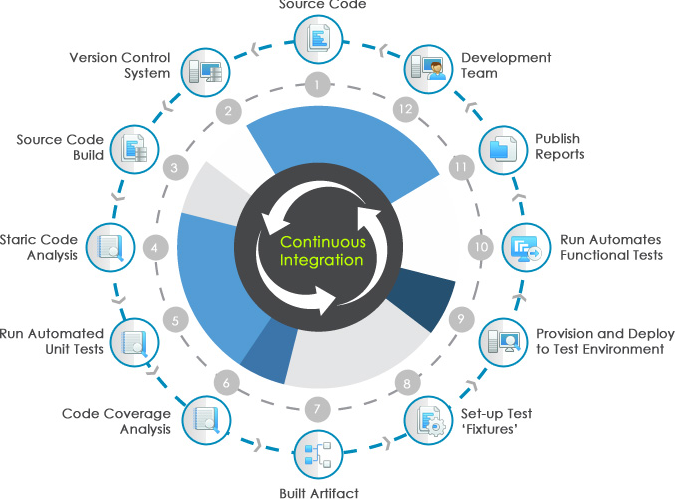
[9. Global Security 27](#_Toc425468845)

[10. Security Realm 27](#_Toc425468846)

[11. Configuring Security in Jenkins 27](#_Toc425468847)

## 1. Continuous Integration Overview

Continuous integration is a process in which all development work is integrated at a predefined time or event and the resulting work is automatically tested and built. The idea is that development errors are identified very early in the process.



## 2. What is the Jenkins integration server?

**Jenkins** is one open source tool to perform continuous integration. The basic functionality of Jenkins is to execute a predefined list of steps based on a certain trigger.

The trigger might for example be an change in a version control system or a time based trigger, e.g., a build every 20 minutes.

The list of steps can for example include:

* perform a software build with Apache Maven or Gradle
* Run a shell script
* Archive the build result
* Afterwards start the integration tests

Jenkins also monitors the execution of the steps and allows to stop the process if one of the steps fails. Jenkins allows to notify users about the build success or failure.

Jenkins can be started via the command line or can run in a web application server. Under Linux you can also install Jenkins as a system service.

## 3. Jenkins VS Other Conventional CI tools

1. It lacks Static analysis of builds, lots of xml coding involved.
2. Making changes to the config files is a painful process (Not recommended stuff).
3. Understanding of Build results are bit difficult for developers.
4. Drowsy web interface
5. Build team need to spend more time in build script modifications.
6. Scheduling the project/job is difficult in remaining CI tools; however Jenkins comes up with CRONTAB syntax.
7. It is difficult to set up the dependent jobs in CI tools like Cruise Control however, setting up of upstream and downstream jobs is a life saver.
8. Jenkins covers all the mechanisms that cover by remaining tools; it is loaded with hundreds of plugins.
9. Jenkins is really good in the productive continuous integration. Since the Jenkins architecture is based on the plugins, most of the work that we trying here is already designed.
10. If we face any issues with the Administration UI, we can also make changes in the background config (which might be a last option).
11. Jenkins can have realms on security, we can restrict the specific user groups to perform the action on jobs and making them have less privileges before they mess up the Jenkins configuration.
12. If any config file changes occur we don’t necessarily restart the service that we do in conventional tool, have an option reload configuration from the disk(dynamic approach).

## 4. Jenkins home page

Here is the Jenkins homepage - <http://jenkins-ci.org/>



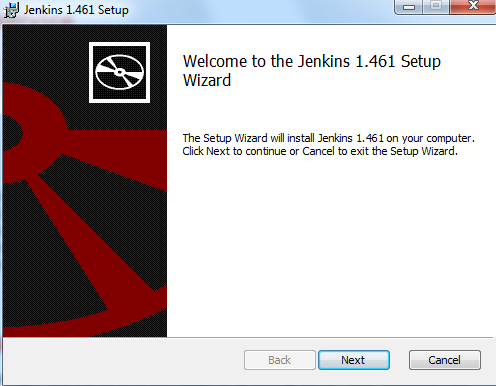
## 5. Install Jenkins as a Windows service

**There are two ways to installations.**

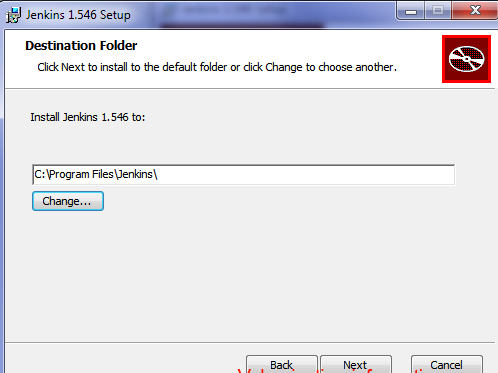
* **MSI installation**
* **Running a WAR file**

**MSI installation**-

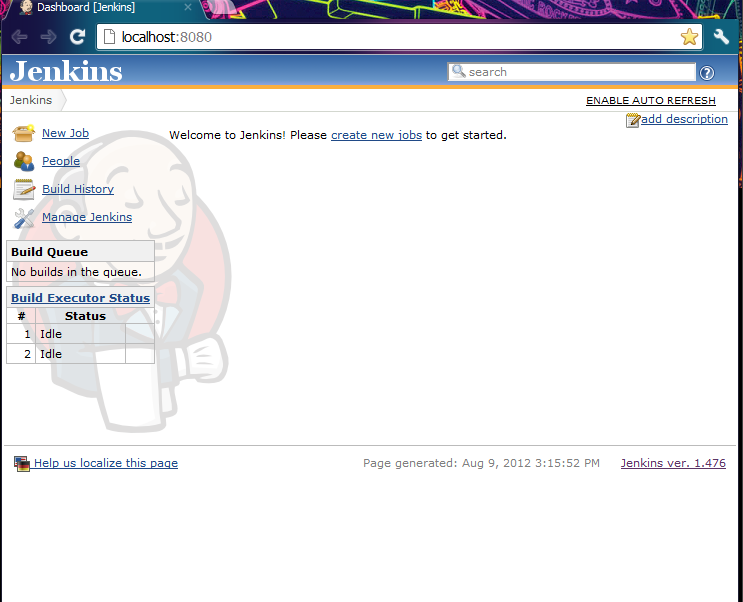
1. Download windows package from Jenkins home page.
2. Run the MSI.

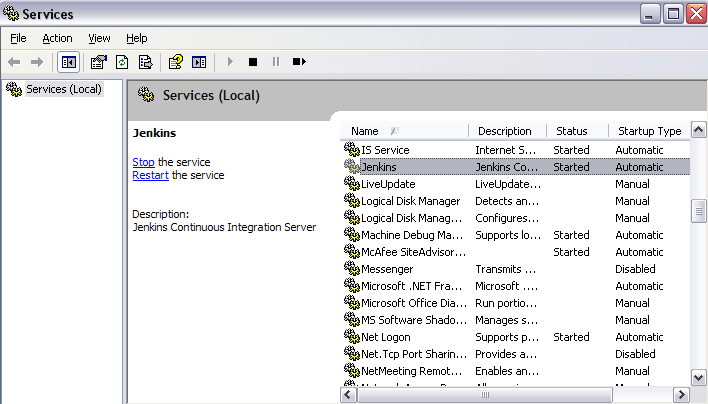
****

1. Click on next.

****

1. Choose the folder for installation.
2. Once the installation is completed on the windows server you should be able to see the web page opening with default port number “8080” and windows service created as below screenshots.

****

****

**Running a WAR file**

1. Download the WAR file from Jenkins home page .

2. Go to command prompt.

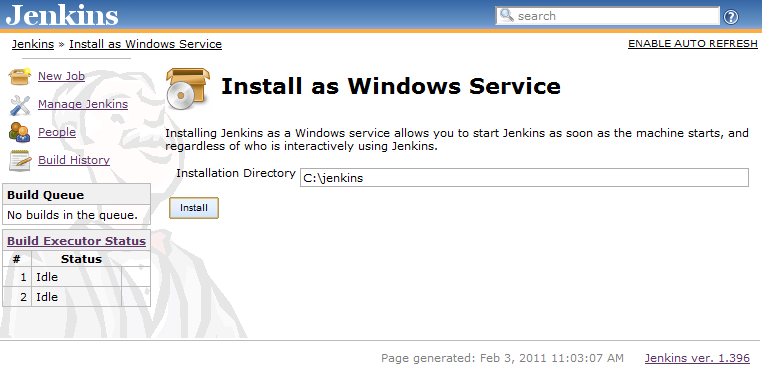
3. Run "java -jar jenkins.war". (You should be having JDK on the server to perform this action)

Now connect to Jenkins by going to the following URL http://<hostname>:8080/.

4. Once Jenkins is started this way, you can also make the Jenkins work based on windows service.

5. Go to Mange Jenkins ->look for the "Install as Windows Service" link in the "Manage Jenkins" page (requires [Microsoft .NET](http://www.microsoft.com/net/) framework **version 2.0**):  

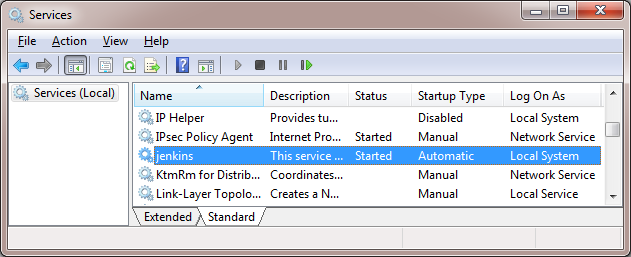

**NOTE**: The "Install as Windows Service" link indicated above will not appear if Jenkins is already running as a service on the machine. You can verifying that Jenkins is running as a service by launching the Services app from the Windows Start Menu and looking for a running service called "jenkins" among the list of all windows services running on the machine.

6.Clicking this link shows you the installation screen:  


7.Choose the directory where Jenkins shall be installed (directory must already exist. If not create it before hand). This will become JENKINS\_HOME and used to store data files and programs alike.  
Upon successful completion of the installation, you should see a page asking you to restart Jenkins.  


This is to re-launch Jenkins as a newly installed Windows service. When you click yes, you'll be asked to wait until the restart completes:  


|  |  |
| --- | --- |
| https://wiki.jenkins-ci.org/images/icons/emoticons/warning.gif | If a restart fails for some reason, check the output from Jenkins, which is stored in the installation directory that you specified.  For W2K8 R2 users you may need to add *C:\Windows\SysWOW64* to the PATH env var. |

At this point you can use the service manager to confirm that Jenkins is running as a service.  


## 6. Introducing Master Slave Setup on Jenkins

## Introduction:

Jenkins is an open-source continuous integration software tool written in the Java programming language for testing and reporting on isolated changes in a larger code base in real time. The software enables developers to find and solve defects in a code base rapidly and to automate testing of their builds. This allows to handle the project related activities such as Build download, deployment, setting up DB, configuration of application, initiating automation suite and providing meaningful reports

In any service or product based project, the application has to be deployed in different machines with different environments. In such case we can configure multiple nodes as slave machines in Jenkins and setup the QA infrastructure in slave machines without any human intervention. Jenkins works on a 'master->many slaves' principle. The master node is responsible for creating jobs and managing the slave nodes, where the jobs are actually performed. The master node is essentially doing management work and farming out preset jobs to the most appropriate site. Each slave then informs the master of the outcome and all of the job results are collated on the master node for easy viewing.

This document explains the step by step process to configure Master slave environment in Jenkins.

## Prerequisites:

Java SDK

.Net framework 3.5 above

Jenkins should be installed.

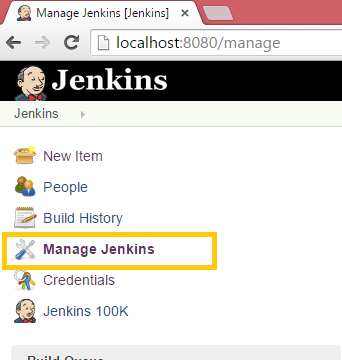
Node and Label Parameter Plug-in

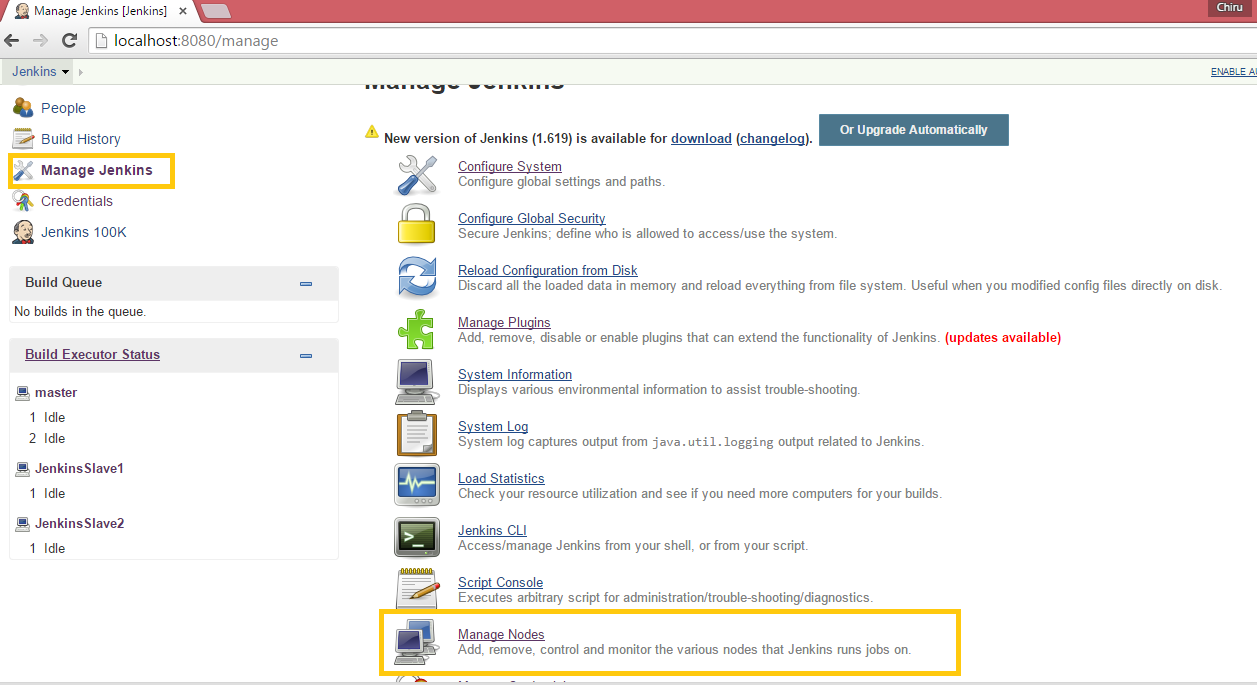
## Creating Node:

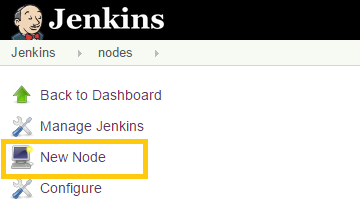
Go to Jenkins dashboard

Select manage Jenkins -> Manage Nodes

Select New Node from left side tab.







## Configuring Node:

Give the node name, example: JenkinsSlave1

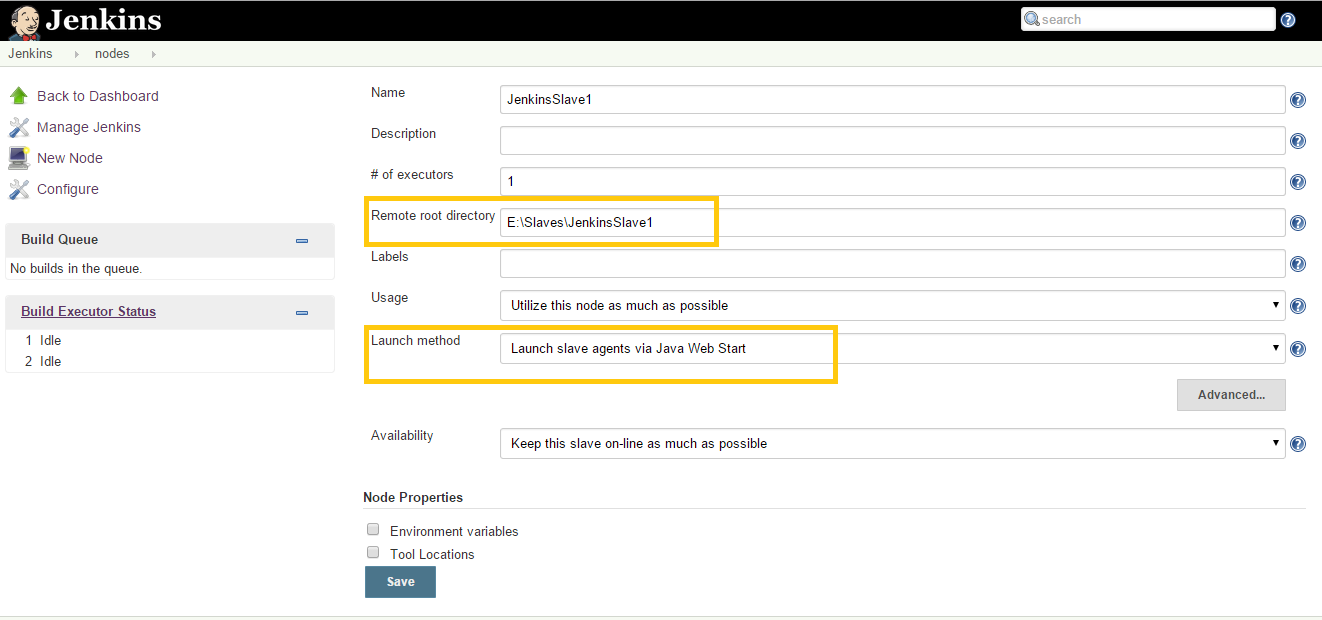
Select Dumb Slave option

Select the remote directory where the slave needs to be placed

Select the launch method “Launch slave agents via Java Web Start”

Save the option





## Launch the JNLP agent:

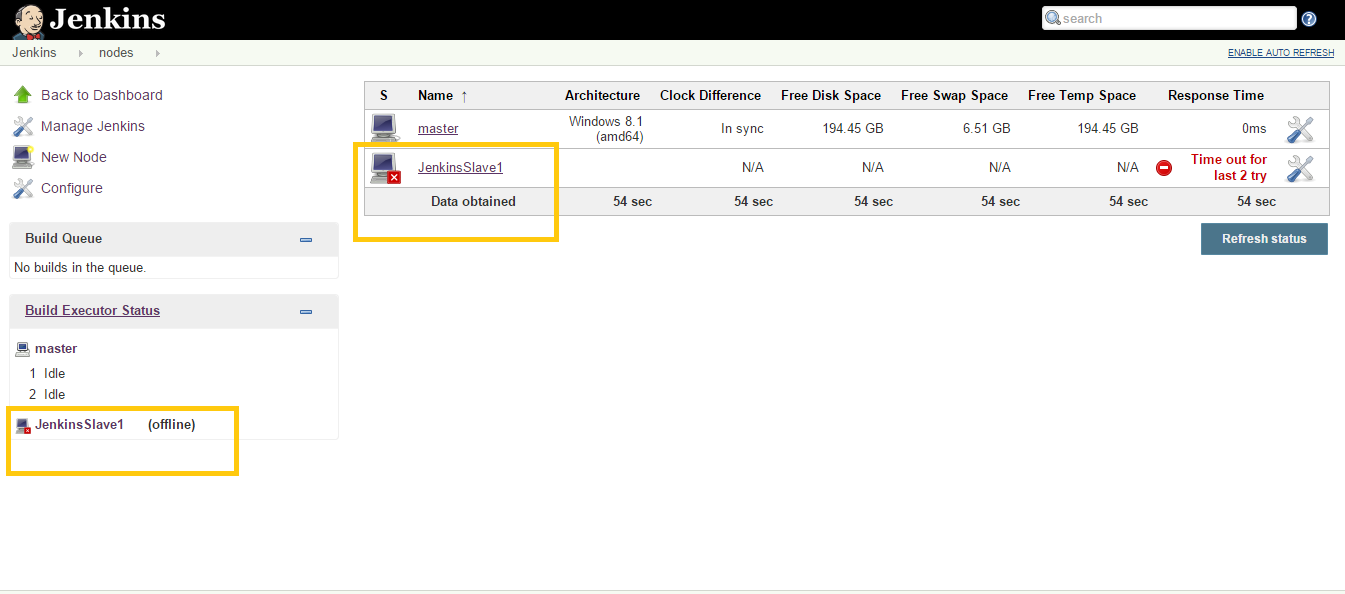
Go to Jenkins-> Nodes

You would be able to see the slave created but offline

Click on the Slave Node, you will see the Slave with different ways it could be connected.

Click Launch agent from browser on slave.

It will download the ‘slave-agent.jnlp’





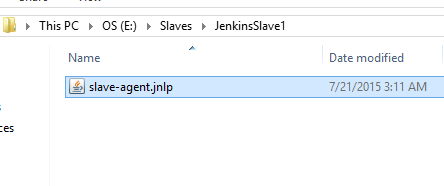
## Launch the node in remote directory:

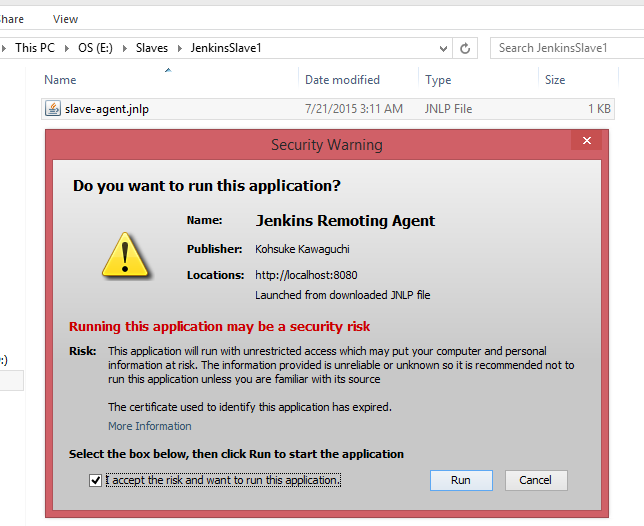
Copy the downloaded ‘slave-agent.jnlp’ to remote directory.

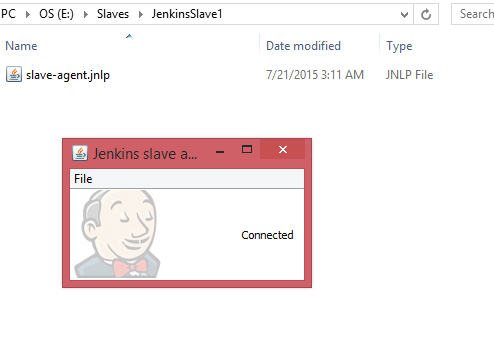
Double click on the file, select run by selecting the agreement.

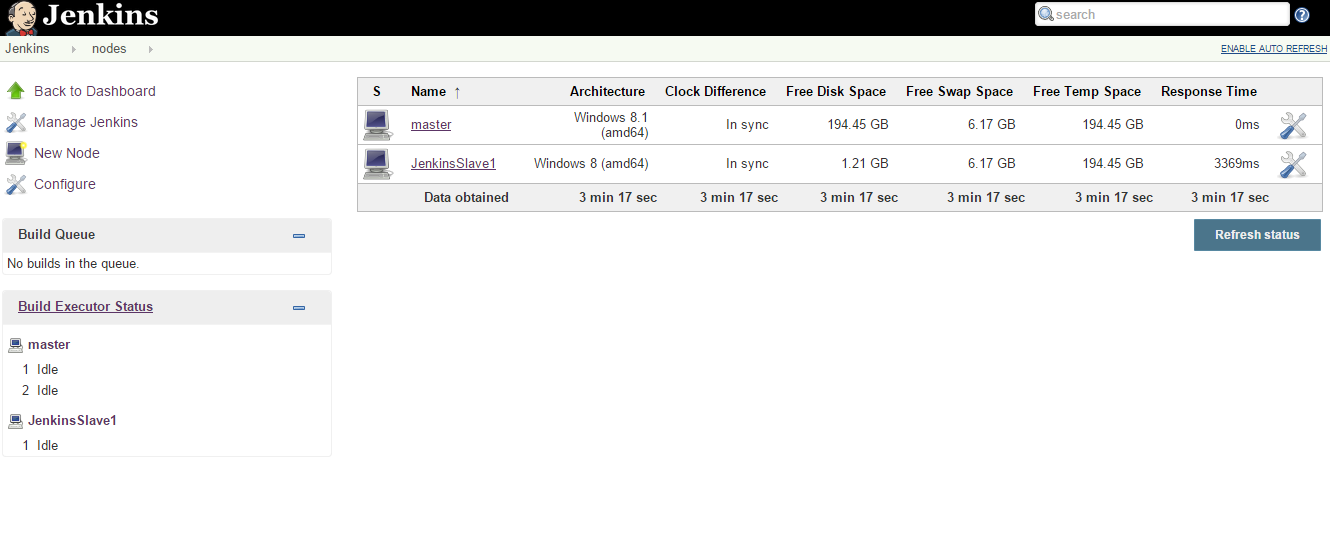
It will pop up a window with status connected.

Check the status of the node from master.









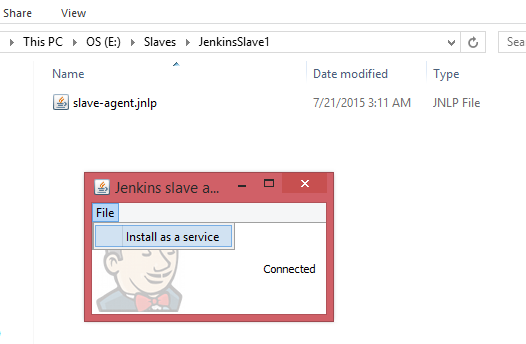
## Launch the slave as windows service

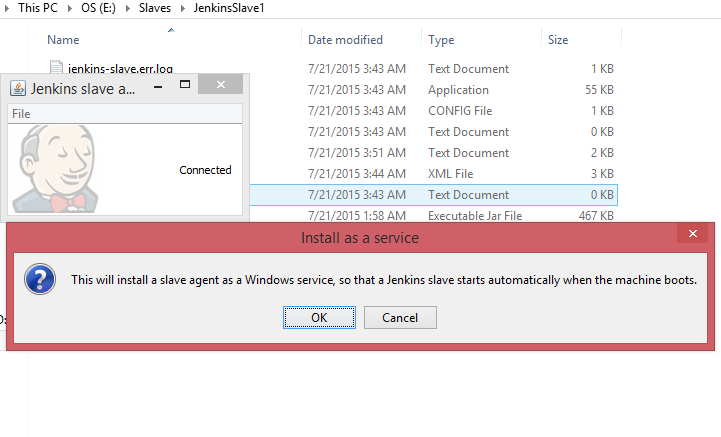
From the java window popped up select File-> Install as windows Service.

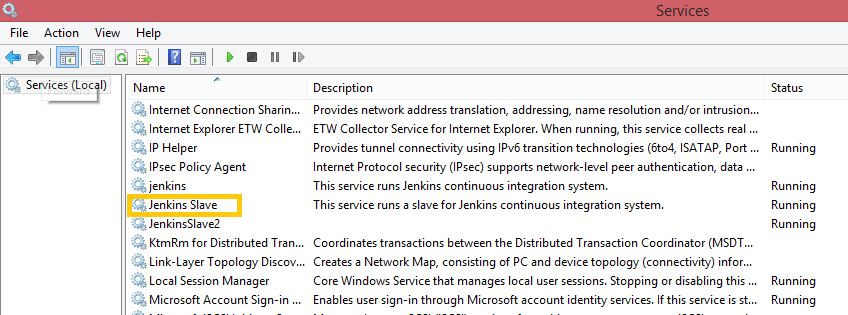
Select OK.

Now Slave is running as windows service.

Open services. MSc , you can see the service running for Jenkins Slave







There are many ways Slaves can be connected depending on the environment and the OS. Above one illustrates simple way of doing master slave setup in Jenkins.

## 7. Main components of Jenkins

1. New Job/Item
2. People
3. Manage Jenkins
4. Build History
5. **New Job/New Item:**Creating a new job is like integrating an action to be performed.

After click on the New Item few variants will be visible.

1. Build Freestyle project: This job is suitable for ANT, NANT.., and we can create underneath it irrespective of the platforms.
2. Maven 2/3 project: If we have a java project which performs the build activities using the Maven2 or version Maven3 (Invention of Apache foundation), we can opt for this job.
3. Build Multi configuration project: if you want have a job with multiple configurations. For e.g., for **Training** environment you need a build with specific version of dependent projects and for the **Performance** environment want to build the same code with latest of the dependent projects, we can opt for this job.
4. Monitor and external job: we can use this job for the automation purposes. If any tasks that are running in the windows scheduler or background running of the shell script for such cases you will put the triggers based upon your requirement.
5. **Pre build actions -** 
   1. **Steps to create a new Freestyle project**
   2. Click on New Item > Provide the Name of the JOB
   3. Select the radio button for Freestyle Software job
   4. Provide the description for the job (It is a good practice to provide the description about the project you are building).
   5. **Discard old build checkbox:**
   6. Selecting this option rotation of logs (old builds) can be restricted to specific no of days and max no of builds that can be stored in the build machine where you are running the Jenkins service.
   7. **Build is parameterized:**
   8. Any arguments that need to be passed as a parameter (file, string, password and choice) for a build we can enable the checkbox and use the argument anywhere within the job.
   9. **Disable Buildcheckbox:**
   10. This option, until you enable it back no one will be able to trigger/force the job.
   11. **Source Code Management section:**
   12. By default CVS and subversion SCM plugin’s will be installed, if you are using any other SCM tool other than above, like GIT and CLEARCASE you can able to find it here If you installed the respective plugin.
   13. For e.g. we are using the subversion, give the repository URL where your project located and local working copy location on your build machine and select the depth of the repository to perform the below actions. SVN update, fresh working copy and SVN update followed by SVN revert.
   14. **Build Triggers Section:**
   15. Build after other projects are builtIf there is any interdependent jobs where running the current job once the parent job completes and depends on the parent job stable, unstable and failure.
   16. **Build Periodically:**
   17. We are going present the cron tab syntax here, based on the timeframe job will schedule.
   18. **Poll SCM:**
   19. Same cron tab syntax where we presented in the build periodic option can be given here. Based on the given timeframe job will search for the SCM changes and if finds anything job will trigger.

**Note: D*ifference between Poll SCM and Build Periodically***

Poll SCM will check for the SCM changes at given time whereas Build periodically will trigger the builds based on the time frame you have given irrespective of the SCM changes.

1. **Build Section:**
   1. This is the step where you can do the compilation activity for your code. We can pass the build file and if we have any properties to it, can pass it.
   2. This section supports for Build tools like NANT, ANT, Maven and external jobs like running the batch scripts, Power shell scripts and shell scripts.
2. **Post Build Action:**
   1. After the main operations like compilation, packaging the artifacts and automation of activities using scripts actions performed on the build section, we can perform the post job actions if required.
   2. Jenkins supports few options like publishing the reports like HTML, MSTESTS, JUNIT test reports, violations in the code and sonar results and publishing the xml’s and most importantly email notifications to the respective APP DEV teams.
3. **People:**

In Jenkins we can find two kinds of user

1. Users who we gave access in global security section (which is under our control)
2. User who make changes in the SCM repositories (Jenkins SCM plugin e.g. subversion)

For e.g. we have configured the SVN url’s (let’s assume that we are using the poll SCM concept upon the repository change auto trigger of the build will happen), Jenkins will automatically pick up the id from the subversion and display the same id in the people section.

1. **Manage Jenkins**

Manage Jenkins was an amalgamation of the main configuration options in Jenkins server. Below are the few options and main usage about that

1. **Configure system:**

This contains the settings of the whole system where you’ll configure once and recall the configuration when we are configuring the actual jobs. E.g. different version of the build tool and various versions of development kits, saving the passwords for global use, setting the environment variables and some external plug in global settings.

1. **Configure global security:**

This is a place we will mainly configure the groups and individuals who need access from where the control has to be taken from.During the setup in access control where we can tag our Jenkins to active directory, we can even connect the Jenkins to the LDAP server where our authentication is maintained and can even create the Jenkins database to give the access.

1. **Manage plugins:**

Since Jenkins is plugin based architecture; all the respective work that needs to be was already done using the plugins, in this option we will add/ remove/ update the compatibility plugins.

Jenkins provides the feasibility to look for the required plug in in the available section with the Action header.

1. **Manage Nodes:**

Jenkins can act as both stand-alone based or server-slave based model. If we have the multiple jobs which are keeping the master busy all the time, to make the load distributed among the slaves (agent) in remote machines. We can configure the architecture from this option.

**System log:**  
By default Jenkins saves all the logs to the Jenkins.lo.er file in the installation using the java.util.logging class. If we want to have the logs based on the plugin changes can go with the Hudson.pluginManager class or if we need the logs to see the builds can use the **Hudson.model.build** classes.

1. **Manage credentials:** This sub section contains the configuration of the saving the authentication of credentials that are used globally while creating the jobs.

## 8. Plugins

Before to start with Jenkins Plugins, Jenkins supports plugins which allow Jenkins to be extended to meet specific needs of individual projects.

One of Jenkins' main strengths is its plugins. Hundreds are already available and brand new plugins become available every week or so. As such, there's surely one or several plugins which cover your needs.

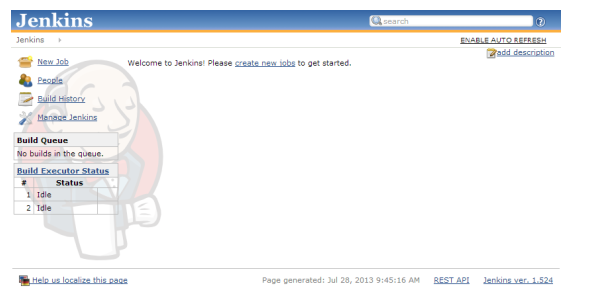
**What Can Plugins Do?**

Jenkins defines extensibility points, which are interfaces or abstract classes that model an aspect of a build system. Those interfaces define contracts of what need to be implemented, and Jenkins allows plugins to contribute those implementations.

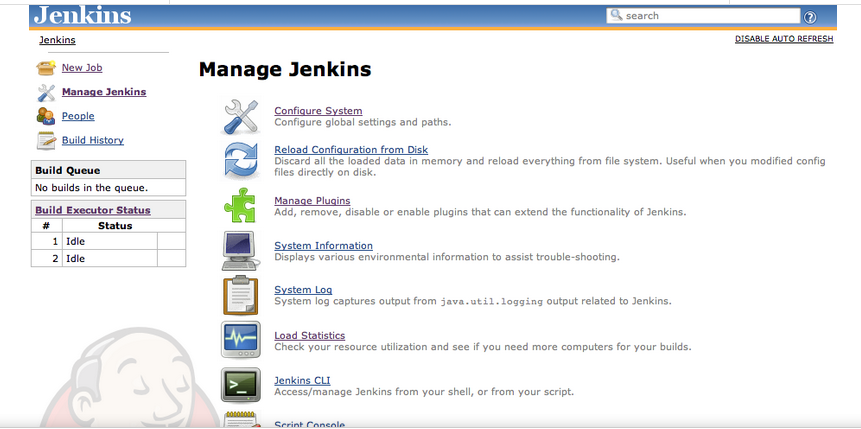
**Installing Plugins**

There is a vast amount of plugins available for Jenkins. Ranging from build tools to FTP and SSH publishers to test coverage reports to Chuck Norris. We're going to install a plugin that keeps track of the disk space used by different builds and jobs. This makes it easier to look for space eaters. To get started, go to Manage Jenkins again. And click on Manage Plugins:

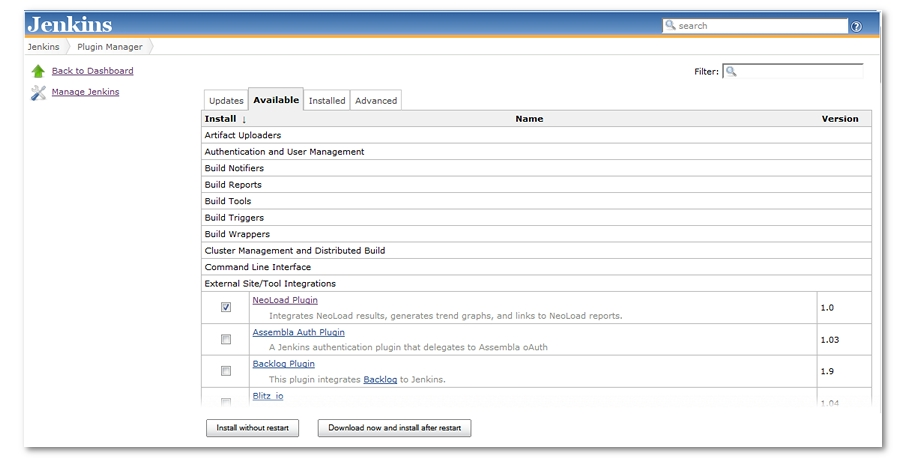
Manage Plugins Screen:

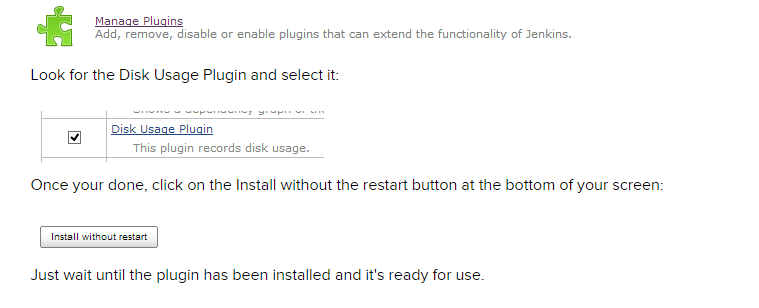


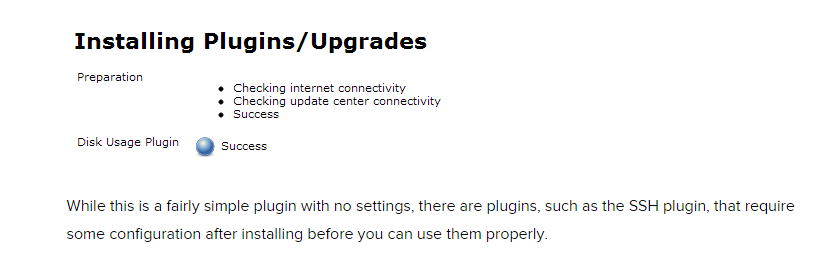
Installation Management Screen:



Manage Plugins Screen:







Jenkins is a vastly used continuous integration tool for the IT industry which is built and used around plugins. Its core is the Jenkins tool and there are 100’s of plugins available to enhance its power and usability.

## 9. GlobalSecurity

Immediately after installation, Jenkins will allow anyone to run anything as user Jenkins, which is bad. This page shows you how to set up basic security using the Configure Global Security page.

The Configure Global Security page has two sections in which you:

* Set the security realm to determine who is allowed access
* Set the authorization to determine what each user is allowed to do

## 10. Security Realm

First, establish the user authentication method. For smaller, more informal installations, you can use Jenkins' own user database. For enterprise installations, you will want to use your corporate service, which allows users to log in to Jenkins with their usual username and password.

## 11. Configuring Security in Jenkins

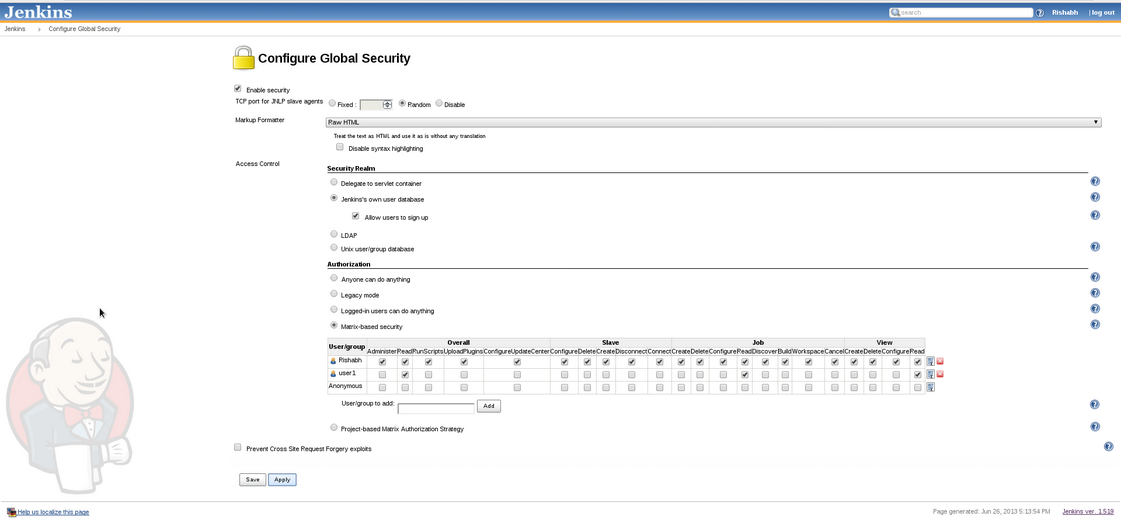
To configure user security and permissions, we should first have some users in Jenkins. To create new users in Jenkins Follow these steps.-

**Manage Jenkins’** Page.

This was a very basic configuration of Jenkins Security where Jenkins manages its own user database. Jenkins allows far more options than just managing its own user database.

* If you have an LDAP server configured in your organization, you can fill in the LDAP server information by selecting ‘LDAP’ in place of ‘Jenkins own user Database’. Thus users from LDAP can login to Jenkins and you need not configure each account manually.
* Similarly, on Windows, you can install a plugin for Active Directory Plugin. Active Directory is similar to LDAP in Linux.
* If you want the users of the machine on which Jenkins is deployed should be able to login to Jenkins, you can use ‘Jenkins user/group database’ in place of ‘Jenkins own user Database’.

1. From the ‘Manage Jenkins’ page, click on Configure Global Security.
2. Check ‘Enable Security’
3. In the Access Control section, there are many options available for configuration.
4. Select Matrix Based Security
5. Add existing users to the user matrix and grant/revoke permissions as required.
6. Clicks save.



Proper configuration of Jenkins Security will ensure smooth and safe run of Jenkins without any surprises.