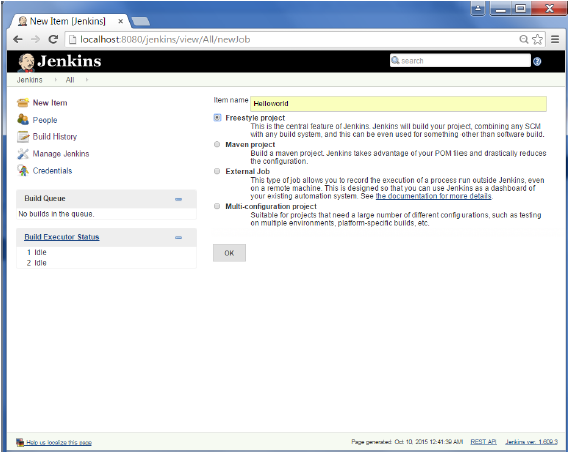
**Jenkins**

Create a job in Jenkins which picks up a simple HelloWorld application, builds and runs the java program.

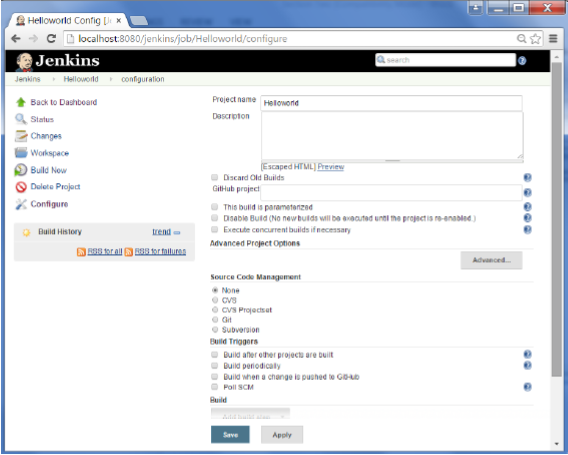
**Step 1** − Go to the Jenkins dashboard and Click on New Item



**Step 2** − In the next screen, enter the Item name, in this case we have named it Helloworld. Choose the ‘Freestyle project option’

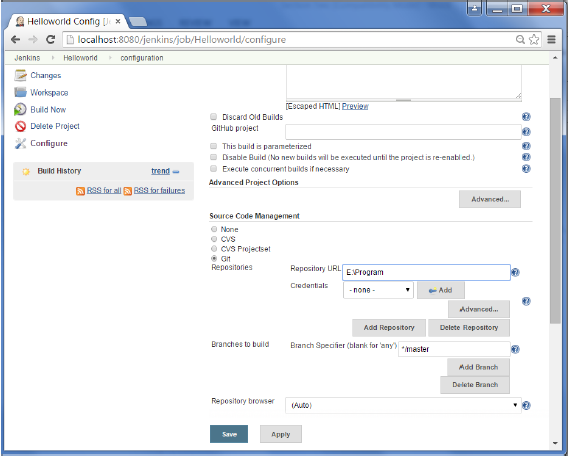


**Step 3** − The following screen will come up in which you can specify the details of the job.

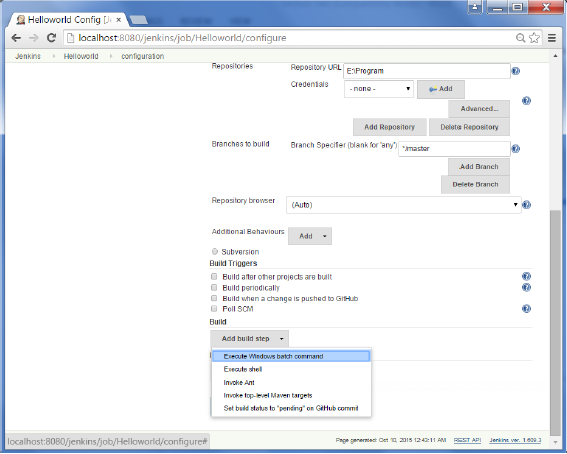


**Step 4** − We need to specify the location of files which need to be built. In this example, we will assume that a local git repository(E:\Program) has been setup which contains a ‘HelloWorld.java’ file. Hence scroll down and click on the Git option and enter the URL of the local git repository.

**Note** − If you repository if hosted on Github, you can also enter the url of that repository here. In addition to this, you would need to click on the Add button for the credentials to add a user name and password to the github repository so that the code can be picked up from the remote repository.

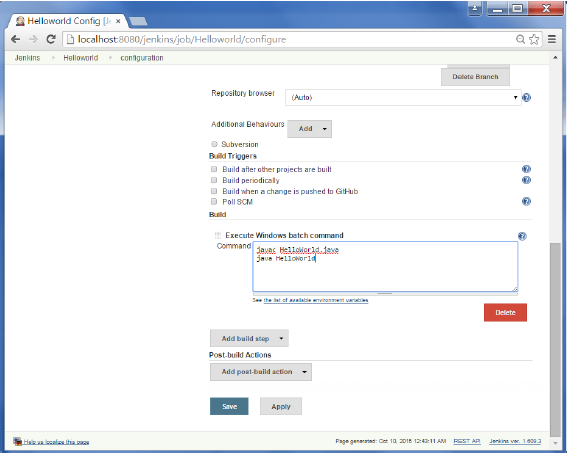


**Step 5** − Now go to the Build section and click on Add build step → Execute Windows batch command

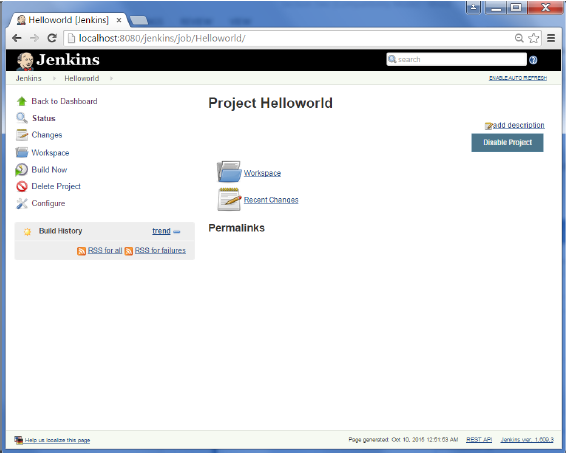


**Step 6** − In the command window, enter the following commands and then click on the Save button.

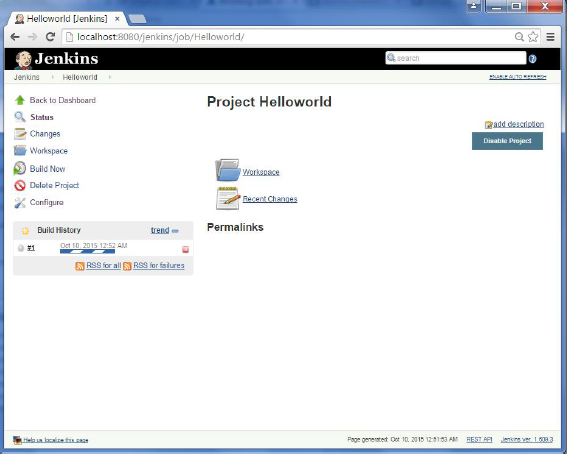
Javac HelloWorld.java  
Java HelloWorld



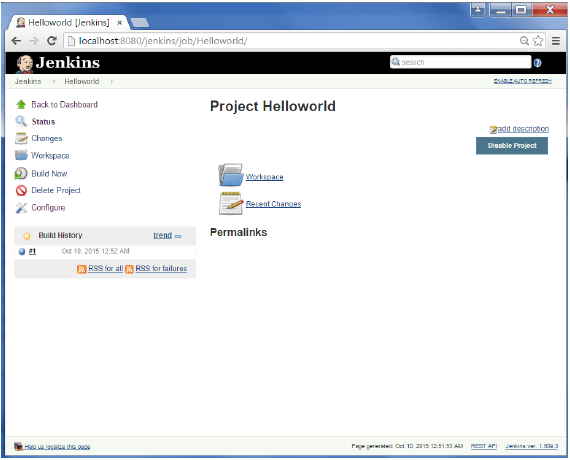
**Step 7** − Once saved, you can click on the Build Now option to see if you have successfully defined the job.



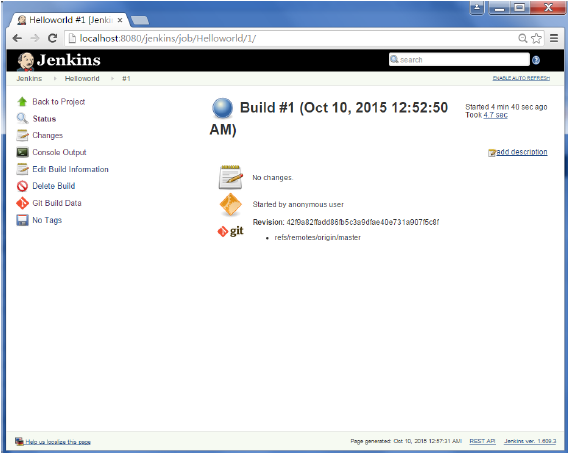
**Step 8** − Once the build is scheduled, it will run. The following Build history section shows that a build is in progress.



**Step 9** − Once the build is completed, a status of the build will show if the build was successful or not. In our case, the following build has been executed successfully. Click on the #1 in the Build history to bring up the details of the build.



**Step 10** − Click on the Console Output link to see the details of the build

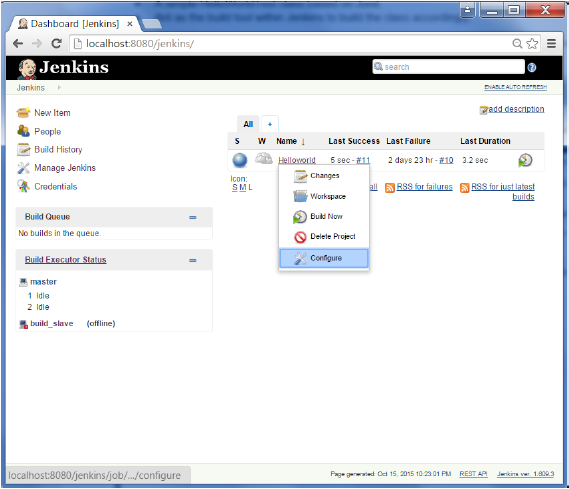
 

## **Example of a Junit Test in Jenkins**

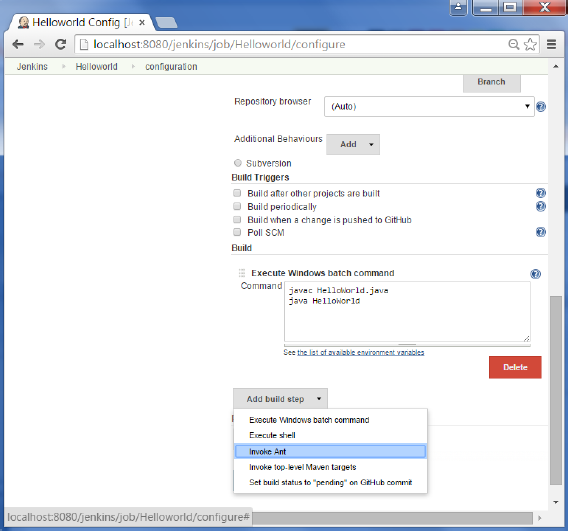
The following example will consider

* A simple HelloWorldTest class based on Junit.
* Ant as the build tool within Jenkins to build the class accordingly.

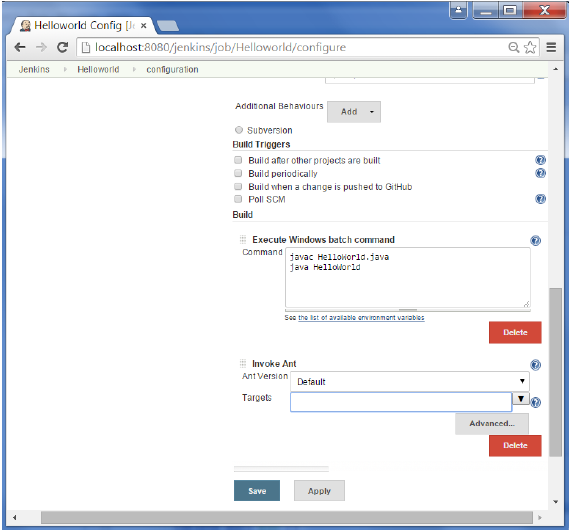
**Step 1** − Go to the Jenkins dashboard and Click on the existing HelloWorld project and choose the Configure option



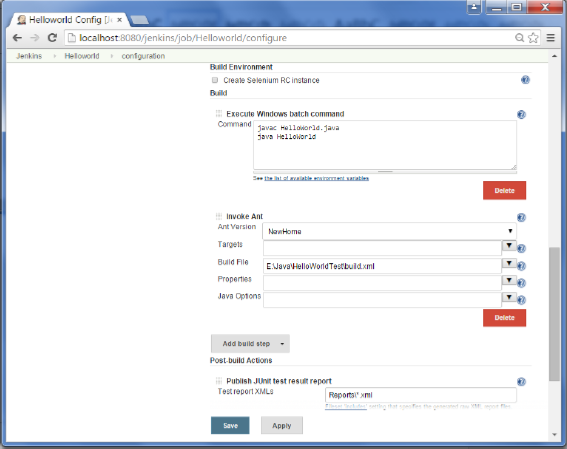
**Step 2** − Browse to the section to Add a Build step and choose the option to Invoke Ant.



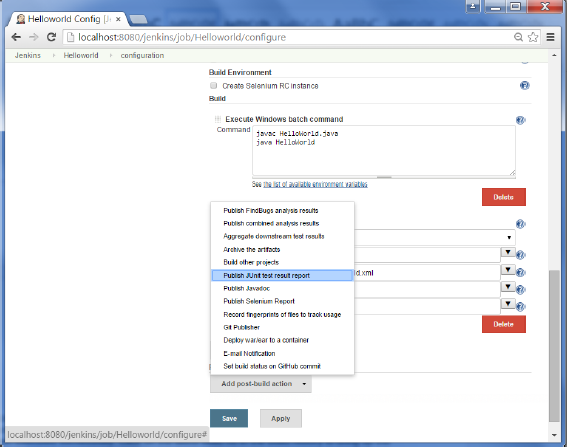
**Step 3** − Click on the Advanced button.



**Step 4** − In the build file section, enter the location of the build.xml file.

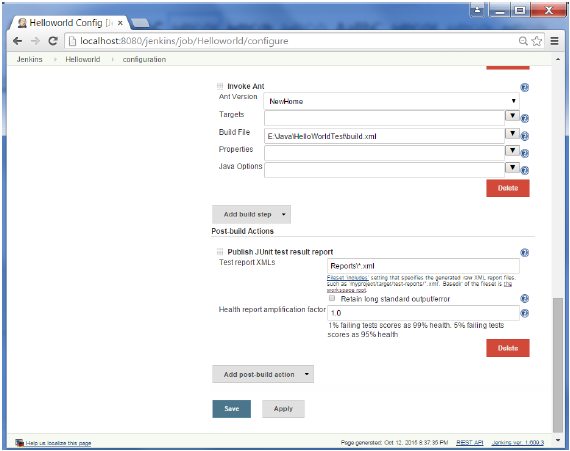


**Step 5** − Next click the option to Add post-build option and choose the option of “Publish Junit test result report”



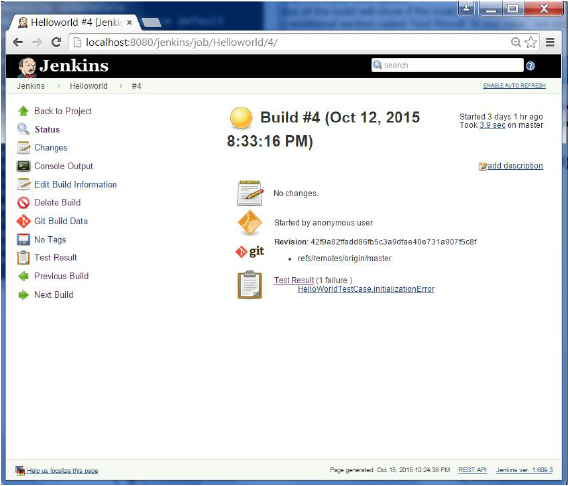
**Step 6** − In the Test reports XML’s, enter the location as shown below. Ensure that Reports is a folder which is created in the HelloWorld project workspace. The “\*.xml” basically tells Jenkins to pick up the result xml files which are produced by the running of the Junit test cases. These xml files which then be converted into reports which can be viewed later.

Once done, click the Save option at the end.



**Step 7** − Once saved, you can click on the Build Now option.

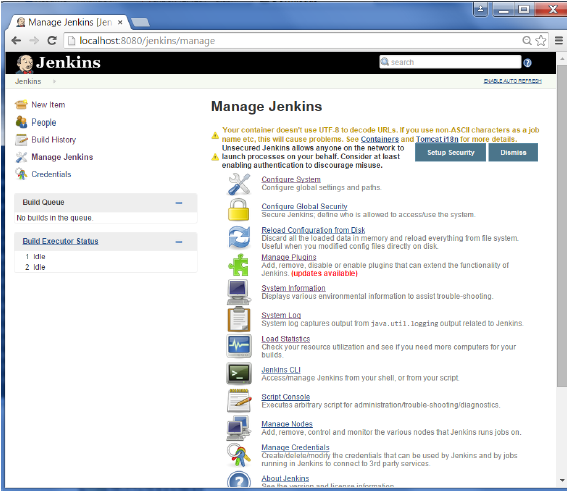
Once the build is completed, a status of the build will show if the build was successful or not. In the Build output information, you will now notice an additional section called Test Result. In our case, we entered a negative Test case so that the result would fail just as an example.



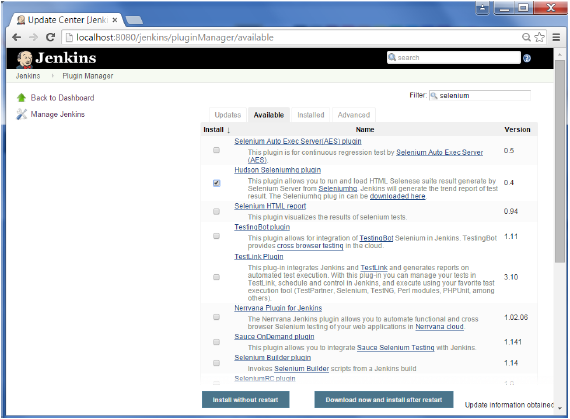
One can go to the Console output to see further information. But what’s more interesting is that if you click on Test Result, you will now see a drill down of the Test results.

how to use Selenium to run automated web tests.

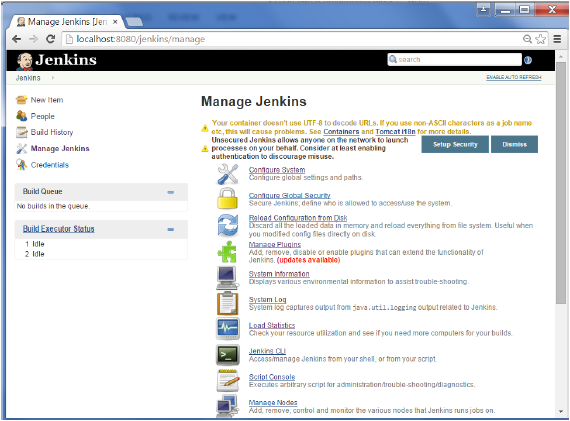
**Step 1** − Go to Manage Plugins.



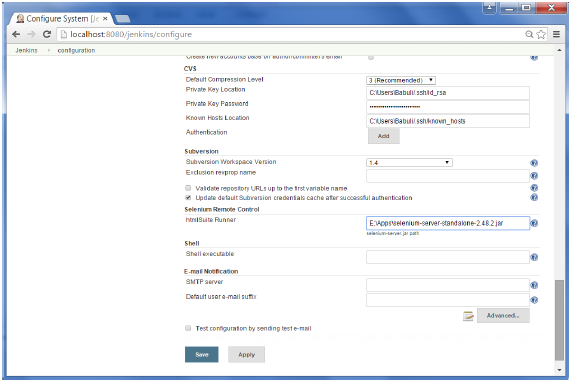
**Step 2** − Find the Hudson Selenium Plugin and choose to install. Restart the Jenkins instance.



**Step 3** − Go to Configure system.

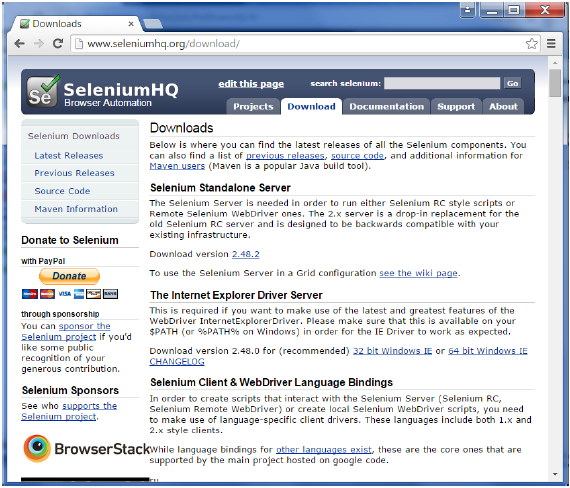


**Step 4** − Configure the selenium server jar and click on the Save button.

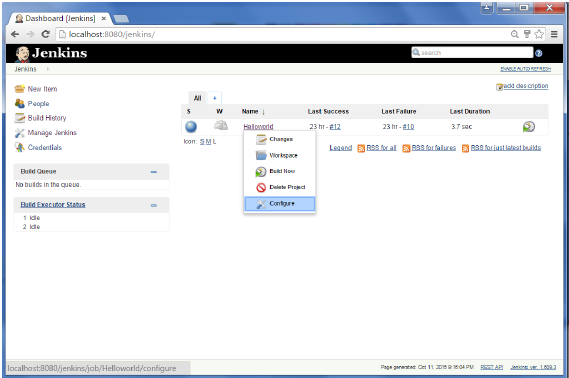


**Note** − The selenium jar file can be downloaded from the location [SeleniumHQ](http://www.seleniumhq.org/download/)

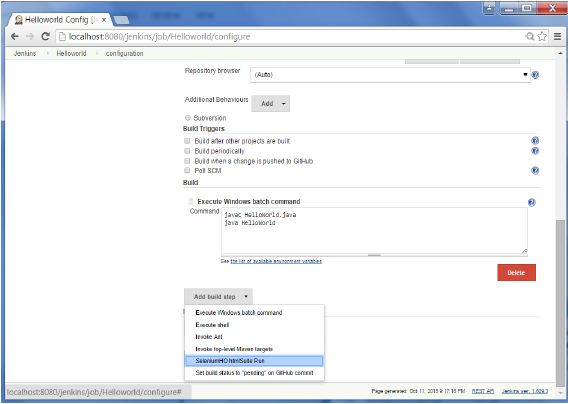
Click on the download for the Selenium standalone server.



**Step 5** − Go back to your dashboard and click on the Configure option for the HelloWorld project.



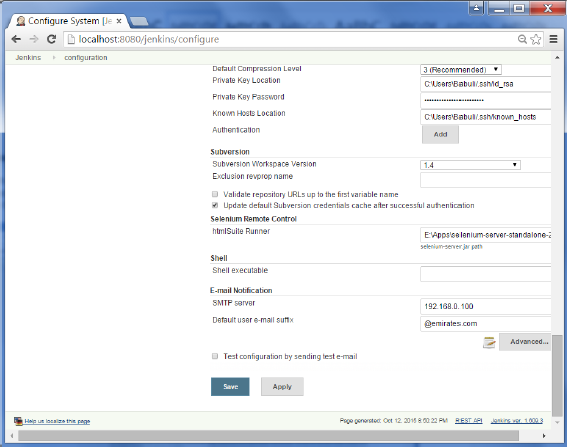
**Step 6** − Click on Add build step and choose the optin of “SeleniumHQ htmlSuite Run”



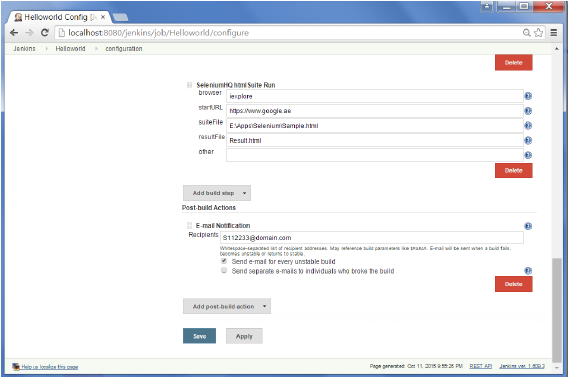
**Step 7** − Add the necessary details for the selenium test. Here the suiteFile is the TestSuite generated by using the Selenium IDE. Click on Save and execute a build. Now the post build will launch the selenium driver, and execute the html test.

Jenkins comes with an out of box facility to add an email notification for a build project.

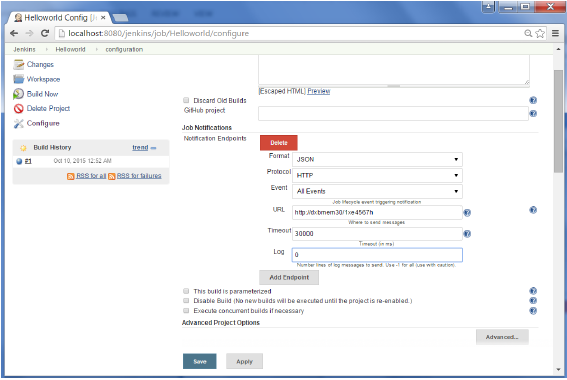
**Step 1** − Configuring an SMTP server. Goto Manage Jenkins → Configure System. Go to the E-mail notification section and enter the required SMTP server and user email-suffix details.



**Step 2** − Configure the recipients in the Jenkins project - When you configure any Jenkins build project, right at the end is the ability to add recipients who would get email notifications for unstable or broken builds. Then click on the Save button.



Apart from the default, there are also notification plugin’s available in the market. An example is the notification plugin from Tikal Knowledge which allows sending Job Status notifications in JSON and XML formats. This plugin enables end-points to be configured as shown below.



Here are the details of each option −

* **"Format"** − This is the notification payload format which can either be JSON or XML.
* **"Protocol"** − protocol to use for sending notification messages, HTTP, TCP or UDP.
* **"Event"** − The job events that trigger notifications: Job Started, Job Completed, Job Finalized or All Events (the default option).
* **"URL"** − URL to send notifications to. It takes the form of "[http://host](http://www.host.com/)" for HTTP protocol, and "host:port" for TCP and UDP protocols.
* **"Timeout"** − Timeout in milliseconds for sending notification request, 30 seconds by default.