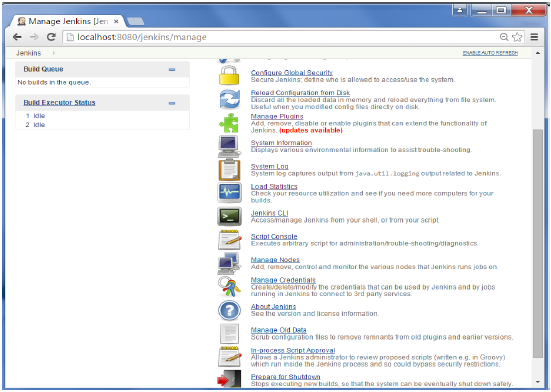
**Slave and Node Setup**

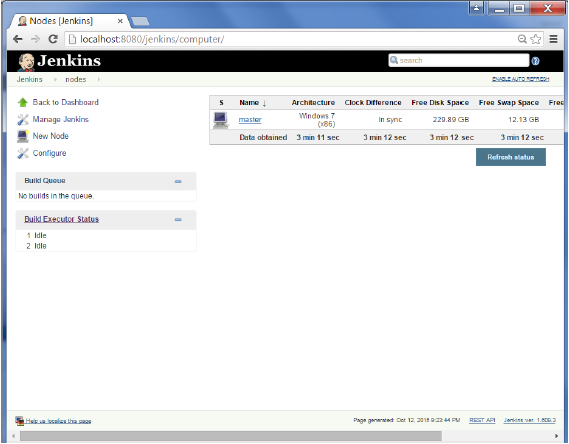
**MTR Review – Metrics and Trends**

**Backup**

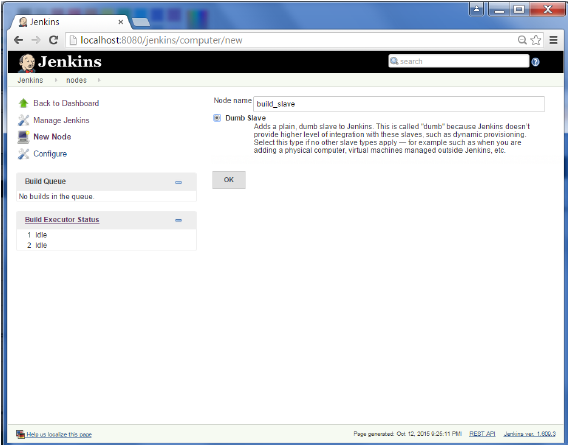
**Step 1** − Go to the Manage Jenkins section and scroll down to the section of Manage Nodes.



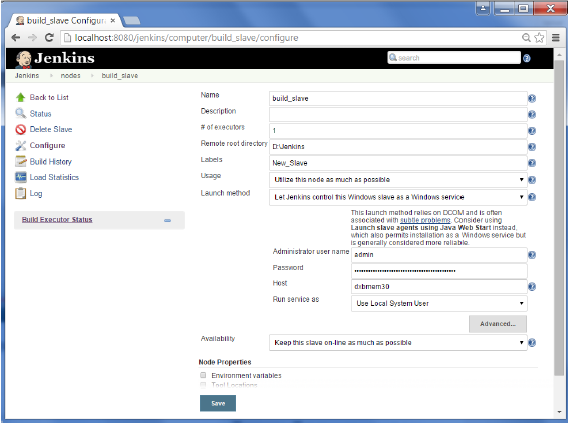
**Step 2** − Click on New Node



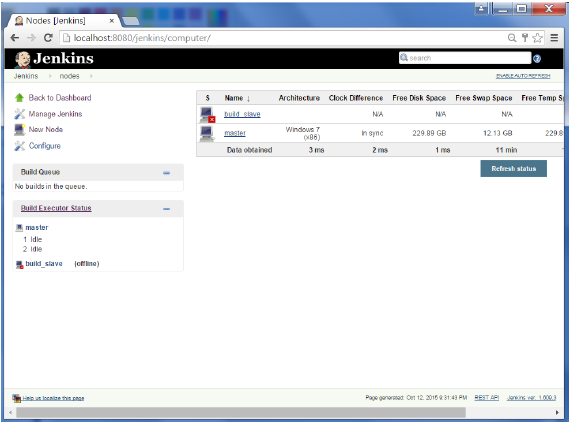
**Step 3** − Give a name for the node, choose the Dumb slave option and click on Ok.

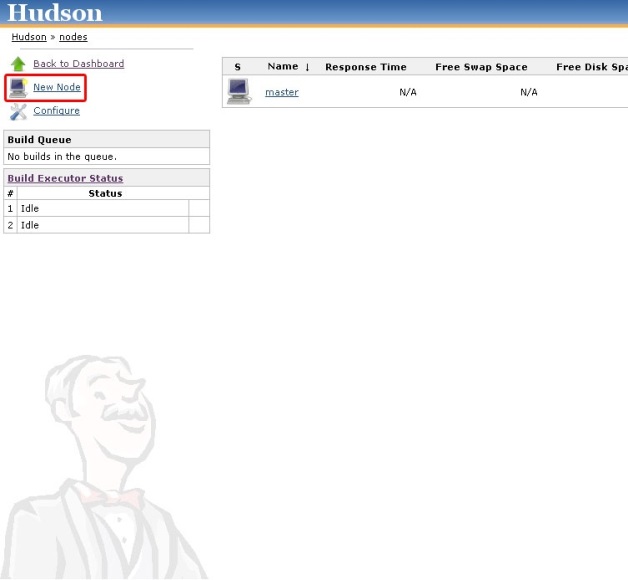
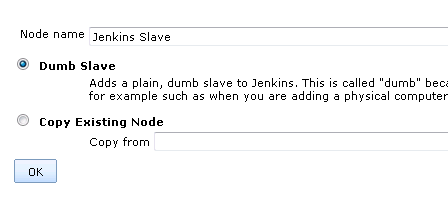
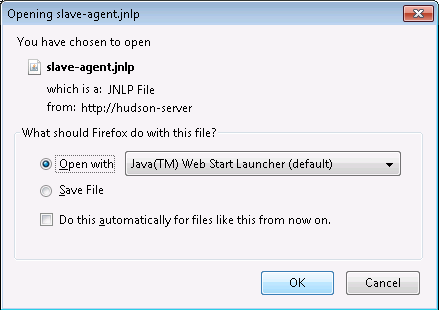
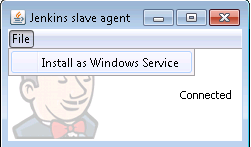


**Step 4** − Enter the details of the node slave machine. In the below example, we are considering the slave machine to be a windows machine, hence the option of “Let Jenkins control this Windows slave as a Windows service” was chosen as the launch method. We also need to add the necessary details of the slave node such as the node name and the login credentials for the node machine. Click the Save button. The Labels for which the name is entered as “New\_Slave” is what can be used to configure jobs to use this slave machine.



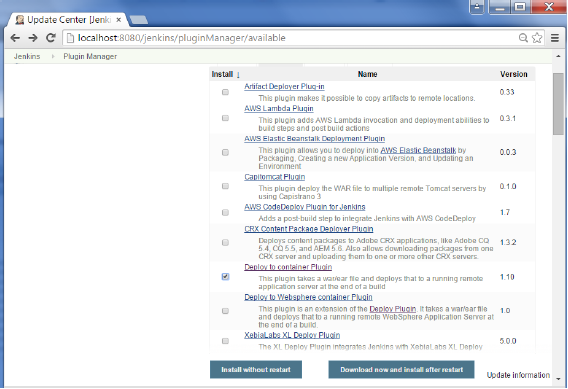
Once the above steps are completed, the new node machine will initially be in an offline state, but will come online if all the settings in the previous screen were entered correctly. One can at any time make the node slave machine as offline if required.



1. On your master machine go to **Manage Jenkins** > **Manage Nodes**.  
   
2. **New Node** --> **Enter Node Name**.
3. **Select Dumb Slave** --> Press **OK**.  
   
4. Fill out the following:
   1. Set a **number of executors** (one or more) as needed.
   2. Set a **Remote FS Root**, a home directory for the master on the slave machine.
      1. For a *Windows slave*, use something like: "C:\Jenkins\"
      2. ***TODO***: add details.
   3. Select the appropriate **Usage** setting:
      1. For an additional worker: *Utilize this slave as much as possible*
      2. For specialized jobs: *Leave this machine for tied jobs only*
   4. **Launch Method**:
      1. An easy way to control a Windows slave is by using *Launch slave agents via Java Web Start*  (Recommended for Windows)
      2. ***TODO***: add steps for other methods.
   5. **Availability** --> *Keep this slave online as much as possible*
      1. ***TODO:*** add details for each option.
   6. Press **OK**.  
      
5. Now you need to connect your slave machine to the master using the following steps.
   1. Open a browser on the **slave machine** and go to the **Jenkins master server** url ([http://yourjenkinsmaster:8080](http://yourjenkinsmaster:8080/)).
   2. Go to **Manage Jenkins** > **Manage Nodes**, Click on the newly created slave machine. You will need to login as someone that has the "Connect" Slave permission if you have configured global security.
   3. Click on the **Launch** button to launch agent from browser on slave.  
      
   4. Run the program.  
      
      1. If you encounter connection issue, then you could enlarge the popup windows to see the master **port used** and check your network configuration (firewall, port forward, ...).
   5. Now you should see the Slave machine connected under **Nodes**.
6. If you want the service to run on start-up of the slave machine do the following (Windows only directions):
   1. In the Slave agent program running on your slave machine, click **File** --> **Install as Windows Service.**  
      
   2. **Start**, type Services and Select the **Services** program.
   3. Find **Jenkins Slave** in the list, Double click to open.
   4. Select **Startup type** --> **Automatic**.
   5. Go to the **Log On** tab, change the **Log on as** to a user of your choice (Special user account Jenkins recommended).
   6. Make sure that auto login is set for the slave machine for the user account, then the VM (or physical computer) should connect and be available when needed.

Automated Deployment

**Step 1** − Go to Manage Jenkins → Manage Plugins. Go to the Available section and find the plugin “Deploy to container Plugin” and install the plugin. Restart the Jenkins server.



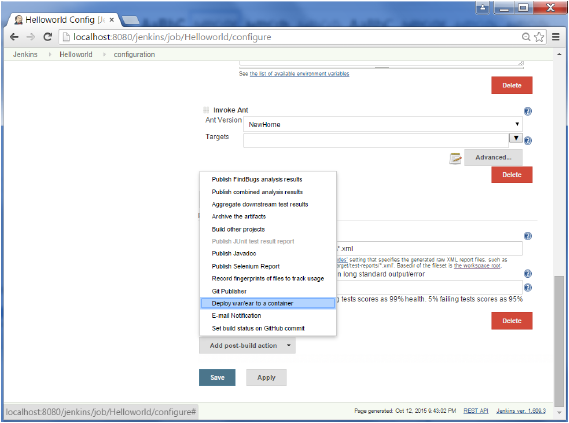
This plugin takes a war/ear file and deploys that to a running remote application server at the end of a build.

Tomcat 4.x/5.x/6.x/7.x

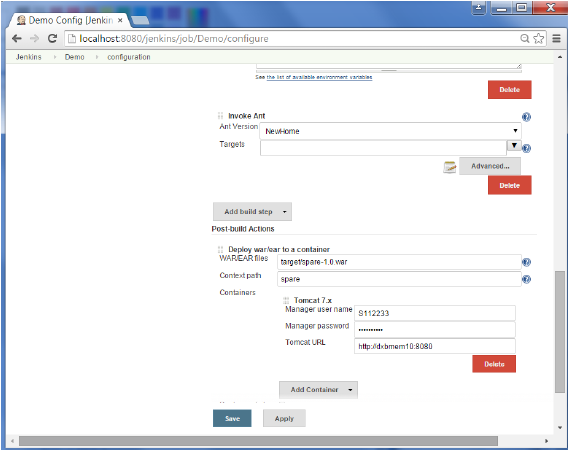
JBoss 3.x/4.x

Glassfish 2.x/3.x

**Step 2** − Go to your Build project and click the Configure option. Choose the option “Deploy war/ear to a container”



**Step 3** − In the Deploy war/ear to a container section, enter the required details of the server on which the files need to be deployed and click on the Save button. These steps will now ensure that the necessary files get deployed to the necessary container after a successful build.



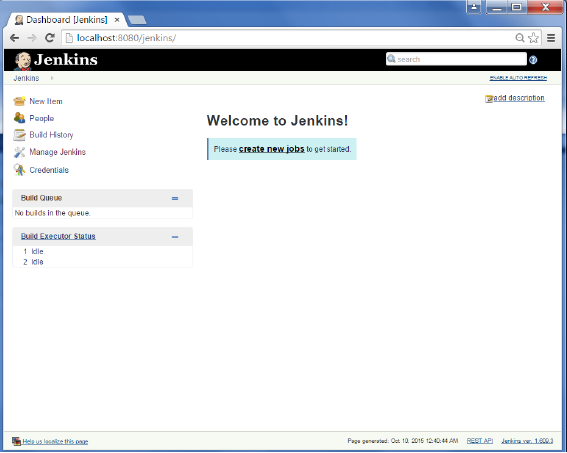
Metrics & Trends

There are various plugins which are available in Jenkins to showcase metrics for builds which are carried out over a period of time. These metrics are useful to understand your builds and how frequently they fail/pass over time. As an example, let’s look at the ‘Build History Metrics plugin’.

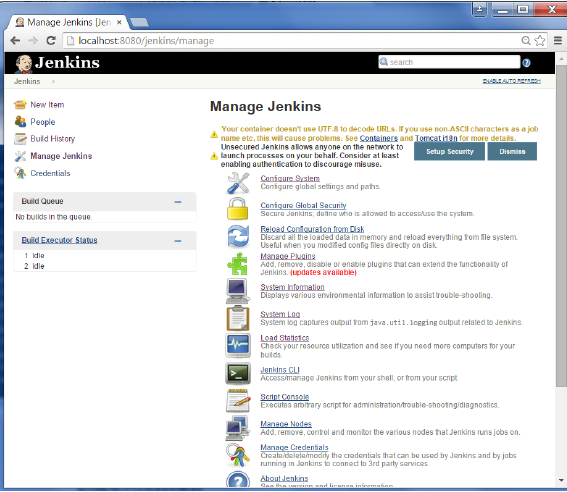
This plugin calculates the following metrics for all of the builds once installed

* Mean Time To Failure (MTTF)
* Mean Time To Recovery (MTTR)
* Standard Deviation of Build Times

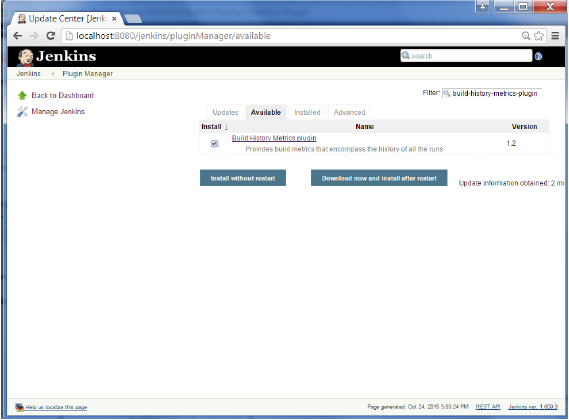
**Step 1** − Go to the Jenkins dashboard and click on Manage Jenkins



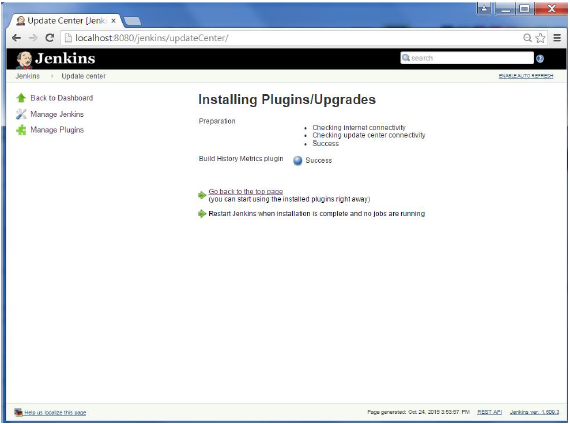
**Step 2** − Go to the Manage Plugins option.



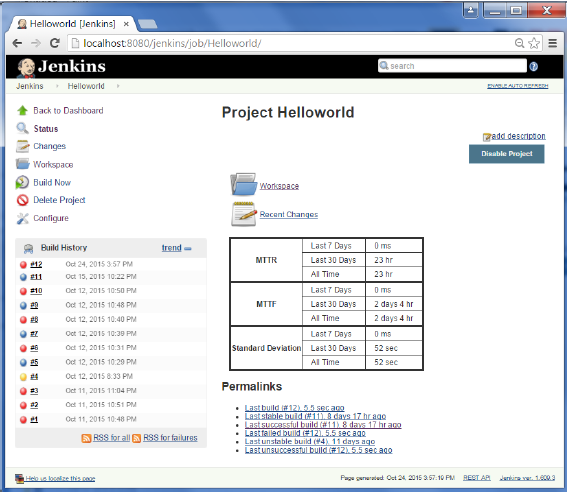
**Step 3** − Go to the Available tab and search for the plugin ‘Build History Metrics plugin’ and choose to ‘install without restart’.



**Step 4** − The following screen shows up to confirm successful installation of the plugin. Restart the Jenkins instance.

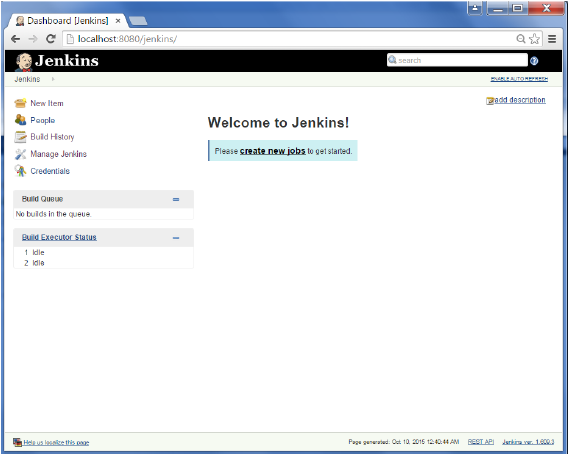


When you go to your Job page, you will see a table with the calculated metrics. Metric’s are shown for the last 7 days, last 30 days and all time.

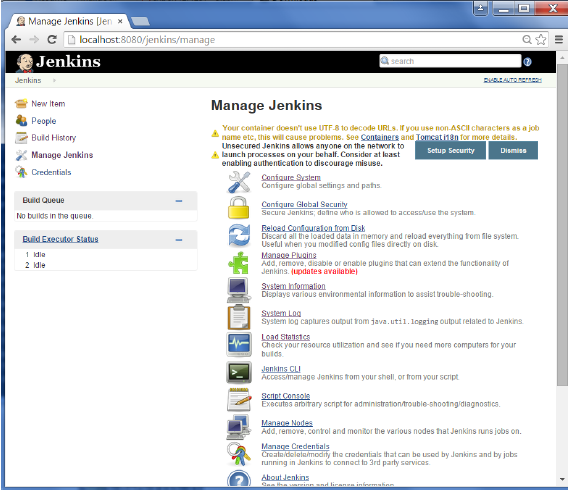


To see overall trends in Jenkins, there are plugins available to gather information from within the builds and Jenkins and display them in a graphical format. One example of such a plugin is the ‘Hudson global-build-stats plugin’. So let’s go through the steps for this.

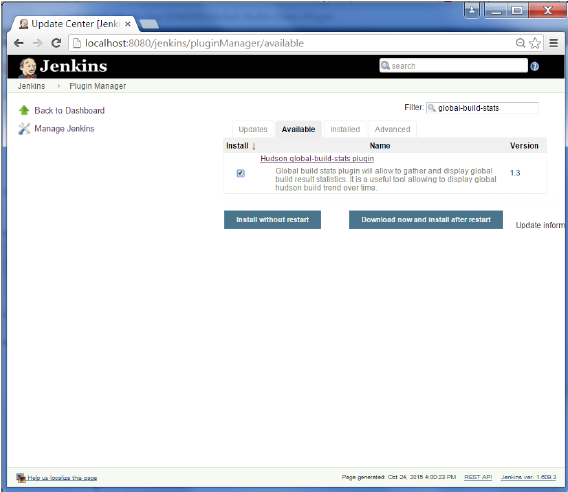
**Step 1** − Go to the Jenkins dashboard and click on Manage Jenkins



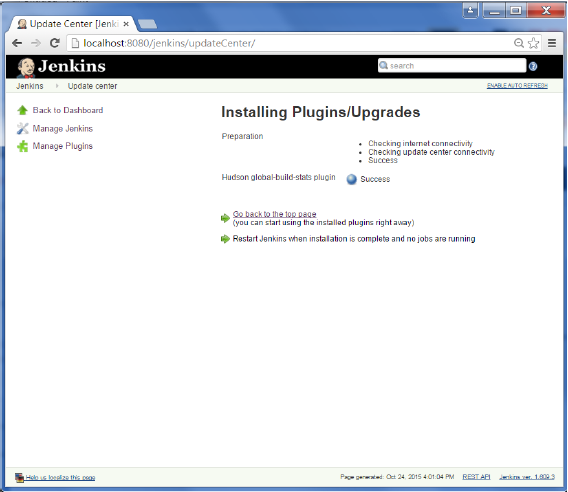
**Step 2** − Go to the Manage Plugins option



**Step 3** − Go to the Available tab and search for the plugin ‘Hudson global-build-stats plugin’ and choose to ‘install without restart’.



**Step 4** − The following screen shows up to confirm successful installation of the plugin. Restart the Jenkins instance.

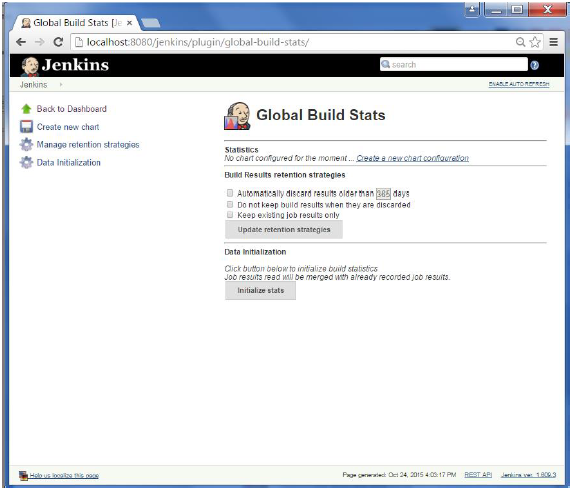


To see the Global statistics, please follow the Step 5 through 8.

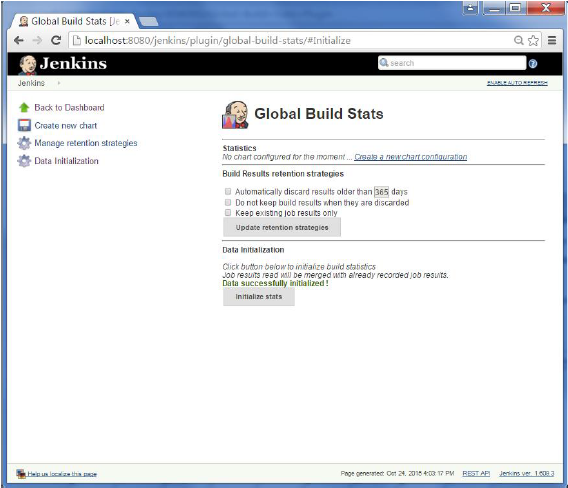
**Step 5** − Go to the Jenkins dashboard and click on Manage Jenkins. In the Manage Jenkins screen, scroll down and now you will now see an option called ‘Global Build Stats’. Click on this link.



**Step 6** − Click on the button ‘Initialize stats’. What this does is that it gather’s all the existing records for builds which have already been carried out and charts can be created based on these results.



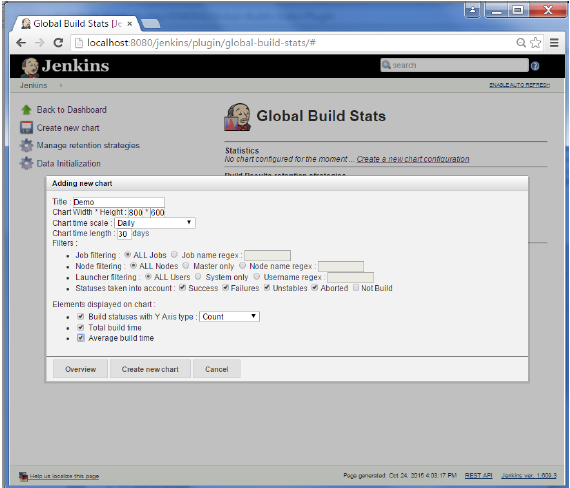
**Step 7** − Once the data has been initialized, it’s time to create a new chart. Click on the ‘Create new chart’ link.



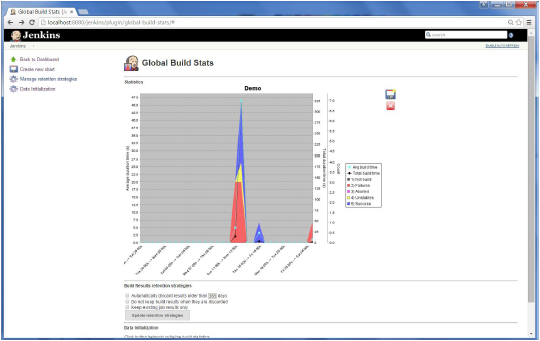
**Step 8** − A pop-up will come to enter relevant information for the new chart details. Enter the following mandatory information

* Title – Any title information, for this example is given as ‘Demo’
* Chart Width – 800
* Chart Height – 600
* Chart time scale – Daily
* Chart time length – 30 days

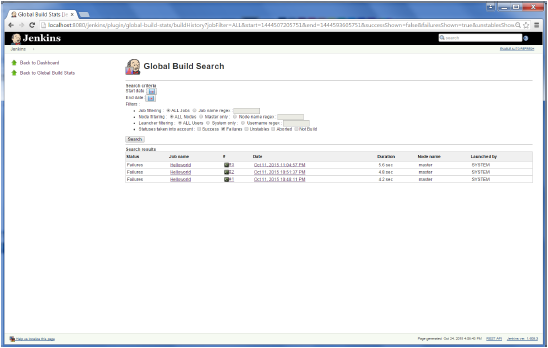
The rest of the information can remain as it is. Once the information is entered, click on Create New chart.



You will now see the chart which displays the trends of the builds over time.

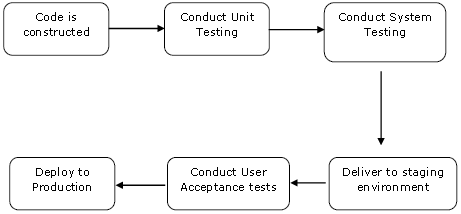


If you click on any section within the chart, it will give you a drill down of the details of the job and their builds.

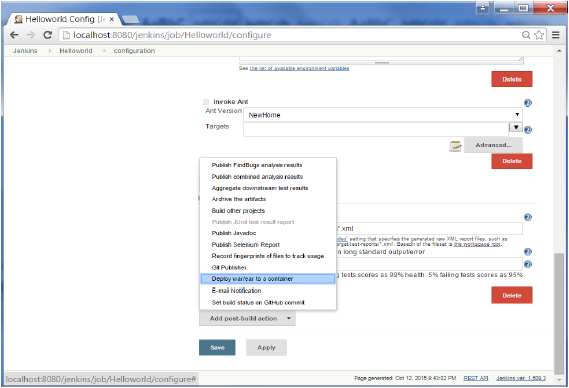


# Continuous Deployment

Jenkins provides good support for providing continuous deployment and delivery. If you look at the flow of any software development through deployment, it will be as shown below.



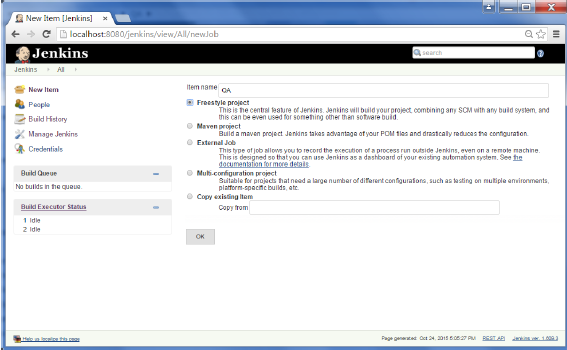
The main part of Continuous deployment is to ensure that the entire process which is shown above is automated. Jenkins achieves all of this via various plugins, one of them being the “Deploy to container Plugin” which was seen in the earlier lessons.



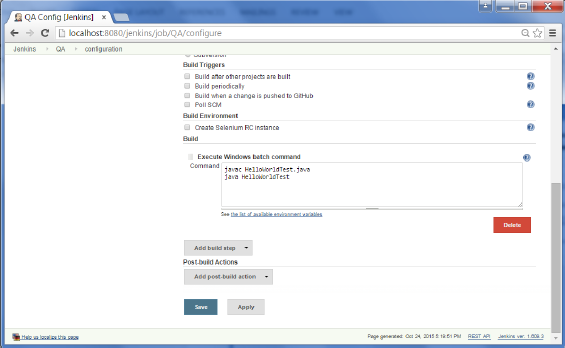
There are plugins available which can actually give you a graphical representation of the Continuous deployment process. But first lets create another project in Jenkins, so that we can see best how this works.

Let’s create a simple project which emulates the QA stage, and does a test of the Helloworld application.

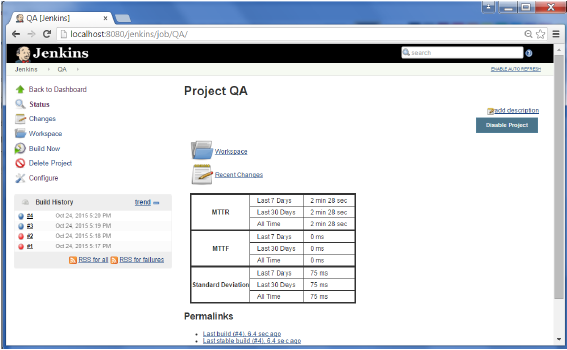
**Step 1** − Go to the Jenkins dashboard and click on New Item. Choose a ‘Freestyle project’ and enter the project name as ‘QA’. Click on the Ok button to create the project.



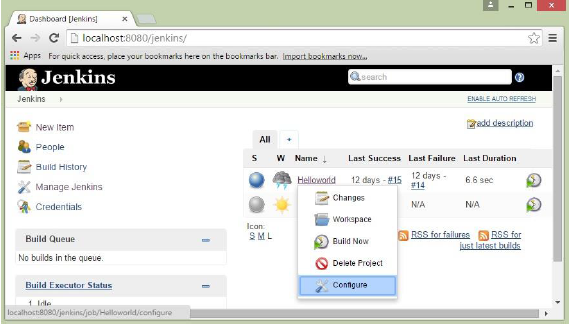
**Step 2** − In this example, we are keeping it simple and just using this project to execute a test program for the Helloworld application.



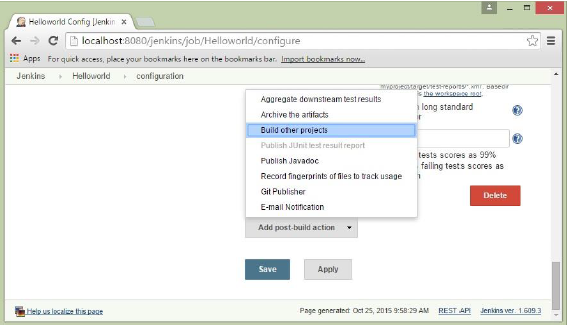
So our project QA is now setup. You can do a build to see if it builds properly.



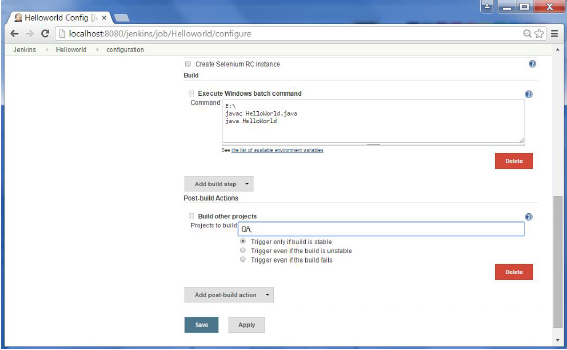
**Step 3** − Now go to you Helloworld project and click on the Configure option



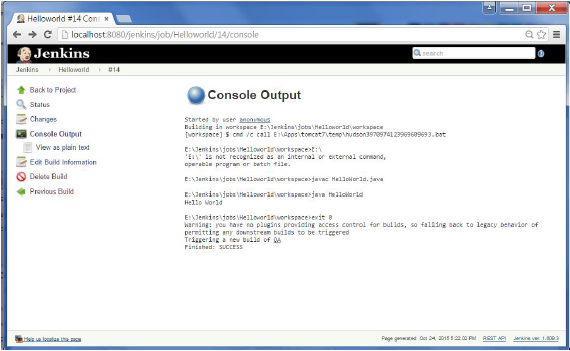
**Step 4** − In the project configuration, choose the ‘Add post-build action’ and choose ‘Build other projects’



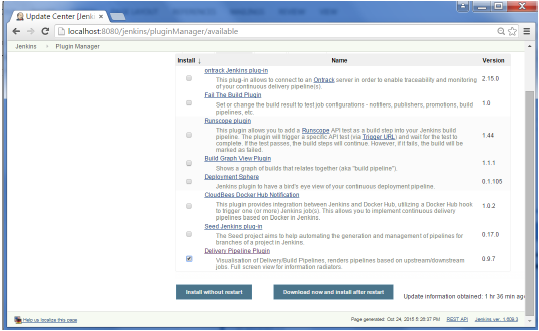
**Step 5** − In the ‘Project to build’ section, enter QA as the project name to build. You can leave the option as default of ‘Trigger only if build is stable’. Click on the Save button.



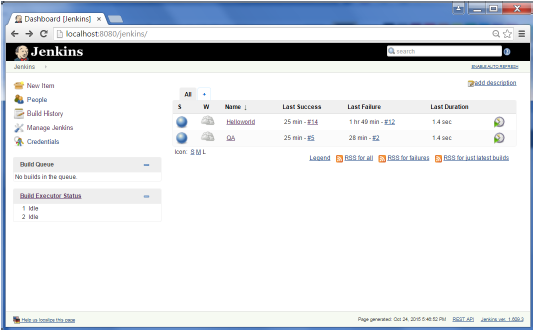
**Step 6** − Build the Helloworld project. Now if you see the Console output, you will also see that after the Helloworld project is successfully built, the build of the QA project will also happen.



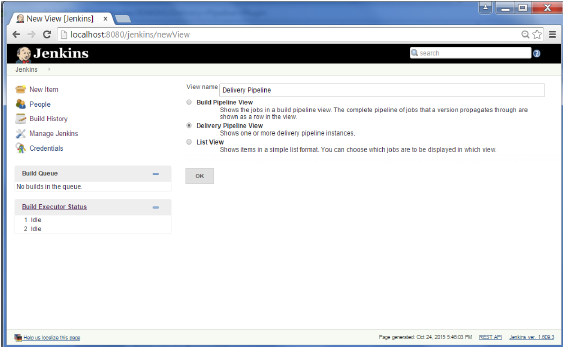
**Step 7** − Let now install the Delivery pipeline plugin. Go to Manage Jenkins → Manage Plugin’s. In the available tab, search for ‘Delivery Pipeline Plugin’. Click On Install without Restart. Once done, restart the Jenkins instance.



**Step 8** − To see the Delivery pipeline in action, in the Jenkins Dashboard, click on the + symbol in the Tab next to the ‘All’ Tab.

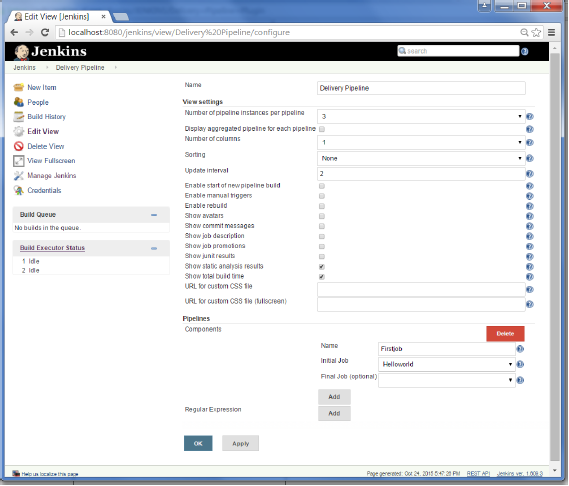


**Step 9** − Enter any name for the View name and choose the option ‘Delivery Pipeline View’.

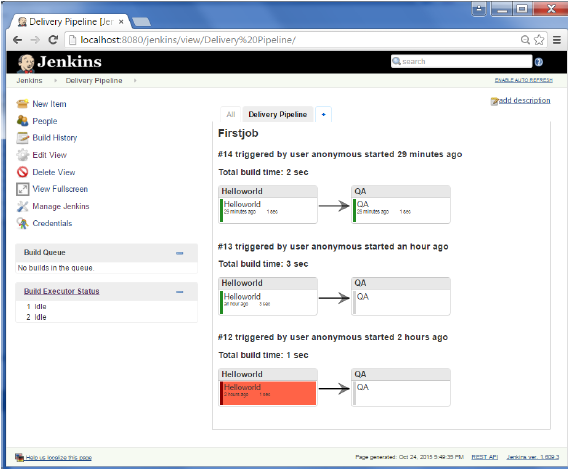


**Step 10** − In the next screen, you can leave the default options. One can change the following settings −

* Ensure the option ‘Show static analysis results’ is checked.
* Ensure the option ‘Show total build time’ is checked.
* For the Initial job – Enter the Helloworld project as the first job which should build.
* Enter any name for the Pipeline
* Click the OK button.

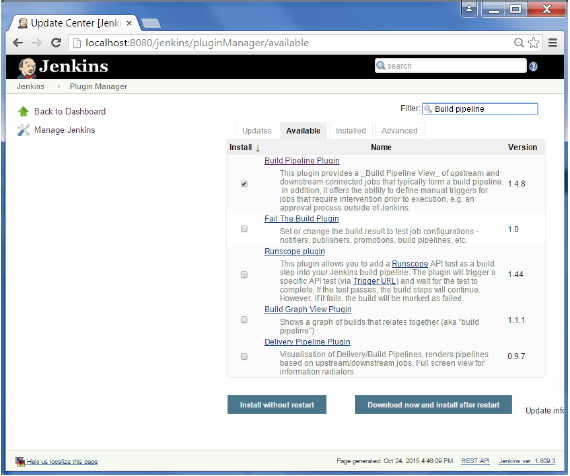


You will now see a great view of the entire delivery pipeline and you will be able to see the status of each project in the entire pipeline.

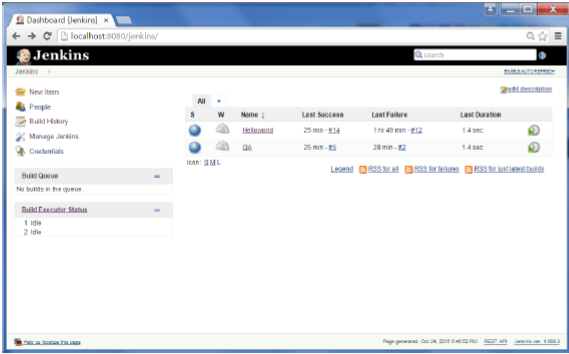


Another famous plugin is the **build pipeline plugin**. Let’s take a look at this.

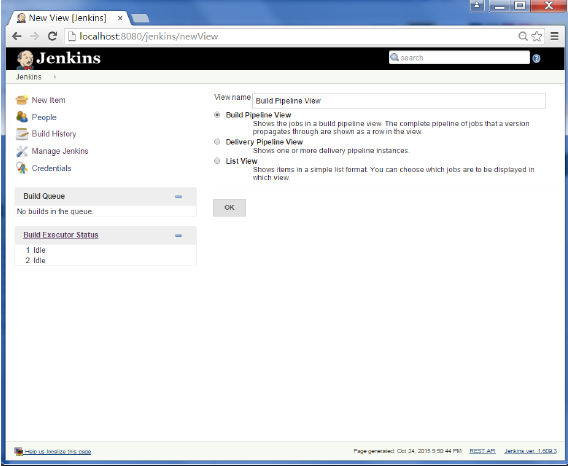
**Step 1** − Go to Manage Jenkins → Manage Plugin’s. In the available tab, search for ‘Build Pipeline Plugin’. Click On Install without Restart. Once done, restart the Jenkins instance.



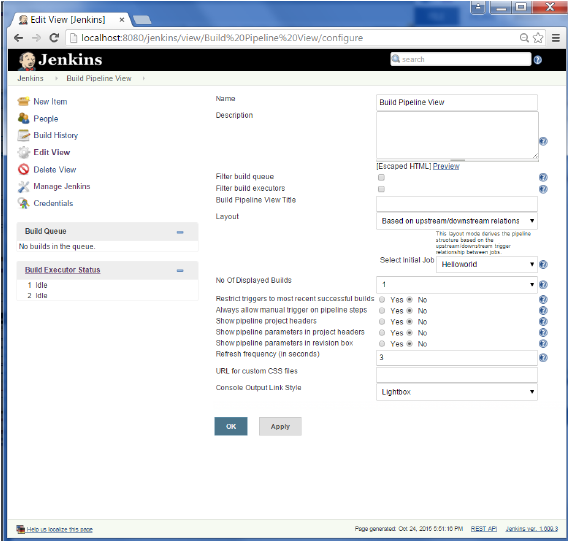
**Step 2** − To see the Build pipeline in action, in the Jenkins Dashboard, click on the + symbol in the Tab next to the ‘All’ Tab.



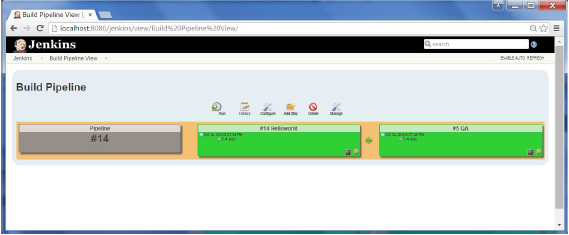
**Step 3** − Enter any name for the View name and choose the option ‘Build Pipeline View’.



**Step 4** − Accept the default settings, just in the Selected Initial job, ensure to enter the name of the Helloworld project. Click on the Ok button.



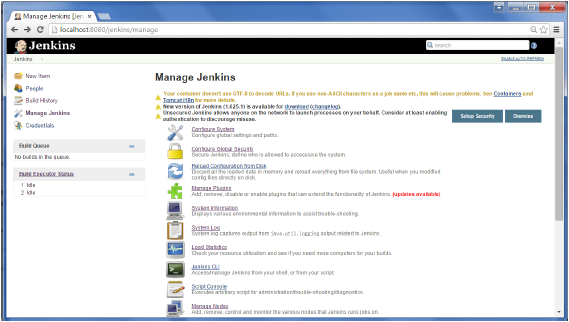
You will now see a great view of the entire delivery pipeline and you will be able to see the status of each project in the entire pipeline.



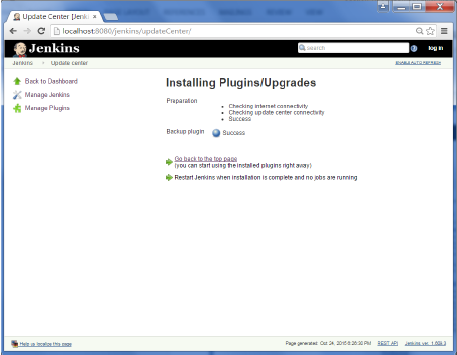
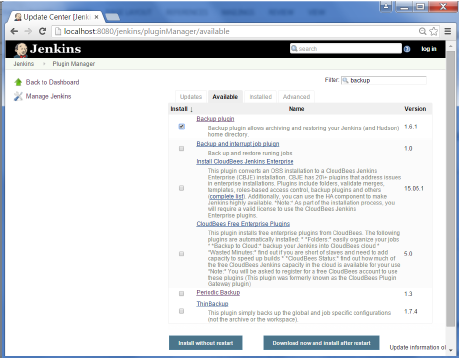
# Backup Plugin

Jenkins has a backup plugin which can used to backup critical configuration settings related to Jenkins. Follow the steps given below to have a backup in place.

**Step 1** − Click on Manage Jenkins and choose the ‘Manage Plugins’ option.



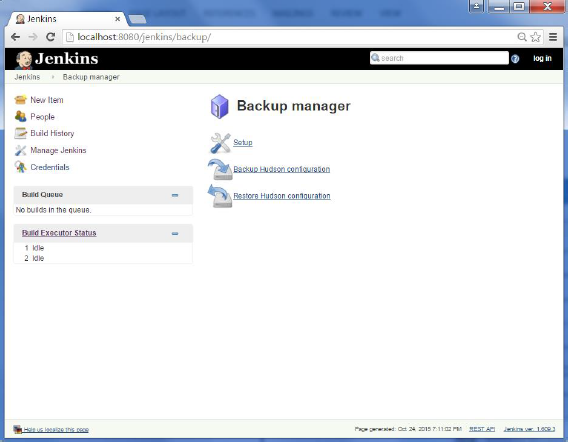
**Step 2** − In the available tab, search for ‘Backup Plugin’. Click On Install without Restart. Once done, restart the Jenkins instance



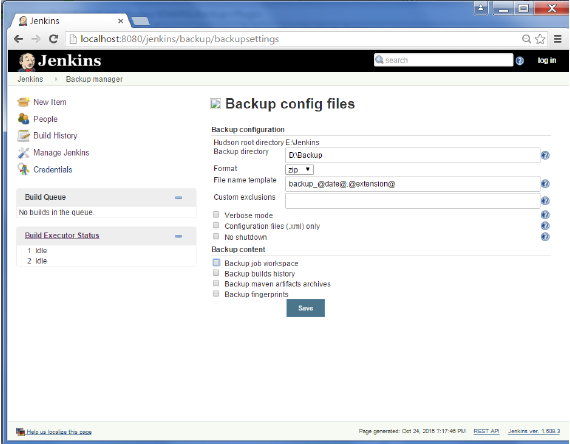
**Step 3** − Now when you go to Manage Jenkins, and scroll down you will see ‘Backup Manager’ as an option. Click on this option.



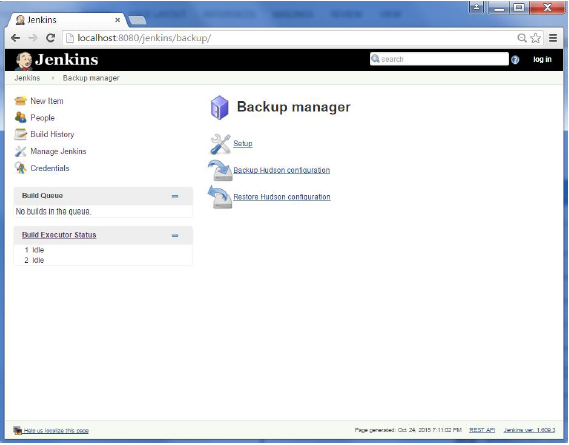
**Step 4** − Click on Setup.



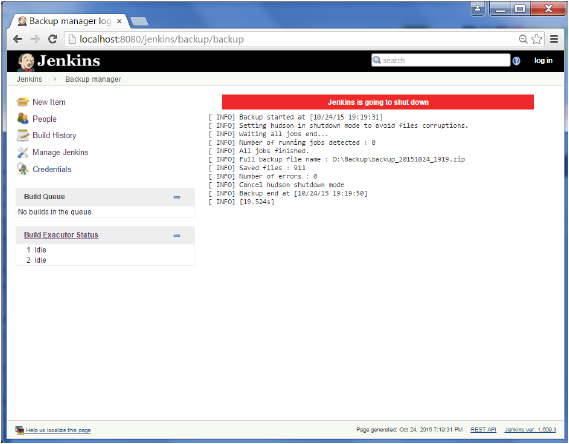
**Step 5** − Here, the main field to define is the directory for your backup. Ensure it’s on another drive which is different from the drive where your Jenkins instance is setup. Click on the Save button.



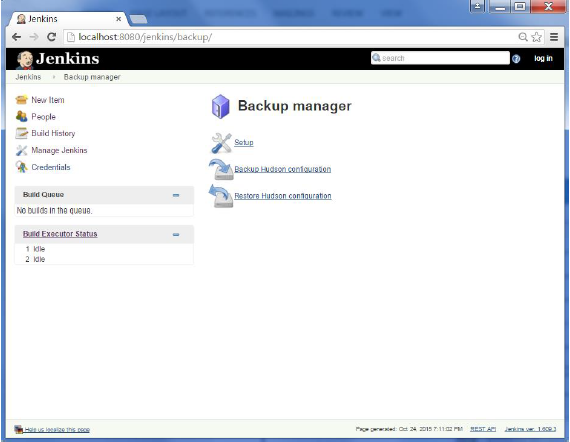
**Step 6** − Click on the ‘Backup Hudson configuration’ from the Backup manager screen to initiate the backup.



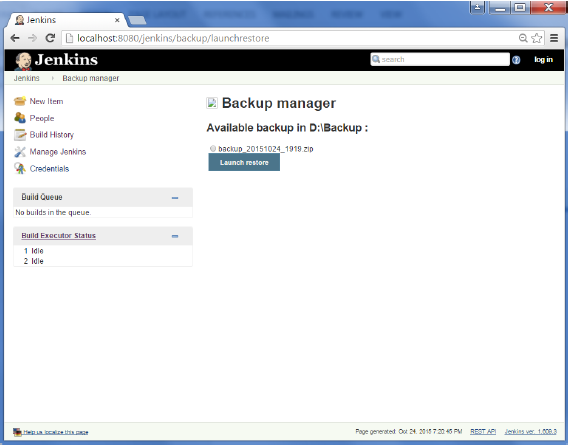
The next screen will show the status of the backup



To recover from a backup, go to the Backup Manager screen, click on Restore Hudson configuration.



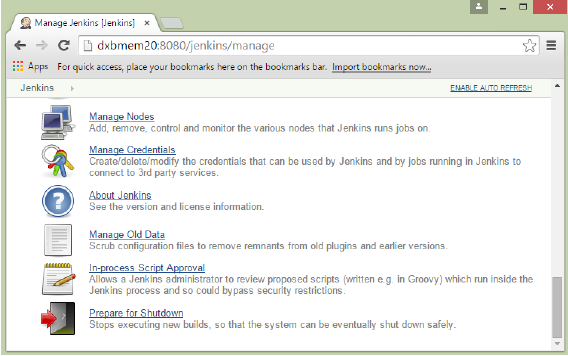
The list of backup’s will be shown, click on the appropriate one to click on Launch Restore to begin the restoration of the backup.



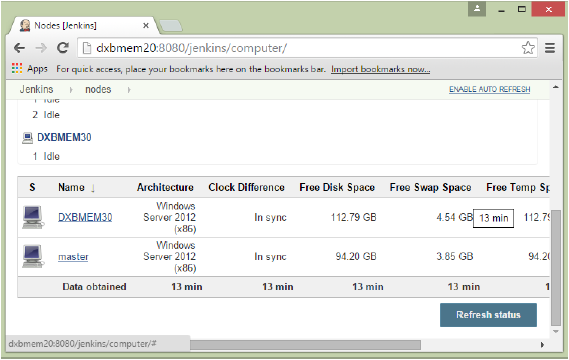
# Remote Testing

Web tests such as selenium tests can be run on remote slave machines via the master slave and selenium suite plugin installation. The following steps show how to run remote tests using this configuration.

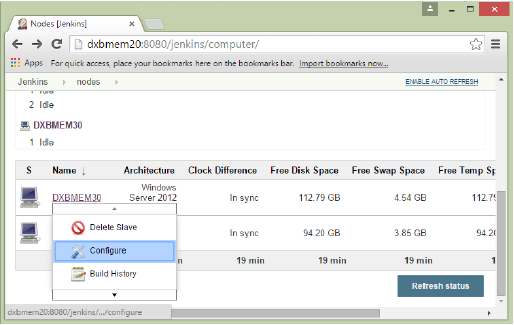
**Step 1** − Ensure your master slave configuration is in place. Go to your master Jenkins server. Go to Manage Jenkins → Manage Nodes.



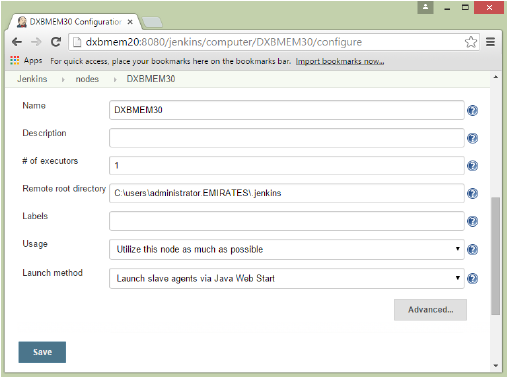
In our node list, the DXBMEM30 label is the slave machine. In this example, both the master and slave machines are windows machines.



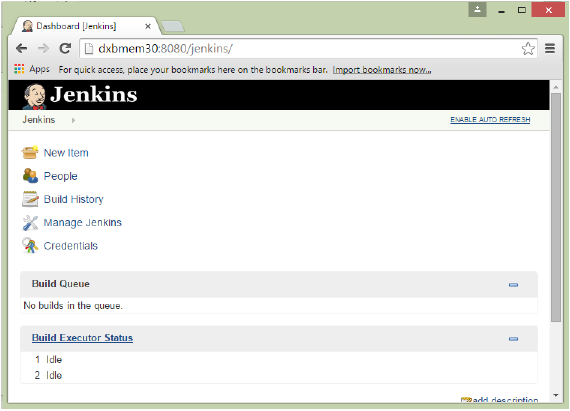
**Step 2** − Click on configure for the DXBMEM30 slave machine.



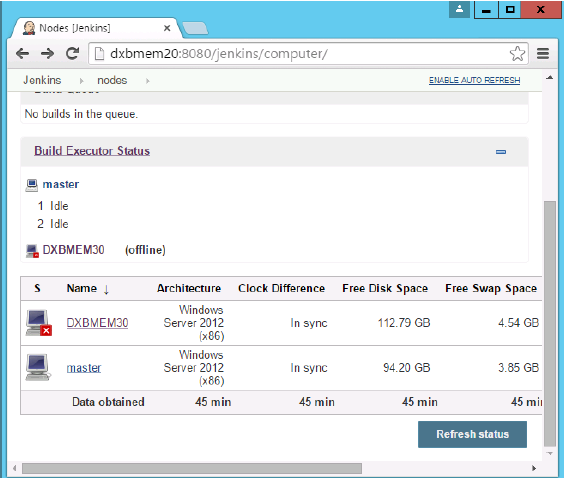
**Step 3** − Ensure the launch method is put as ‘Launch slave agents via Java Web Start’



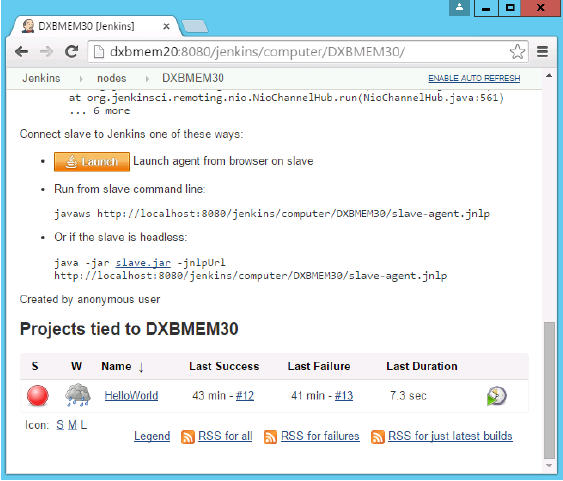
**Step 4** − Now go to your slave machine and from there, open a browser instance to your Jenkins master instance. Then go to Manage Jenkins → Manage Nodes. Go to DXBMEM30 and click on



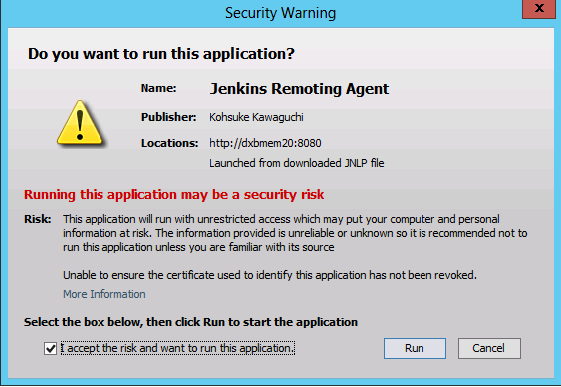
**Step 5** − Click on the DXBMEM30 instance.



**Step 6** − Scroll down and you will see the Launch option which is the option to Start ‘Java Web Start’



**Step 7** − You will be presented with a Security Warning. Click on the Acceptance checkbox and click on run.

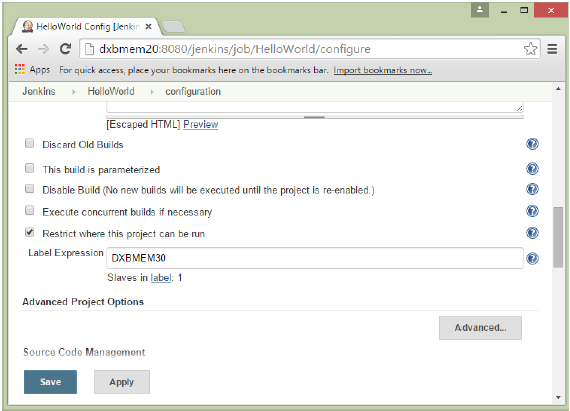


You will now see a Jenkins Slave window opened and now connected.

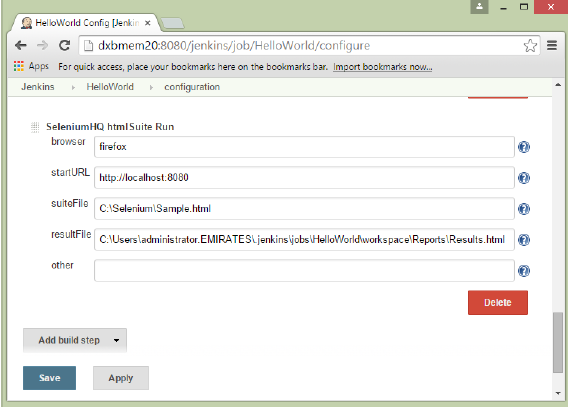


**Step 8** − Configuring your tests to run on the slave. Here, you have to ensure that the job being created is meant specifically to only run the selenium tests.

In the job configuration, ensure the option ‘Restrict where this project can be run’ is selected and in the Label expression put the name of the slave node.



**Step 9** − Ensure the selenium part of your job is configured. You have to ensure that the Sample.html file and the selenium-server.jar file is also present on the slave machine.



Once you have followed all of the above steps, and click on Build, this project will run the Selenium test on the slave machine as expected.