**CHAPTER 7**

**SETTING UP AUTOMATED BUILD/TEST SYSTEM (JENKINS) OVER CLOUD**

**CHAPTER 7**

**SETTING UP AUTOMATED BUILD/TEST SYSTEM (JENKINS) OVER CLOUD**

# **7.1 Introduction**

This Chapter specifies the setting up the infrastructure over cloud to achieve automated (dynamic and adaptive) Build system to achieve continuous code integration , doing automated build periodically on the latest code checked In repository over cloud, doing code coverage and static code analysis with tools(SonarQube)and plugins working with Build System and then continuous delivery and deployment of software. And Generates reports for inspections and corrections. This Build System supports any programming language and Static Code analysis supports continuous inspection of code quality and automatic review to detect bugs, code smalls, code complexity and security vulnerabilities works with 20+ Programming Languages

Jenkins is used for setting up adaptive build and test system.

Jenkins is a free and open-source continuous integration, Continuous Build, Continuous Test and Continuous Delivery software tool written in the Java programming language for testing and reporting on isolated changes in a larger code base dynamically in real time. Hence as per the reports generated engineers can find and solve defects/bugs/problems in a code base rapidly and to automate testing of their builds.

Jenkins have around 1,400 plugins to support the automation of all kinds of development and software testing tasks. i.e. building projects, running tests, doing static code analysis, and deploying are only few of many processes that engineers automate with Jenkins. Plugins span in five areas includes platforms, User Interface, administration, source code management, and, build management. It can be hosted in a cloud-based container or Virtual Machine. It follows the concept of Agent, which is typically a server machine, or container, which connects to a Jenkins master server and executes tasks as and when directed.

# **7.2 Prerequisite**

## **7.2.1 Hardware Requirements:**

1. A Physical/Virtual Machine
   1. Installed memory(RAM) 16.0 GB or 32.0 GB
   2. Either 32 or 64-bit Operating System.
      1. Now a day’s most of the server machines are coming with 64 bit configuration.
   3. Processor® CPU @2.80 GHz
      1. The more virtual CPU, more it will be beneficial to execute concurrent threads.
   4. Hard Disk : 200 GB
      1. The more hard disk space we will have the more it will be beneficial.
2. There may be 2 or more machine (virtual, physical) but more machines may be required as per project requirements.
   1. Setting up Slave machine is followed in coming sections.

## **Software Requirements:**

### **7.2.2.1 Java: As Jenkins itself a java application, hence proper supporting java version should be installed before installing Jenkins**.

* 1. Java 8 Latest Edition (I installed 8u\_151, 32 or 64 bit version or both together)
  2. Download java from <http://www.oracle.com/technetwork/java/javase/downloads/java-archive-javase8-2177648.html>

Select “Accept License Agreement” and Windows x86 and follow the installation instructions.

Set **JAVA\_HOME**:

Right click My Computer and select Properties.

1. On the Advanced tab, select Environment Variables, and then edit JAVA\_HOME to point to where the JDK software is located, for example, C:\Program Files (x86)\Java\%jdk%
   1. Here jdk = jdk1.8.0\_151.
2. Verify Java version at command prompt by running command **java –version.**
   1. **Command will run successfully it Java is installed properly and will show values as following.**



**Figure 7.1**: Command to Check Java Version

### **Get: This is essential as we need to get project content from GIT repository, windows system may come with preinstalled Get but it’s always better to install latest version of GIT.**

* 1. Download “GIT” from <https://gitforwindows.org/> , you will get latest version always. Or just go to <http://git-scm.com/download/win> and the download will start automatically. It will download latest 64 bit GIT version.
  2. Run Setup and follow the instructions, let default components be installed.
     1. Select Get Editor of your choice or keep the default one as shown in picture.



**Figure 7.2**: Choosing the default editor by Git

* 1. Either select Use Get from Get Bash only or “Use Get and optional Unix tools” optional UNIX tools may help in configuring having more command line option



**Figure 7.3**: Adjusting Git Path Environment

* 1. Select First Option “Use open ssl Library”

**Figure 7.4**: Choosing HTTPS transport backend

* 1. Click Next and configure “line ending conversions as mentioned below”



**Figure 7.5**: Configuring the line ending conversions

* 1. Click next and select default MinTTY terminal.
  2. Click Next and let default option be selected.
  3. Finally click install and setup will install selected components.
  4. Click Finish.
  5. Now Right Click My Computer -> Properties ->Advanced System Settings->EnvironmentVariables->New

**Figure 7.6**: Adding system variable Git

* 1. Set New system variable GIT =C:\Program Files\Git;C:\ProgramFiles\Get\cmd;C:\Program Files\Get\bin;
  2. Add %GIT% to Path variable.

### **Download and Install Notepad++ to edit configuration files in projects and Sonar.**

* + - * 1. <https://notepad-plus-plus.org/download/v7.5.5.html>
        2. Either 32 bit or 64 bit.
        3. Let install with default options.

### Download 7-Zip from <https://www.7-zip.org/download.html>

* + - * 1. Just click and install.
        2. This utility is useful to extract compressed files and decompress the contents to send them over mail.

### **Download and Install** Android

* + - * 1. Android Studio will install all necessary tools and Libraries required for any android project.
        2. Download latest android studio from <https://developer.android.com/studio/index.html>
        3. To install Android Studio on Windows, proceed as follows:

Launch the .exe file you downloaded. Follow the setup to install Android Studio and additional SDK tools (Install all tools).

You may see short video at <https://developer.android.com/studio/install.html>

* + - * 1. Setup “ANDROID\_HOME=C: \Users\dc-user\AppData\Local\Android\Sdk” environment variable, here dc-user is y username, in your case it may be different.

### Download and Install MS Visual Studio Latest Version.

* + - * 1. This need only for the VS projects and No need to install on every Machine but on a specific machine as this take much space.
        2. All components need to be installed on a specific slave node as it takes about 40 GB of space.
        3. Download community edition from <https://www.visualstudio.com/downloads/>
        4. Community edition may expire after a month hence if you have full Visual Studio Professional with Key then use it.
        5. Follow the instruction mentioned at :
        6. <https://docs.microsoft.com/en-us/visualstudio/install/install-visual-studio>
        7. Add “MSBUILD\_HOME=C: \Program Files (x86)\MicrosoftVisual Studio\2017\Community\MS Build\15.0” in environment variable and add %MSBUILD\_HOME” in path variable.

# **7.3 Setup Sonar**

**This includes setting up sonarqube and sonarqube scanner.**

## **Downloading sonarqube server and scanner**

* + - * 1. Download latest sonar version from <https://www.sonarqube.org/downloads/>
        2. You may either download latest or LTS (Long term supported) version.
        3. Once it is download then follow the instructions from <https://docs.sonarqube.org/display/SONAR/Get+Started+in+Two+Minutes>

1. Unzip the SonarQube distribution after its downloaded. ([Download Page](http://www.sonarsource.org/downloads/))

2. Start the SonarQube server:

|  |  |
| --- | --- |
|  | |
| <https://docs.sonarqube.org/display/SCAN/Analyzing+with+SonarQube+Scanner> |  |
|  | |
| Either select Windows 64 bit or Any and extract under “C:\Apps”.  I used Windows 64 bit ver. | |

## **Setting up environment variables**

Set SONARQUBE\_HOME=C:\Apps\sonarqube-7.0

Set SONAR\_RUNNER\_HOME=C:\Apps\sonar-scanner-3.0.3.778

Add %SONARQUBE\_HOME%\bin\windows-x86-32 to path variable.

After updating path as per step Sonar can be run just by typing StartSonat.bat

## **Running Sonar as a service (Install and Run from the locations)**

Run Sonar as a Windows Service  
SONARQUBE\_HOME%/bin/windows-x86-32/InstallNTService.bat

StartSonar service  
%SONARQUBE\_HOME%/bin/windows-x86-32/StartNTService.bat

Sonar service will run automatically and same need to configure with slave nodes as well.

## **Modify the port where Sonar is running, go to following**.

**C:\Apps\sonarqube-6.7\conf (you will see sonar.properties and wrapper.conf)**

1. Update the sonar-properties file for port as 9010  
sonar.web.host: 0.0.0.0  
sonar.web.port: 9010  
sonar.web.context: /

2. Change the wrapper.conf and add following line  
wrapper.java.additional.3=-Djava.io.tmpdir=../../temp/

3. Run Sonar as a Windows Service  
%SONARQUBE\_HOME%/bin/windows-x86-32/InstallNTService.bat

4. Start Sonar service  
%SONARQUBE\_HOME%/bin/windows-x86-32/StartNTService.bat

5. Change the Sonar runner properties file  
sonar.host.url=http://localhost:9010

Other Sonar Related Configuration with Jenkins will be given next.

# **7.4 Self-Signed Certificates and Windows Credential Manager**

## **7.4.1 Configure Self Signed SSL on Get for Windows**

The self-signed certificate can be configured on Windows using the following steps:

### **Using Internet Explorer.**

* 1. open the URL
  2. Once the page has loaded click on the padlock next to the address bar
  3. Select the root certificate
  4. Now click on the "View Certificate" then "Details" tab and then the "Copy to file" button
  5. Select the "**Base64 encoded**" option and then export to a **.cer** file on your local file system
  6. Now configure Get to use the downloaded .cer file using the Get config command

|  |
| --- |
| **get config --global http.sslCAInfo C:/Users/e069511/certificate.cer** |

### **Exporting a certificate using the Chrome browser**

|  |  |
| --- | --- |
|  | 1. Connect to the website using SSL ([https://whatever](https://whatever/)) 2. Since Chrome version 56, you do the following:   go to the Three Dots Menu -> More Tools -> Developer Tools,  Then click on the Security Tab.  See **Security Overview** with a **View certificate** button.   1. Click on the **View certificate** button.   A modal window will open. It has two panes.  The top one shows the trust hierarchy of the  sites certificate (the last one listed), the intermediate certificate(s),  And the root certificate (the topmost one).  The second pane, shows the details of the certificates.  There may be zero or more intermediate certificates.  The root certificate equipped with a gold-bordered icon. The others have a blue border. See the screen shot below.   1. To export a certificate:    1. First click on the certificate's icon in the trust hierarchy.    2. The certificate will be shown in the main part of the modal.    3. Click on the certificate's large icon in the main part of the modal.    4. Export the certificate to your desktop.   [enter image description here](https://i.stack.imgur.com/Hdmot.png) |

**Figure 7.7**: How to save a remote server ssl certificate

Use Unix2Dos for appending certificate information in existing certificate.

## **7.4.2 Configuring Credential Storage with GIT**

We are using Standard Windows Credential Manager with GIT to store credentials called as wincred.

Run Following command get config --global credential.helper wincred

And then do checkout/update from repository, a window will popup asking GIT credentials (username and password). Then it will store them permanently until changed with GIT itself.

Changes at Windows credential Manager can be made manually as well. Type “Credential Manager” inside Run



**Figure 7.8**: Windows Credential Manager

Click on “Credential Manager”, you may see options to Add, Edit and Delete under section “Generic Credentials”.

# **7.5 Setting up Jenkins**

## **7.5.1 Download and Installation**

a) Download latest Jenkins windows installer from <https://jenkins.io/download/>

Either LTS or Weekly version

**Windows**

To install from the website, using the installer:

* [Download the latest package](http://mirrors.jenkins.io/windows/latest) (Jenkins.msi)
* Open the package and follow the instructions
  1. Check installation instructions at <https://jenkins.io/doc/book/installing/>

After Downloading Jenkins zip file, decompress it and install as following

**WAR file**

The Web application Archive, file version can be installed on any operating system or platform which supports Java.

**To install the WAR file version of Jenkins:**

1. Download the [stable Jenkins WAR file](http://mirrors.jenkins.io/war-stable/latest/jenkins.war) on your machine.
2. Open up a terminal/command prompt window to the download directory.
3. Run the command java -jar jenkins.war.
4. Browse to http://localhost:8080 and wait until the **Unlock Jenkins** page appears.
5. Continue with the [Post-installation setup wizard](https://jenkins.io/doc/book/installing/#setup-wizard) mentioned next .

After Downloading and installing, we need to do post installation setup wizard.

<https://jenkins.io/doc/book/installing/#setup-wizard>

## **7.5.2 Post-installation setup wizard**

After successful installation and run of Jenkins using one of the procedures above, the post-installation setup wizard begins.

This setup wizard takes through few quick "one-off" steps to unlock Jenkins, customize it with plugins and create the first administrator user through which you can continue accessing Jenkins.

### **7.5.2.1 Unlocking Jenkins**

When new Jenkins instance starts, asked to unlock it using an automatically-generated password.

In windows it will be at following location: C:\Program Files (x86)\Jenkins\secrets

1. Browse to http://localhost:8080 (or whichever port you configured for Jenkins when installing it) and wait until the **Unlock Jenkins** page appears.



**Figure 7.9**: Unlocking Jenkins

1. From Jenkins console output, copy automatically-generated alphanumeric password (between the 2 sets of asterisks).



**Figure 7.10**: Jenkins Default Password

1. On the **Unlock Jenkins** page, paste this password into the **Administrator password** field and click **Continue**.  
   **Notes:**
   * If you ran Jenkins in Docker in detached mode, you can access the Jenkins console log from the Docker logs ([above](https://jenkins.io/doc/book/installing/#accessing-the-jenkins-console-log-through-docker-logs)).
   * The Jenkins console log indicates the location (in the Jenkins home directory) where this password can also be obtained. This password should be entered in the setup wizard on new Jenkins installations before you can access Jenkins’s main UI. This password also serves as the default administrator account’s password (with username "admin") if you happen to skip the subsequent user-creation step in the setup wizard.

### **7.5.2.2 Customizing Jenkins with plugins**

After [unlocking Jenkins](https://jenkins.io/doc/book/installing/#unlocking-jenkins), the **Customize Jenkins** page appears. Any number of useful plugins can be installed as part of initial setup.

Click one of the two options shown:

* **Install suggested plugins** - install the recommended set of plugins, which are based on common use cases.
* **Select any plugins to install** – may choose which set of plugins to initially install.

|  |  |
| --- | --- |
|  | Choose **Install suggested plugins if not sure of plugins initially.**.  You can install (or remove) additional Jenkins plugins at a  Later point in time via the [**Manage Jenkins**](https://jenkins.io/doc/book/managing) > [**Manage Plugins**](https://jenkins.io/doc/book/managing/plugins/) page in Jenkins. |
|  |  |

The wizard shows progression of Jenkins being configured and chosen set of Jenkins plugins being installed. This process may take a few minutes.

### **7.5.2.3 Creating the first administrator user**

Finally, after [customizing Jenkins with plugins](https://jenkins.io/doc/book/installing/#customizing-jenkins-with-plugins), Jenkins asks to create first administrator user.

1. When “**Create First Admin User”** page appears, in the respective fields then click **Save and Finish**.
2. When the **Jenkins is ready** page appears, click **Start using Jenkins**.  
   **Notes:**
   * This page may indicate **Jenkins is almost ready!** Instead and if so, click **Restart**.
   * If the page does not automatically refresh after a minute, use your web browser to refresh the page manually.
3. If necessary, log in to Jenkins with the credentials of the user, just created and ready to start using Jenkins!

|  |  |
| --- | --- |
|  | From this point on, the Jenkins UI is only  Accessible by providing valid username and password credentials. |

## **7.5.3 Initial Configuration:**

1. Jenkins installs as a service. Hence it always restart as soon as system reboots for any update. No need to start manually. Just need to access with IP: Port combination with login credentials.
2. Is somehow Jenkins is stopped then same may be restarted by typing following in browser
   1. Type in browser IP:port/restart,
   2. Or alternatively start services.msc (start->Run) and restart Jenkins as following.

**Figure 7.11**: Starting Jenkins Service

After creating first Administrator Account, any number of users can be added to Jenkins with credentials as following.

a) Go to Manage Jenkins -> Manage Users





**Figure 7.12**: Manage Jenkins/Users

b) Click on Manage Users and Then Create User



**Figure 7.13**: Jenkins Users Database

1. Fill the details and click on create user. As soon as user is created, then same credentials can be used to login with Jenkins.



**Figure 7.14:** Create Jenkins Users

## 7.5.4 Creating Sample Project

* + **Go to Jenkins home page and click on New Item**

**Figure 7.15**: Jenkins New Item

* + Select Freestyle Project and Enter Project Name

**Figure 7.16**: Jenkins Project Name

* + Click Ok and a Project Configuration Wizard will open.

## **7.5.4 Install and Configure Plug-Ins**

### From the web UI:

The simplest and most common way of installing plugins is through the

**Manage Jenkins** > **Manage Plugins**, available to administrators of a Jenkins environment.

Under the **Available** tab, plugins available can be downloaded.



**Figure 7.17**: Manage Plugins

Click on Manage Plug-In and search for the plugin you need to install. 

**Figure 7.18**: Search available plugins

Select the Plugin and choose either Install without restart or Download now and install after restart.

### Advanced Installation:

Click on Advanced Tab, You may upload any downloaded .hip (plugin file) and upload annually via below mentioned method.



**Figure 7.19**: Upload plugin

All plugins can be found and browse from <https://plugins.jenkins.io/>

Now Following Plugins need to be installed by above mentioned methods

### List of recommended plugins

1. Get Plugin <https://wiki.jenkins.io/display/JENKINS/Git+Plugin>
2. SonarQube plugin
3. SonarQube Scanner
4. Fire Line: is a static code analysis program to look for bugs in Java Code.
5. Sonar Quality Gates: If done setup, it fails the build whenever Quality Gate criteria in sonar analysis aren’t meet.
6. Junit Plugin: Allows Junit-format test results to be published
7. Grade Plugin, used to invoke a Grade build script as the main build step.

Configured under Manage Jenkins 🡪 Global Tool Configuration

Dependency: [struts](https://plugins.jenkins.io/structs) (version: 1.3)

1. Apache Maven comes by default with standard Jenkins installation
2. MS Build Plugin: allows MS Build to build .NET Projects

<https://wiki.jenkins.io/display/JENKINS/MSBuild+Plugin>

Dependency: Struts (1.3)

1. Notification: Default plugin comes with Jenkins but another plugin also available in market from Tikal Knowledge which allows sending Job Status notifications in JSON and XML formats
2. Distributed Build: A Slave computer can be configured to offload build projects from master computer. It’s an inbuilt feature and can be done with manage Jenkins, where it asks the details of node slave machine.
   1. Workspace Cleanup plugin: Deletes workspace after a build is finished
   2. Windows slave plugin : connects to windows machine and start slave agents on them

## **7.5.5 Configure Plugins: Global Tools Configuration:**

Check Jenkins 🡪 Manage Jenkins 🡪 Global Tool Configuration



**Figure 7.20:** Manage jenkins

1. Keep Maven Configuration as it is. “Default Maven Settings”



**Figure 7.21**: Global Tool Configuration

1. Configure JDK as shown in figure

**Figure 7.22**: Configure JDK

1. Configure GIT as following.

**Figure 7.23**: Configure GIT

1. Configure Grade as following.



**Figure 7.24**: Configure Gradle

1. Configure MS Build as following.

**Figure 7.25**: Configure MSBuild

* 1. Here installation path may also exist on Slave nodes, May not necessary all on master.

1. Configure SoanrQube Scanner for MS Build. Select latest version from drop down list.

**Figure 7.26**: Add SonarQube Scanner for MSBuild

1. Configure SonarQube Scanner as following. Select Latest version from drop down

**Figure 7.27**: Add SonarQube Scanner

1. Configure ANT as following. Select latest version from dropdown list.

**Figure 7.28**: Configure ANT

1. Configure MAVEN as following. Select latest version from dropdown list.

**Figure 7.29**: Configure Maven

## **7.5.6 Configure global settings and paths**.



Go to Manage Jenkins🡪 Configure System



**Figure 7.30**: Global Tool Configurations

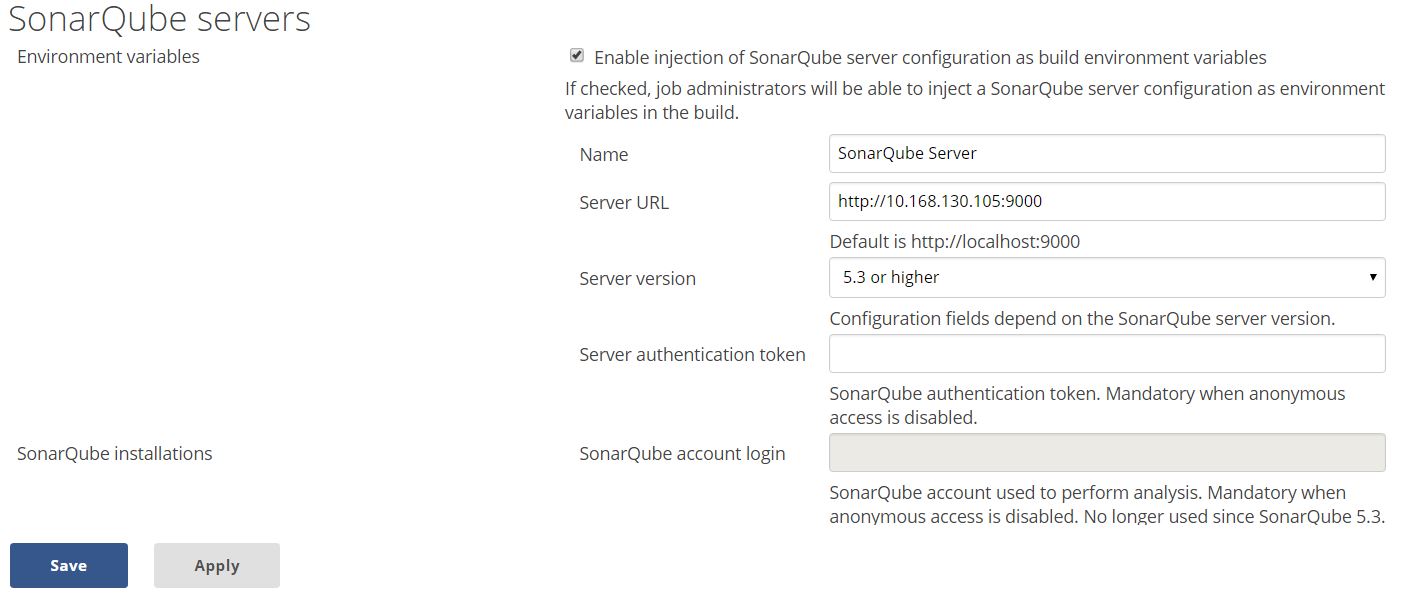
* 1. Configure Maven Project Configuration as following.

**Figure 7.31**: Maven Project Configuration

* + 1. I have selected default view as all, but any specific view can be selected.

**Figure 7.32**: Default View

* 1. Configure setting for SonarQube server as following. Mention full URL: port instead of localhost.

****

**Figure 7.33**: Configure SonarQube Servers

* 1. Configure Slave Status as following.

**Figure 7.34**: Slave Port Status

The status of slave machine can be shown as following.



Figure 7.35: Slave Machine Status

* 1. Jenkins Location.



**Figure 7.36**: Jenkins Location

* 1. Configure Quality Gates - Sonarqube

**Figure 7.37**: Configure Sonarqube Quality Gates

* 1. Configure Get plugin user.name and user.email values.
  2. Configure Extended Email configuration.
  3. Configure Email Notification.

**Figure 7.38**: Configure E-mail notification

We need to configure Java Mail as same is used by Jenkins and it doesn’t comes with default java installation. Always take latest version of Java Mail API.

Configuring Java Mail for Email notification

* + 1. Update and configure latest Java Mail API, add jar file in classpath. Refer following documents
       1. <http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-eeplat-419426.html#javamail-1.4.5-oth-JPR>
       2. <https://javaee.github.io/javamail/#overview_description>
    2. Extract Java Mail latest version under Installed Java. i.e. C:\Program Files (x86)\Java
       1. Add Java Mail “mail.jar” into environment variable JAVAMAIL\_JAR as following.

JAVAMAIL\_JAR= C:\Program Files (x86)\Java\javamail-1.4.5\mail.jar

* + - 1. Add CLASSPATH = %JAVAMAIL\_JAR% into environment variable.
    1. If Using Windows Platform, need to do settings in McAfee
       1. Right-click Access Protection and select Properties
       2. Click the Access Protection tab
       3. Under Categories on the left, select Anti-virus Standard Protection
       4. In the right pane, select Prevent mass mailing worms from sending mail, then click Edit
       5. In the Processes to exclude section, type the process name, then click OK to close the Rule details window
       6. Click Apply then close the Access Protection Properties window
    2. Add following (processes to exclude) chrome.exe, eclipse.exe, java.exe, csrss.exe, jenkins.exe
  1. Configure ANDROID\_HOME as following



**Figure 7.39**: Configure Android SDK

# **7.6 Setting up Project in Jenkins**

## **7.6.1 Configure Project**

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



Figure 7.40: Automation Build Setting

**Step 2** − from next screen, enter Item name, in this present case item named is HelloWorld. Choose the ‘Freestyle project option’

**Figure 7.41**: Configure Project Name

**Step 3** − the following screen can specify the details of the job.



**Figure 7.42**: Configure job notifications

Give Label expression Name of Nodes (slaves), you may also specify custom workspace as mentioned.

**Step 4** − we need to specify the location of files which need to be built. 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 repository so that the code can be picked up from the remote repository.



**Figure 7.43**: Configure GIT Repository URL

Step5: Configure Build Triggers

**Figure 7.44**: Configure Build Triggers

If Project have dependency on another project then select option “Build after other projects are built” as well.



**Figure 7.45**: Build Triggers

**Step 6** − Now go to the Build section and click on Add build step → Execute Windows batch command, You may Invoke Ant, Grade Script as per your project configuration.

**Figure 7.46**: Add Build Step

**Step 7** − in the command window, enter the following commands and then click on the Save button.

javac HelloWorld.java

Java HelloWorld



**Figure 7.47**: Configure Windows Batch Command

Alternatively->



**Figure 7.48:** Configure Root POM

**Step 8** - Configure Sonar Scanner



**Figure 7.49**: Configure Sonar Scanner Properties

Analysis properties-> can be written as following.

## 7.6.2 Sample sonar-project.properties as following

----------------------------------------------------------------------

#Configure here general information about the environment, such as SonarQube DB details for example

#No information about specific project should appear here

#Default SonarQube server

sonar.host.url=http://0.0.0.0

#Default source code encoding

sonar.sourceEncoding=UTF-8

# required metadata

sonar.projectKey= HelloWorld

#Java project analyzed with the SonarQube Runner

sonar.projectName=HelloWorld

sonar.projectVersion=1.0

#Language

sonar.language=java

#java version used by source files:

sonar.java.source=1.8

#Comma-separated paths to directories with sources (required)

sonar.sources=src/

#path to Java project compiled classes (optional)

sonar.java.binaries=bin/com

#sonar.java.binaries=target/classes

#comma-separated list of paths to libraries (optional

sonar.java.libraries=lib

#path to test source directories (optional)

#sonar.tests=testDir1, testDir2

**Step 9** – Configure Editable Email Notification

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

**Step 10**− 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 11** − Click on the Console Output link to see the details of the build

# **7.7 Setting up Dashboard**.

**Visit**

[**https://plugins.jenkins.io/build-monitor-plugin**](https://plugins.jenkins.io/build-monitor-plugin)

**Go to archive and download latest from**

[**https://updates.jenkins.io/download/plugins/build-monitor-plugin/**](https://updates.jenkins.io/download/plugins/build-monitor-plugin/)

**Upload downloaded plugin (.hip file) and restart the Jenkins.**

****

**Figure 7.50:** Uploading plugins

Now you may select the Build Monitor View to see the Status of Builds.

Give the Name Sample and Configure the Projects included.



**Figure 7.51**: Creating Sample Projects



**Figure 7.52**: Configure Build Monitor

# **7.8 Distributed Execution (Master and Slave Node)**

**Note: Prepare VM or Physical Machines have same kind of environment configuration and software installed. Bur additional software may be installed on slave nodes for additional task but there is no need to install Jenkins on slave node.**

## **7.8.1 Configure Master Machine**

* + 1. On Master Machine go to Manage Jenkins -> Manage Nodes



**Figure 7.53**: Manage Nodes

* + 1. Click on New Node

**Figure 7.54**: Configure New Node

* + 1. Enter Node Name and Select Permanent Agent then press OK.

**Figure 7.55**: Select Agents

* + 1. Fill the information as following.
       1. Set a **number of executors**
          1. (One or more) as needed.
       2. Set a **Remote FS Root**
          1. A home directory for the master on the slave machine.
          2. For a *Windows slave*, use something like: "C:\Jenkins\"
       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**
          1. *Keep this slave online as much as possible*
          2. TODO: add details for each option.
       6. Press **OK**.



**Figure 7.56**: Configure Build Slave Machine

## **7.8.2 Configure Slave Machine**

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.

Click on the **Launch** button to launch agent from browser on slave.



**Figure 7.57**: Launch Agent Jenkins Slave

Agent will be downloaded and then you may start the same



**Figure 7.58**: Jenkins agent launch

# **7.9 Advantages of Automated Build/Test System**

Continuous build is a necessary condition for continuous testing

It helps to improve product quality

Minimize the chances of build failures at the time of delivery

Eliminate dependency of manual build and Unit Testing Process

Keep track of History of Builds

Produces consistent builds with clean build for every check-in

Test Status and results can be presented on dash board.

Automated Regression and Functional Test Suite can be executed with/for every Build

Multiple execution of build process Load can be shared with Master and Slave configuration

# **References**

## Mastering Jenkins, 27 October 2015

By Jonathan McAllister

## [Continuous Delivery with Docker and Jenkins,](https://www.amazon.in/Continuous-Delivery-Docker-Jenkins-Leszko/dp/1787125238/ref=sr_1_3?ie=UTF8&qid=1539083868&sr=8-3&keywords=Jenkins" \o "Continuous Delivery with Docker and Jenkins) 24 August 2017

By [Rafal Leszko](https://www.amazon.in/Rafal-Leszko/e/B079VRD375/ref=sr_ntt_srch_lnk_3?qid=1539083868&sr=8-3)

## [Learning Continuous Integration with Jenkins: A beginner's guide to implementing Continuous Integration and Continuous Delivery using Jenkins 2, 2nd Edition,](https://www.amazon.in/Learning-Continuous-Integration-Jenkins-implementing/dp/1788479351/ref=sr_1_8?ie=UTF8&qid=1539083868&sr=8-8&keywords=Jenkins" \o "Learning Continuous Integration with Jenkins: A beginner's guide to implementing Continuous Integration and Continuous Delivery using Jenkins 2, 2nd Edition) 20 December 2017

## [Storage Management in Data Centers: Understanding, Exploiting, Tuning, and Troubleshooting Veritas Storage Foundation](https://www.amazon.in/Storage-Management-Data-Centers-Troubleshooting/dp/3540850228/ref=sr_1_17?ie=UTF8&qid=1539086234&sr=8-17&keywords=Data+Center" \o "Storage Management in Data Centers: Understanding, Exploiting, Tuning, and Troubleshooting Veritas Storage Foundation)

By Volker Herminghaus and Albrecht Scriba

## Jenkins 2: Up and Running- Evolve Your Deployment Pipeline for Next Generation Automation, 15 June 2018

By Brent Laster

## Jenkins: The Definitive Guide: Continuous Integration for the Masses

By John Ferguson Smart

## [Hybrid Cloud for Dummies,](https://www.amazon.in/Hybrid-Cloud-Dummies-Judith-Hurwitz/dp/8126542918/ref=sr_1_3?ie=UTF8&qid=1539086707&sr=8-3&keywords=Setting+up+cloud" \o "Hybrid Cloud for Dummies) 2013

By Judith Hurwitz and Marcia Kaufman

https://en.wikipedia.org/wiki/Jenkins\_(software)

https://en.wikipedia.org/wiki/Continuous\_delivery

https://en.wikipedia.org/wiki/Software\_build

https://en.wikipedia.org/wiki/Build\_automation

https://en.wikipedia.org/wiki/Acceptance\_test%E2%80%93driven\_development

https://en.wikipedia.org/wiki/Behavior-driven\_development

## [Hands-On Automation Testing with Java for Beginners: Build automation testing frameworks from scratch with Java,](https://www.amazon.in/Hands-Automation-Testing-Java-Beginners/dp/1789534607/ref=sr_1_5?ie=UTF8&qid=1539087490&sr=8-5&keywords=Build+Automation" \o "Hands-On Automation Testing with Java for Beginners: Build automation testing frameworks from scratch with Java) 29 September 2018

By Rahul Shetty

## [Android Continuous Integration: Build-Deploy-Test Automation for Android Mobile Apps,](https://www.amazon.in/Android-Continuous-Integration-Build-Deploy-Test-Automation-ebook/dp/B0755S4C7J/ref=sr_1_28?ie=UTF8&qid=1539087863&sr=8-28&keywords=Build+Automation" \o "Android Continuous Integration: Build-Deploy-Test Automation for Android Mobile Apps) 26 August 2017

By Pradeep Macharla

## [Build Automation: A Successful Design Process,](https://www.amazon.in/Build-Automation-Successful-Design-Process/dp/1979461937/ref=sr_1_62?ie=UTF8&qid=1539087935&sr=8-62&keywords=Build+Automation" \o "Build Automation: A Successful Design Process) 5 November 2017

By Gerard Blokdyk

https://jenkins.io/doc/

https://wiki.jenkins.io/display/JENKINS/Extend+Jenkins

https://en.wikipedia.org/wiki/Virtual\_machine

https://cloud.vmware.com/

https://buildbot.net/