**Name – Nabanita Sarkar**

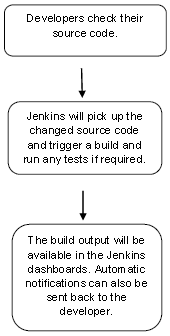
**Employee ID-815944**

**Jenkins Notes: -**

Jenkins is a powerful application that allows continuous integration and continuous delivery of projects, regardless of the platform you are working on. It is a free source that can handle any kind of build or continuous integration. You can integrate Jenkins with a number of testing and deployment technologies.

Why Jenkins?

Jenkins is a software that allows continuous integration. Jenkins will be installed on a server where the central build will take place. The following flowchart demonstrates a very simple workflow of how Jenkins works.



Along with Jenkins, sometimes, one might also see the association of Hudson. Hudson is a very popular open-source Java-based continuous integration tool developed by Sun Microsystems which was later acquired by Oracle. After the acquisition of Sun by Oracle, a fork was created from the Hudson source code, which brought about the introduction of Jenkins.

What is Continuous Integration?

Continuous Integration is a development practice that requires developers to integrate code into a shared repository at regular intervals. This concept was meant to remove the problem of finding later occurrence of issues in the build lifecycle. Continuous integration requires the developers to have frequent builds. The common practice is that whenever a code commit occurs, a build should be triggered.

Starting Jenkins

Open the command prompt. From the command prompt, browse to the directory where the jenkins.war file is present. Run the following command

D:\>Java –jar Jenkins.war

What is a Jenkins Pipeline?

Jenkins Pipeline (or simply "Pipeline") is a suite of plugins which supports implementing and integrating *continuous delivery pipelines* into Jenkins.

A *continuous delivery pipeline* is an automated expression of your process for getting software from version control right through to your users and customers.

Jenkins Pipeline provides an extensible set of tools for modeling simple-to-complex delivery pipelines "as code". The definition of a Jenkins Pipeline is typically written into a text file (called a Jenkinsfile) which in turn is checked into a project’s source control repository.

Jenkins Home Directory

Jenkins needs some disk space to perform builds and keep archives. One can check this location from the configuration screen of Jenkins. By default, this is set to ~/.jenkins, and this location will initially be stored within your user profile location. In a proper environment, you need to change this location to an adequate location to store all relevant builds and archives. Once can do this in the following ways

* Set "JENKINS\_HOME" environment variable to the new home directory before launching the servlet container.
* Set "JENKINS\_HOME" system property to the servlet container.
* Set JNDI environment entry "JENKINS\_HOME" to the new directory.

The following example will use the first option of setting the "JENKINS\_HOME" environment variable.

First create a new folder E:\Apps\Jenkins. Copy all the contents from the existing ~/.jenkins to this new directory.

Set the JENKINS\_HOME environment variable to point to the base directory location where Java is installed on your machine.

Jenkins URL

By default, the Jenkins URL points to localhost. If you have a domain name setup for your machine, set this to the domain name else overwrite localhost with IP of machine. This will help in setting up slaves and while sending out links using the email as you can directly access the Jenkins URL using the environment variable JENKINS\_URL which can be accessed as ${JENKINS\_URL}.

Email Notification

In the email Notification area, you can configure the SMTP settings for sending out emails. This is required for Jenkins to connect to the SMTP mail server and send out emails to the recipient list.

Jenkins Architecture

Jenkins follows Master-Slave architecture to manage distributed builds. In this architecture, slave and master communicate through TCP/IP protocol.

Jenkins architecture has two components:

* Jenkins Master/Server
* Jenkins Slave/Node/Build Server

Java Setup

Since Jenkins is written in Java. Therefore, Java must be installed on your system. To download and install the Java, go to our previous chapter of this tutorial.

Once the java has been installed properly on your system and Java environment variable has been set, then you can verify it by using the following commands:

1. C:\ echo %JAVA\_HOME%
3. C:\ javac -version
5. C:\ java -version

GitHub Setup for Jenkins

Jenkins is a CI (Continuous Integration) server and this means that it needs to check out source code from a source code repository and build code. Jenkins has outstanding support for various source code management systems like Subversion, CVS etc.

Github is the fast becoming one of the most popular source code management systems. It is a web based repository of code which plays a major role in DevOps. GitHub provides a common platform for many developers working on the same code or project to upload and retrieve updated code, thereby facilitating continuous integration. Jenkins works with Git through the Git plugin.

Connecting a GitHub private repository to a private instance of Jenkins can be tricky.

To do the GitHub setup, make sure that internet connectivity is present in the machine where Jenkins is installed.

* In the Home screen of the Jenkins (Jenkins Dashboard), click on the **Manage Jenkins** option on the left hand side of the screen.
* Now, click on the **Manage Plugins** option.
* In the next page, click on the "Available tab".
* The "Available" tab gives a list of plugins which are available for downloading. In the Filter tab type, type the "Git Plugin".
* Select the Git Plugin.
* Click on the "**install without restart**". The plugin will take some time to finish downloading depending on your internet connection, and will be installed automatically.
* You can also click on "**Download now and install after restart**" button in which the git plugin is installed after restart.
* If you already have the Git plugin installed then go to "Installed" tab and in filter option type Git plugin.

Jenkins - Setup Build Jobs

Let's create and run a job in Jenkins for simple **HelloWorld** in Java.

**Step 1:** Go to the Jenkins dashboard and click on the New Item.

**Step 2:** In the next page, enter the item name, and select the 'Freestyle project' option. And click OK. Here, my item name is HelloWorld.

**Step 3:** When you enter the OK, you will get a configuration page. Enter the details of the project in the Description section.

**Step 4:** On the Source Code Management section, select the **Git** option, and specify the Repository URL.

To do that you should have proper github setup on your system.

**Step 5:** Add the Repository URL in the **Source Code Management** section.

You can also use a local repository. And if your GitHub repository is private, Jenkins will first validate your login credentials with GitHub and only then access the source code from your GitHub repository.

**Step 6:** Now, it is time to build the code. Click on "**Add build step**" and select the "**Execute Windows batch command**".

**Step 7:** Enter the following command to compile the java code.

1. javac HelloWorld.java
2. java HelloWorld

**Step 8:** Click Apply and then Save button.

**Step 9:** Once you saved the configuration, then now can click on **Build Now** option.

**Step 10:** After clicking on **Build Now**, you can see the status of the build on the Build History section.

**Step 11:** Click on **Console Output** from the left side of the screen to see the status of the build you run. It should show the success message.