

Jenkins

Jenkins Introduction

Jenkins, is an open source Continuous Integration, cross-platform tool written in Java. Kohsuke Kawaguchi is Creator of the Jenkins CI server in 2004. Initially, it was called Hudson, but in 2011 it was renamed to Jenkins because of disputes with Oracle.

The tool simplifies the process of integration of changes in to the project and delivery of fresh builds to users.

Continuous Integration: Continuous Integration (CI) is the process of automating the build and testing of code every time a team member commits changes to version control.

(OR)

Continuous Integration is a development practice where developers integrate code into a shared repository frequently, preferably several times a day. Each integration can then be verified by an automated build and automated tests. While automated testing is not strictly part of CI it is typically implied.

One of the key benefits of integrating regularly is that you can detect errors quickly and locate them more easily. As each change introduced is typically small, pinpointing the specific change that introduced a defect can be done quickly.

CI – What does it really mean?

At a regular frequency (ideally at every commit), the system is:

Integrated All changes up until that point are combined into the project

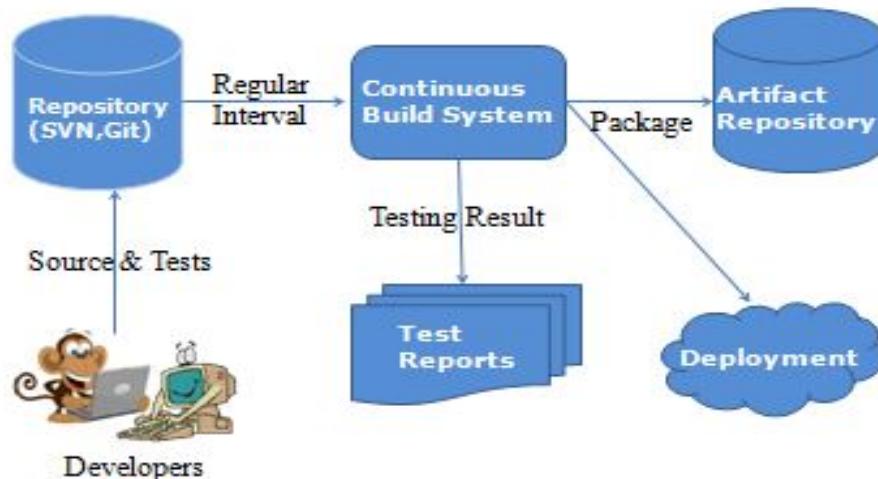
Built The code is compiled into an executable or package

Tested Automated test suites are run

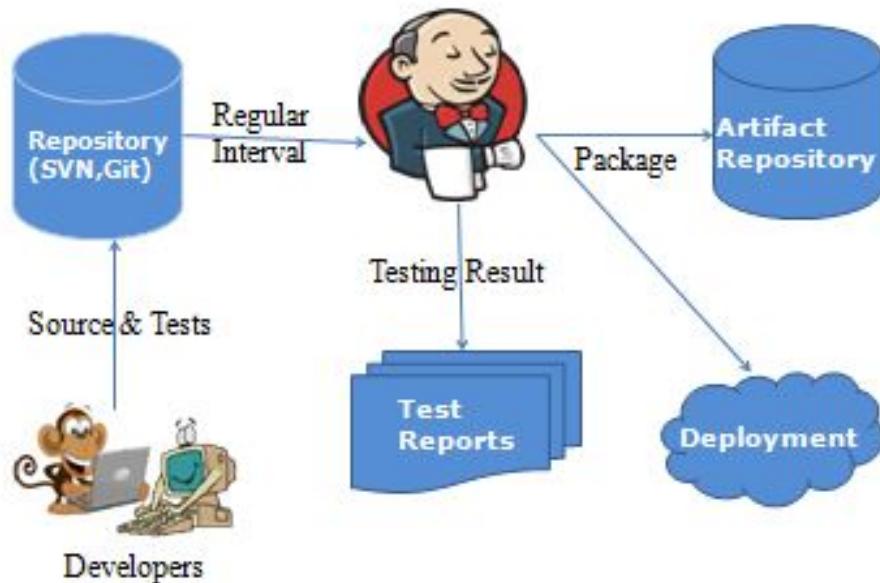
Archived Versioned and stored so it can be distributed as is, if desired

Deployed Loaded onto a system where the developers can interact with it

CI Flow



Below diagram CI flow with Jenkins as Build tool.



CI – Benefits

- Immediate bug detection
- No integration step in the lifecycle
- A deployable system at any given point
- Record of evolution of the project

CI – The tools

Code Repositories

GitHub, SVN, CVS, TFS and Mercurial,

Continuous Build Systems

Jenkins, Bamboo, Cruise Control, Team City, Circle CI and Travis CI

Test Frameworks

JUnit, Cucumber, CppUnit

Artifact Repositories

Nexus, Artifactory, Archiva

Build Tools

Ant, Maven and Gradle

Continuous Delivery: Any and every successful build that has passed all the relevant automated tests and quality gates can potentially be deployed in to production via fully automated one click process.

Continuous Deployment: The practicing of automatically deploying every successful build directly into production without any manual steps knows as Continuous deployment.

(OR)

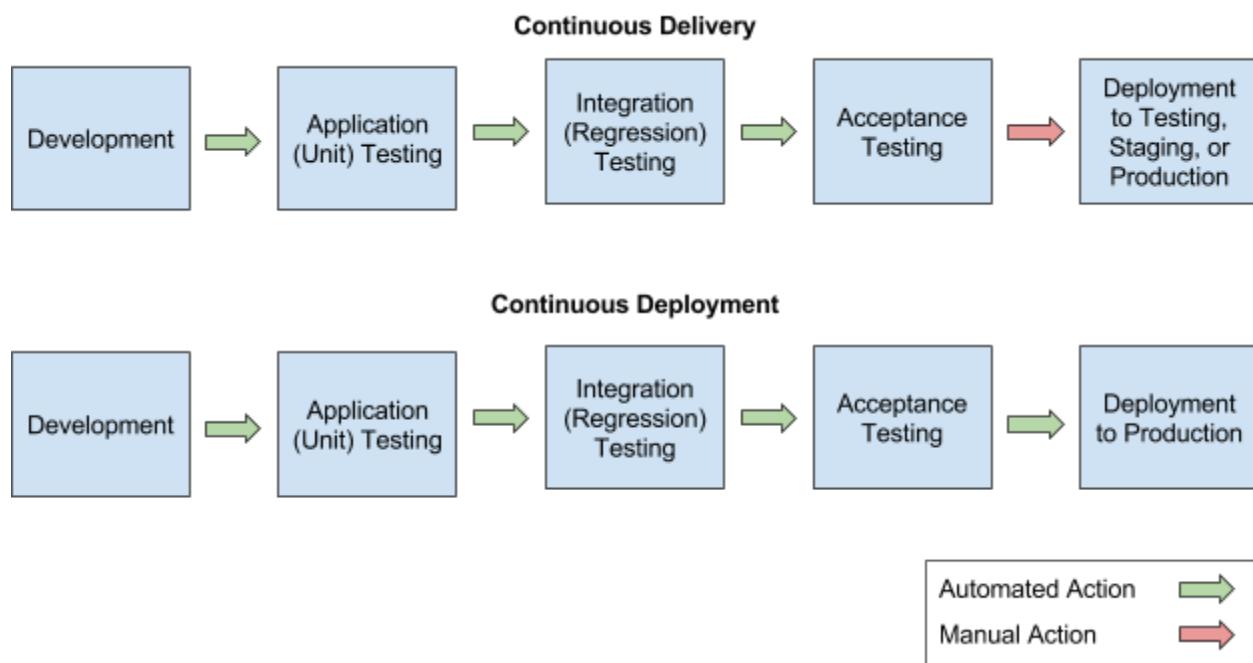
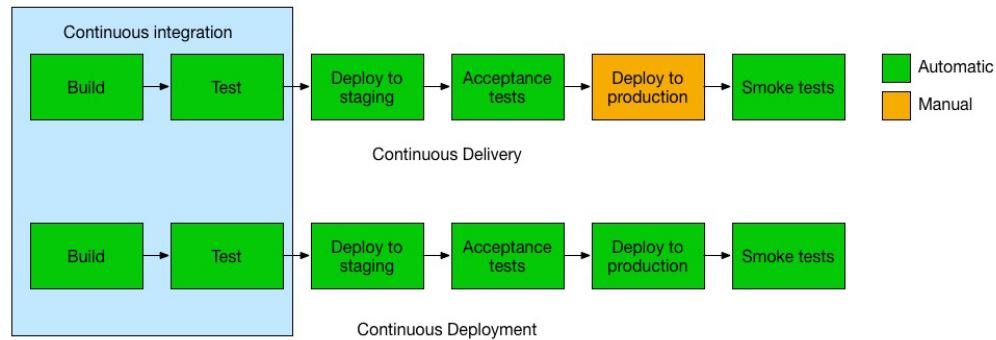
It is closely related to Continuous Integration and refers to keeping your application deployable at any point or even automatically releasing to a test or production environment if the latest version passes all automated tests.

CONTINUOUS DELIVERY



CONTINUOUS DEPLOYMENT





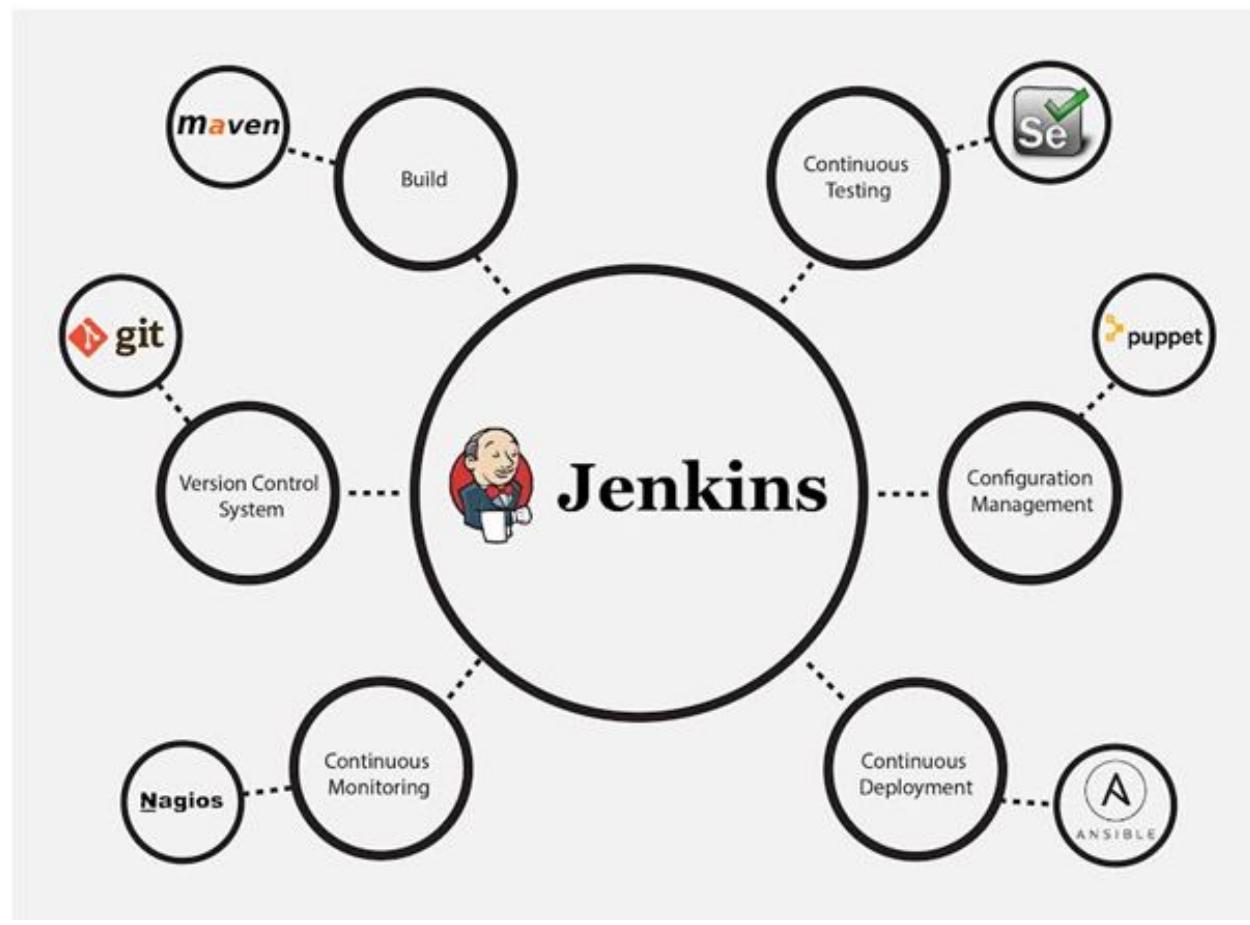
What Jenkins can do?

- Integrate with many different Version Control Systems (GitHub, CVS, SVN, TFS ...)
- Generate test reports (JUnit)
- Push the builds to various artifact repositories
- Deploy directly to production or test environments
- Notify stakeholders of build status (Through Email)

Benefits of Jenkins

- ✓ Its an open source tool with great community support.
- ✓ Easy to install and It has a simple configuration through a web-based GUI, which speeds up the Job
- ✓ It has around 900+ plugins to ease your work. If a plugin does not exist, just code it up and share with the community.
- ✓ Its built with Java and hence, it is portable on all major platforms.
- ✓ Good documentation and enriched support articles/information available on internet which will help beginners to start easy
- ✓ Specifically, for a test only project, it is used to schedule jobs for regression testing without manual intervention and hence monitor infrastructural and functional health of a application. It can be used like a scheduler for integration testing and also can be used to validate new deployments/environments on a single click on a Build now button.

The diagram below depicts that Jenkins is integrating various DevOps stages:



List of popular continuous Integration tools

<u>SNo</u>	<u>Product</u>	<u>Is Open Source?</u>
1	Jenkins	Yes
2	Bamboo	No
3	Cruise Control	Yes
4	Travis CI	Yes and Paid also
5	Circle CI	Yes and Paid also
6	GitLab CI	Yes and Paid
7	TeamCity	Yes and Paid

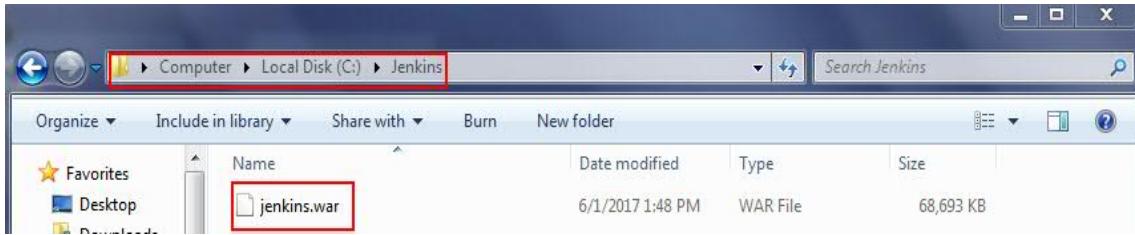
Jenkins Installation

- Jenkins is java based CI tool, so we need to install jdk/jre before installing.
- **Pre Requisite Software:** Java (Check weather java is installed or not with java –version command)
- We can Install Jenkins in 3 ways.

Method 1:

Step 1: Download jenkins.war file from <https://jenkins.io/download/>

Step 2: Now copy 'jenkins.war' in a folder. I have created one folder like Jenkins in C drive and placed there.



Step 3: Now go to the command prompt and execute the below statement via command line.

```
java -jar jenkins.war
```

```
C:\Jenkins>dir
Volume in drive C has no label.
Volume Serial Number is C6A9-A8C7

Directory of C:\Jenkins

07/01/2017  10:20 PM    <DIR>
07/01/2017  10:20 PM    <DIR>
06/01/2017  01:48 PM    70,340,821 jenkins.war
                  1 File(s)   70,340,821 bytes
                  2 Dir(s)  253,322,514,432 bytes free

C:\Jenkins>java -jar jenkins.war
```

The command prompt shows the directory listing of the Jenkins folder on drive C. It then executes the command 'java -jar jenkins.war'. The output shows the Java command being run.

Note: java -jar jenkins.war ---> With Default Port No 8080
java -jar jenkins.war --httpPort=8081 ---> With customised Port No
java -jar jenkins.war --help ---> Displays the all possible options.
nohup java -jar jenkins.war > \$LOGFILE 2>&1 ---> Even if terminal close also job will run in background.

Step 4: After executing the above command, you should see something like below.

```
C:\Windows\system32\cmd.exe - java -jar jenkins.war
C:\jenkins>java -jar jenkins.war
Running From: C:\jenkins\jenkins.war
Jul 02 2017 7:25:56 AM Main deleteWinstoneTempContents
WARNING: Failed to delete the temporary Winstone file C:\Users\IBM_ADMIN\AppData\Local\Temp\winstone\jenkins.war
+10mJul 02 2017 7:25:56 AM org.eclipse.jetty.util.log.JavaUtilLog.info
INFO: Logging to org.eclipse.jetty.util.log:java.util.logging.SimpleFormatter
Jul 02 2017 7:25:56 AM winstone.Logger logInternal
INFO: Beginning extraction from war file
+13mJul 02 2017 7:26:12 AM org.eclipse.jetty.util.log.JavaUtilLog.warn
INFO: Jetty-9.2.z-SNAPSHOT
Jul 02 2017 7:26:20 AM org.eclipse.jetty.util.log.JavaUtilLog.info
INFO: No ContextHandler registered
INFO: Jenkins home directory: C:\Users\IBM_ADMIN\jenkins\found at: $user.home/.jenkins
Jul 02 2017 7:26:20 AM org.eclipse.jetty.util.log.JavaUtilLog.info
INFO: Started webapp at /C:/Users/IBM_ADMIN/.jenkins/war/.AVAILABLE<>(C:/Users/IBM_ADMIN/.jenkins/war)
INFO: Started ServerConnector@62435e:8080@HTTP/1.1(0.0.0.0:8080)
Jul 02 2017 7:26:29 AM org.eclipse.jetty.util.log.JavaUtilLog.info
INFO: Started 63423ms
Jul 02 2017 7:26:30 AM winstone.Logger logInternal
INFO: Winstone Servlet Engine v2.0 running; controlPort=disabled
Jul 02 2017 7:26:34 AM Jenkins.InitReactorRunner$1 onAttained
INFO: Started initialization
Jul 02 2017 7:26:34 AM Jenkins.InitReactorRunner$1 onAttained
INFO: Listed all plugins
Jul 02 2017 7:26:44 AM Jenkins.InitReactorRunner$1 onAttained
INFO: Preparing all plugins
Jul 02 2017 7:27:02 AM Jenkins.InitReactorRunner$1 onAttained
INFO: Started all plugins
Jul 02 2017 7:27:04 AM Jenkins.InitReactorRunner$1 onAttained
INFO: Started all extensions
Jul 02 2017 7:27:04 AM Jenkins.InitReactorRunner$1 onAttained
INFO: Loaded all jobs
Jul 02 2017 7:27:02 AM hudson.model.AsyncPeriodicWork$1 run
INFO: Starting periodic metadata
Jul 02 2017 7:27:02 AM Jenkins.InitReactorRunner$1 onAttained
INFO: Completed initialization
Jul 02 2017 7:27:05 AM org.springframework.context.support.AbstractApplicationContext prepareRefresh
INFO: Refreshing org.springframework.context.support.GenericXmlApplicationContext@230b79580: defining beans [Root WebApplicationContext]; startup date [Sun Jul 02 07:27:05 IST 2017]; root of factory hierarchy
Jul 02 2017 7:27:05 AM org.springframework.beans.factory.support.DefaultListableBeanFactory preInstantiateSingletons
INFO: Pre-instantiating singletons in org.springframework.context.support.GenericXmlApplicationContext@230b79580: defining beans [Root WebApplicationContext]; root of factory hierarchy
Jul 02 2017 7:27:05 AM org.springframework.beans.factory.support.StaticWebApplicationContext@4665f7c: defining beans [AuthenticationManager]; root of factory hierarchy
Jul 02 2017 7:27:06 AM org.springframework.context.support.StaticWebApplicationContext@4665f7c: display name [Root WebApplicationContext]; startup date [Sun Jul 02 07:27:06 IST 2017]; root of factory hierarchy
INFO: Refreshing org.springframework.context.support.AbstractApplicationContext@5c8a972b: display name [Root WebApplicationContext]; startup date [Sun Jul 02 07:27:06 IST 2017]; root of factory hierarchy
INFO: Bean factory for application context org.springframework.context.support.GenericXmlApplicationContext@5c8a972b: defining beans [filter.legacy]; root of factory hierarchy
Jul 02 2017 7:27:06 AM org.springframework.beans.factory.support.DefaultListableBeanFactory preInstantiateSingletons
INFO: Pre-instantiating singletons in org.springframework.beans.factory.support.DefaultListableBeanFactory@4a478df: defining beans [filter.legacy]; root of factory hierarchy
Jul 02 2017 7:27:07 AM Jenkins.install.SetupWizard init
INFO:
*****
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:
71525b24261a4bed866afc8e481a40cb
This may also be found at: C:\Users\IBM_ADMIN\.jenkins\secrets\initialAdminPassword
*****
*****
*****
Jul 02 2017 7:27:20 AM hudson.model.UpdateSite updateData
INFO: Obtained the latest update center data file for UpdateSource default
Jul 02 2017 7:27:21 AM hudson.model.UpdateSite updateData
INFO: Obtained the latest update center data file for UpdateSource default
Jul 02 2017 7:27:22 AM hudson.WebAppMain$3 run
INFO: Jenkins is fully up and running
Jul 02 2017 7:27:23 AM hudson.model.DownloadService$Downloadable load
INFO: Obtained the updated data file for hudson.tasks.MavenInstaller
Jul 02 2017 7:27:26 AM hudson.model.DownloadService$Downloadable load
INFO: Obtained the updated data file for hudson.tools.JDKInstaller
Jul 02 2017 7:27:26 AM hudson.model.AsyncPeriodicWork$1 run
INFO: Finished Download metadata: 24,929 ns
```

Step 5: Once Jenkins has started, you should now be able to access it by using URL -
<http://localhost:8080>

Note: Jenkins will run on 8080 port number by default.

Jenkins 1.534 and earlier ships with Winstone, so no additional installation needed. Just run with java -jar jenkins.war.

Jenkins comes bundled as a WAR file that you can run directly using an embedded servlet container. Jenkins uses the lightweight Winstone servlet engine to allow you to run the server out of the box, without having to configure a web server yourself.

Getting Started

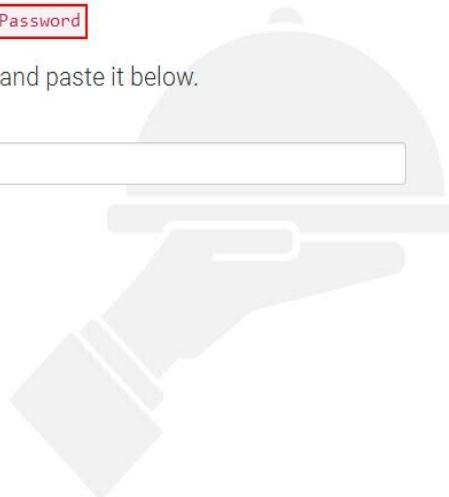
Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

C:\Users\IBM_ADMIN\.jenkins\secrets\initialAdminPassword

Please copy the password from either location and paste it below.

Administrator password



Continue

Getting Started



Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Jenkins 2.46.3

Getting Started

Getting Started

✓ Folders Plugin	✓ OWASP Markup Formatter Plugin	⌚ build timeout plugin	⌚ Credentials Binding Plugin	** bouncycastle API Plugin Folders Plugin ** Structs Plugin ** JUnit Plugin OWASP Markup Formatter Plugin PAM Authentication plugin ** Windows Slaves Plugin ** Display URL API Jenkins Mailer Plugin LDAP Plugin
⌚ Timestamper	⌚ Workspace Cleanup Plugin	⌚ Ant Plugin	⌚ Gradle Plugin	
⌚ Pipeline	⌚ GitHub Organization Folder Plugin	⌚ Pipeline: Stage View Plugin	⌚ Git plugin	
⌚ Subversion Plug-in	⌚ SSH Slaves plugin	⌚ Matrix Authorization Strategy Plugin	✓ PAM Authentication plugin	
✓ LDAP Plugin	⌚ Email Extension Plugin	✓ Mailer Plugin		** - required dependency

Jenkins 2.50

Getting Started

Create First Admin User

Username:	devops
Password:	*****
Confirm password:	*****
Full name:	DevOps
E-mail address:	opstrainingblr@gmail.com

Jenkins 2.46.3

Continue as admin

Save and Finish

Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

[Start using Jenkins](#)

Jenkins 2.46.3

Dashboard [Jenkins] x +

localhost:8080 IBM Most Visited Mail

Jenkins Jenkins >

New Item People Build History Manage Jenkins My Views Credentials

Welcome to Jenkins!

Please [create new jobs](#) to get started.

Build Queue

No builds in the queue.

Build Executor Status

1 Idle
2 Idle

Page generated: Jul 2, 2017 7:59:31 AM IST REST API Jenkins ver. 2.46.3

Method 2:

You can simply deploy the Jenkins WAR in your application server - with Tomcat, for example, you can simply place the jenkins.war file in the webapps Tomcat directory.

Start the tomcat server as follows.

>cd TOMCAT_HOME/bin directory

Windows

>startup.bat (OR) > catalina.bat start

MAC/Linux:

#startup.sh (OR) > catalina.sh start

Tomcat will run 8080 port on by default.

Note: netstat -a will give the port numbers.

If you are running Jenkins on an application server, the URL to use to access Jenkins will be slightly different. For example, on a Tomcat installation by default, you can access Jenkins in your web browser at <http://localhost:8080/jenkins>.

Method 3: Windows

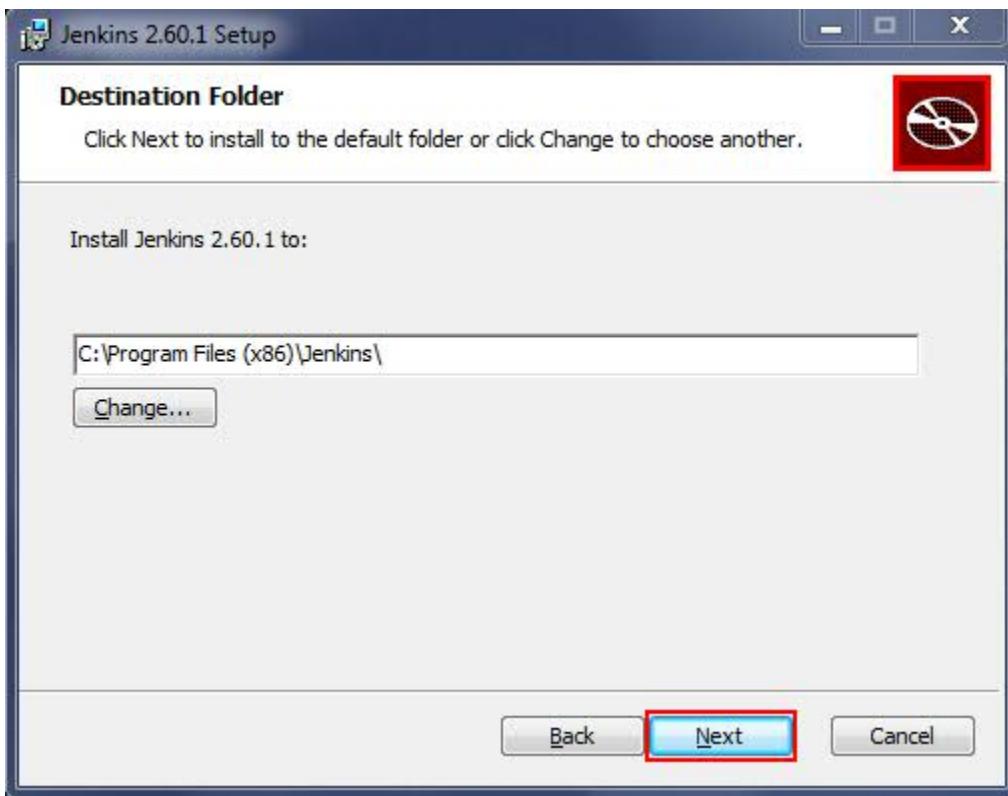
Click on below icon.



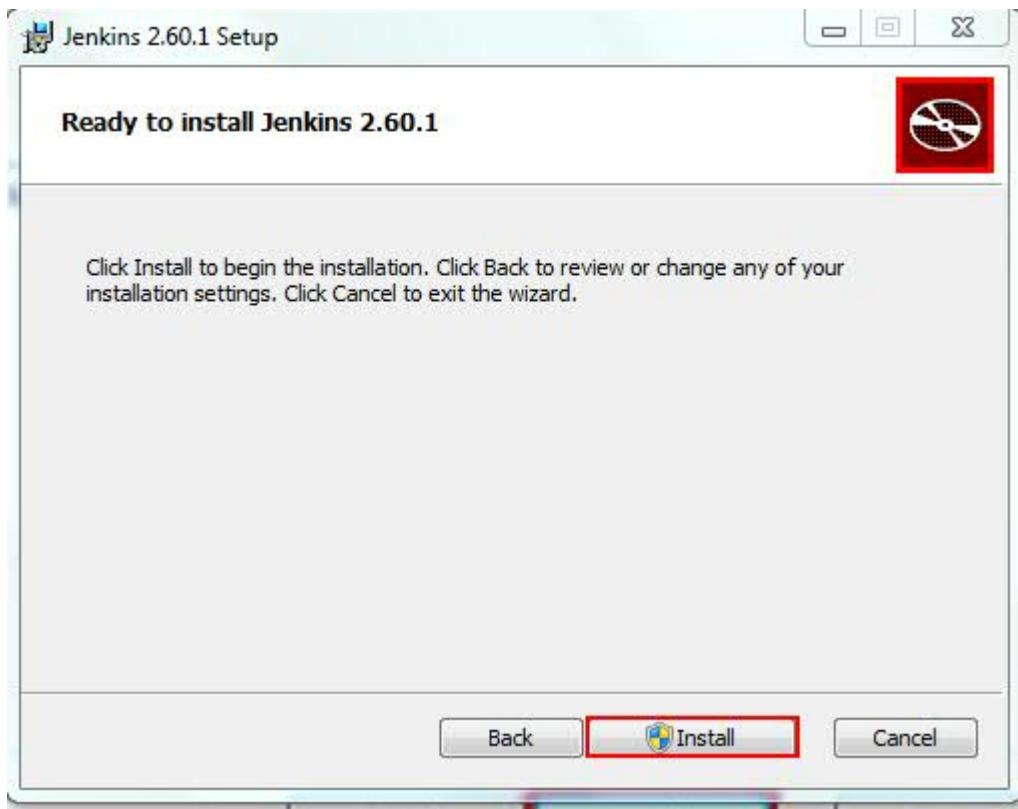
You will get the below screen



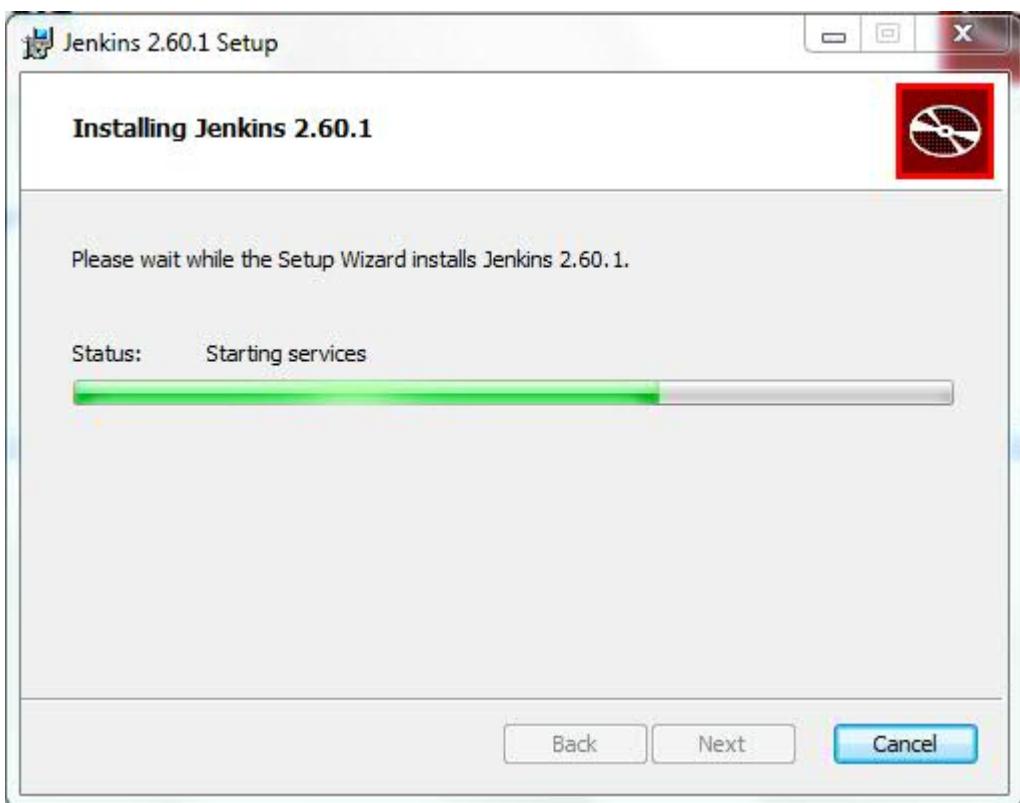
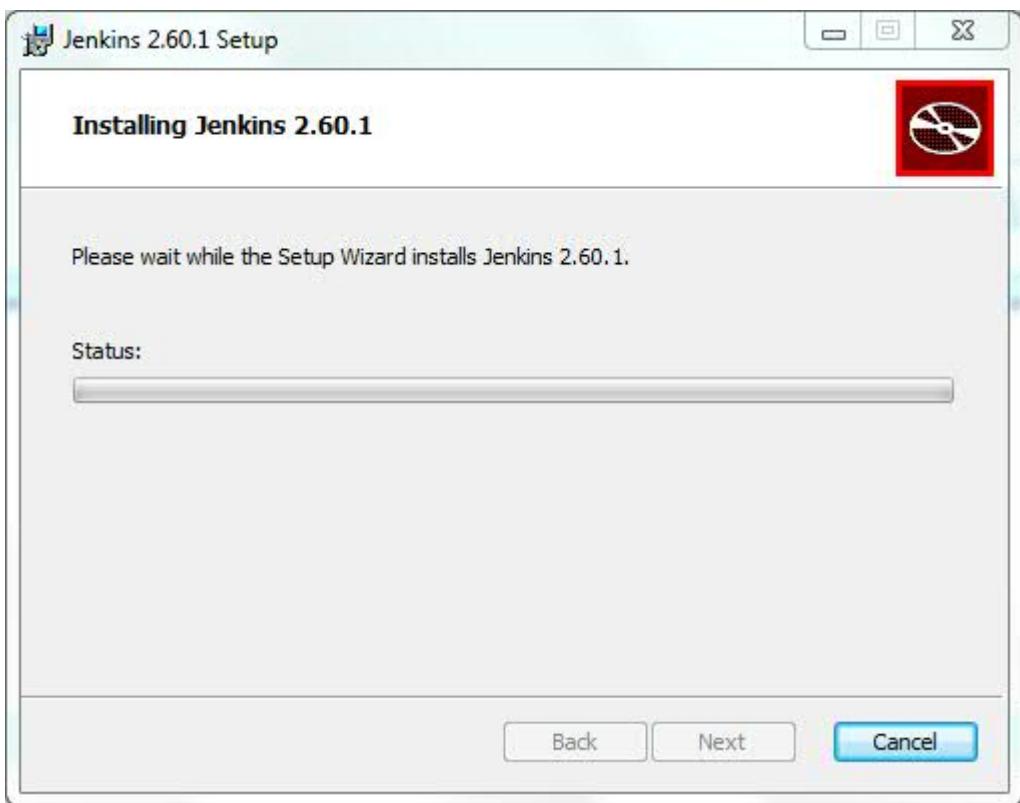
Click on Next



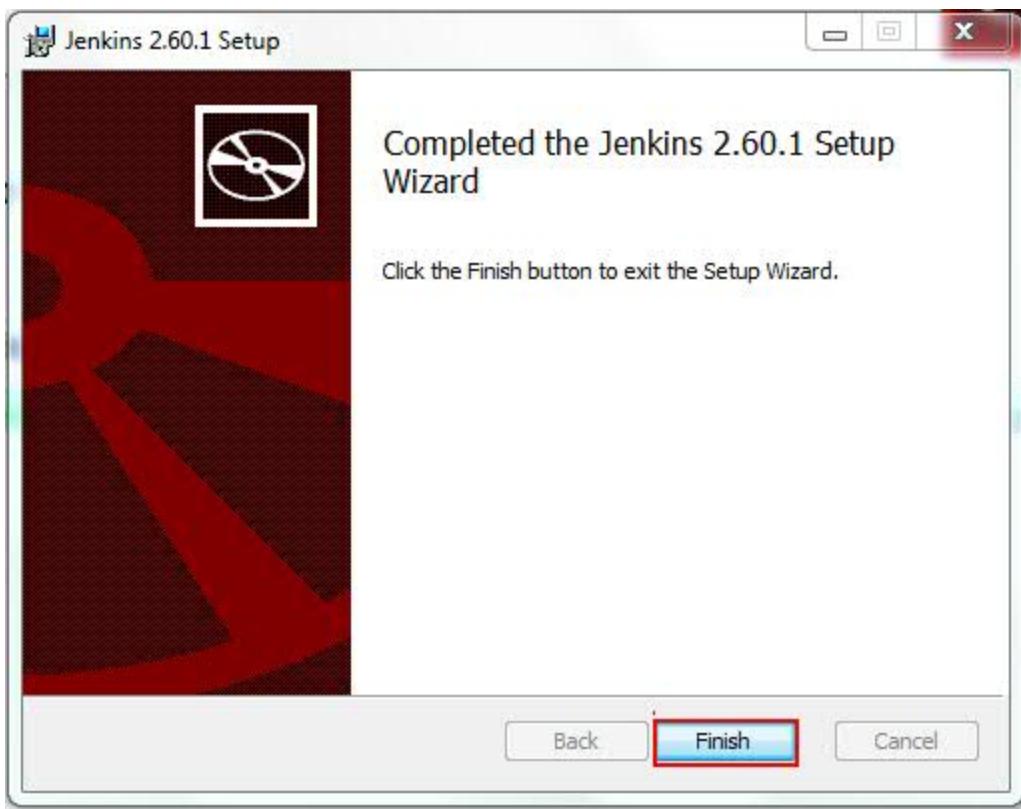
Click on Next



Click on Install



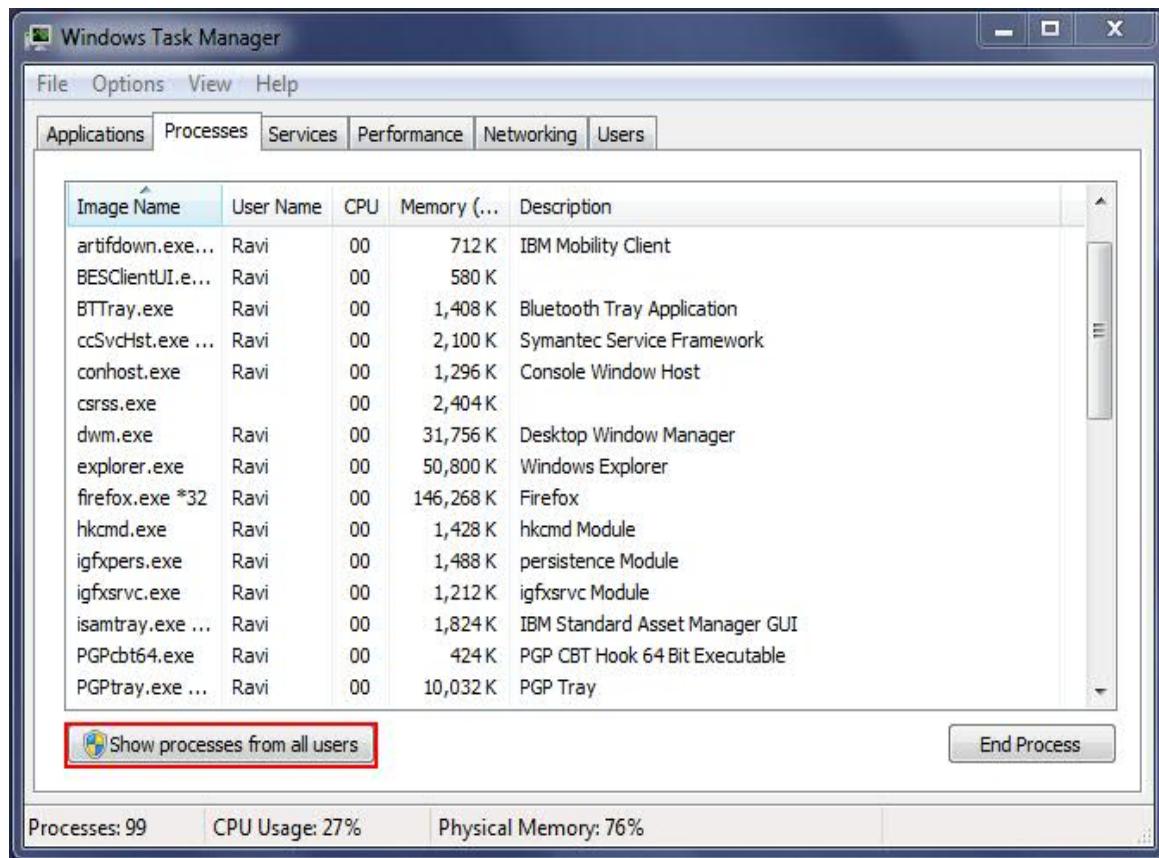
Click on Finish



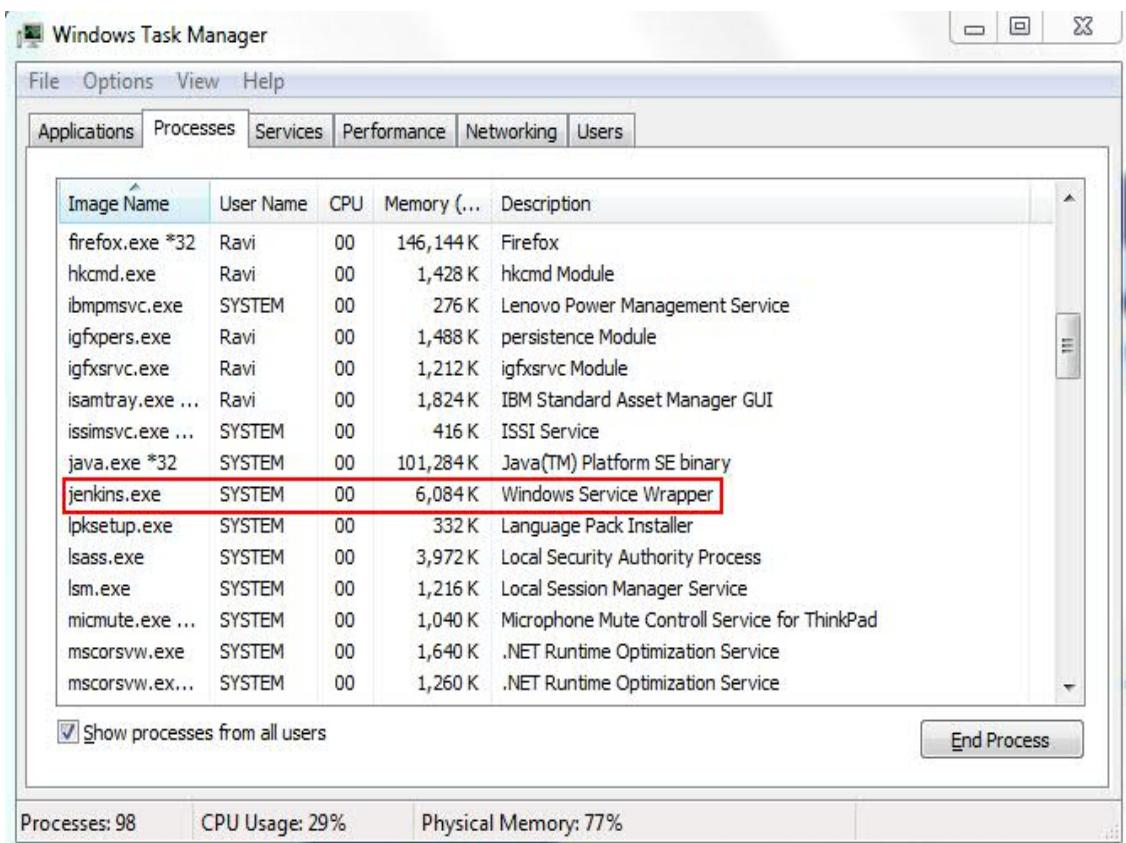
Once it successfully installed it will open by default in browser. Otherwise open the below url in browser.

<http://localhost:8080>

Once it got installed successfully, you will see Jenkins service in task manager list as follows.

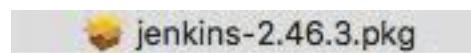


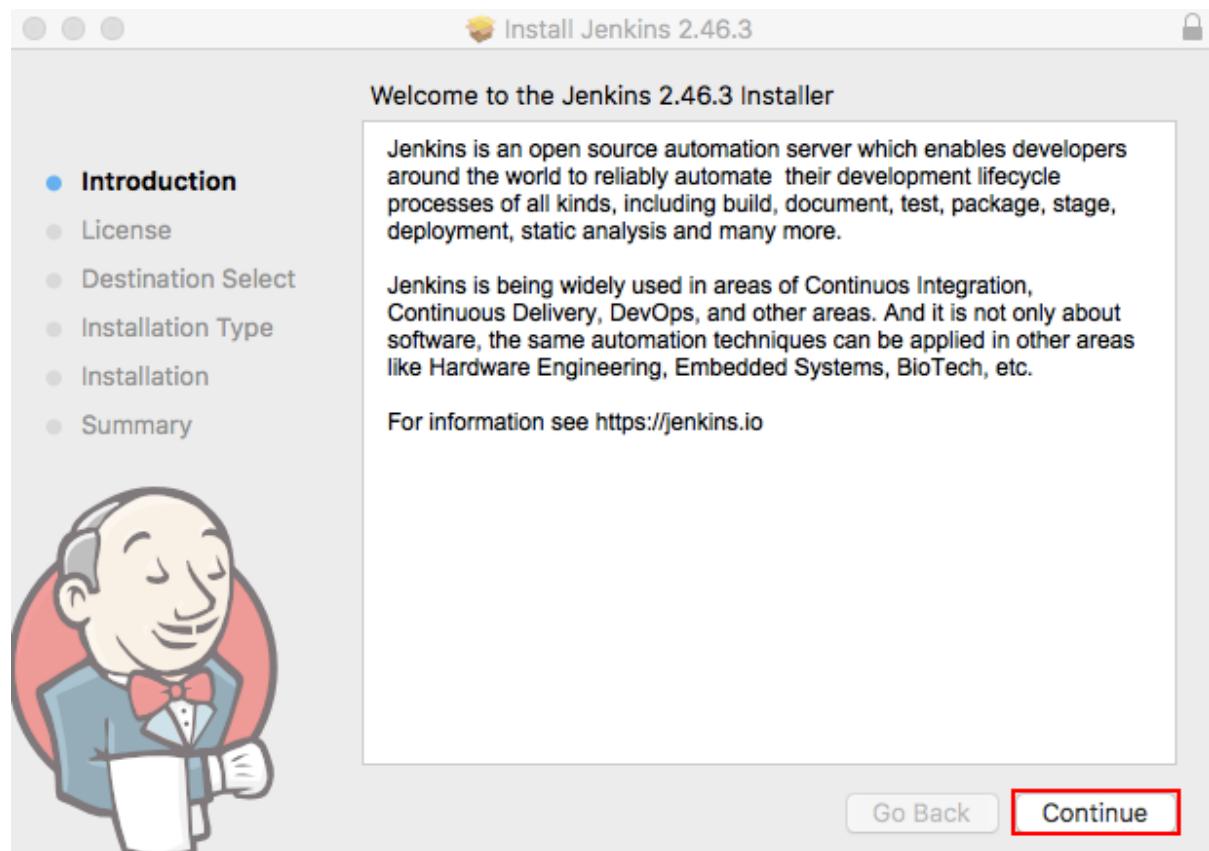
Note: Sometimes you will not see, so click on Show process from all users.



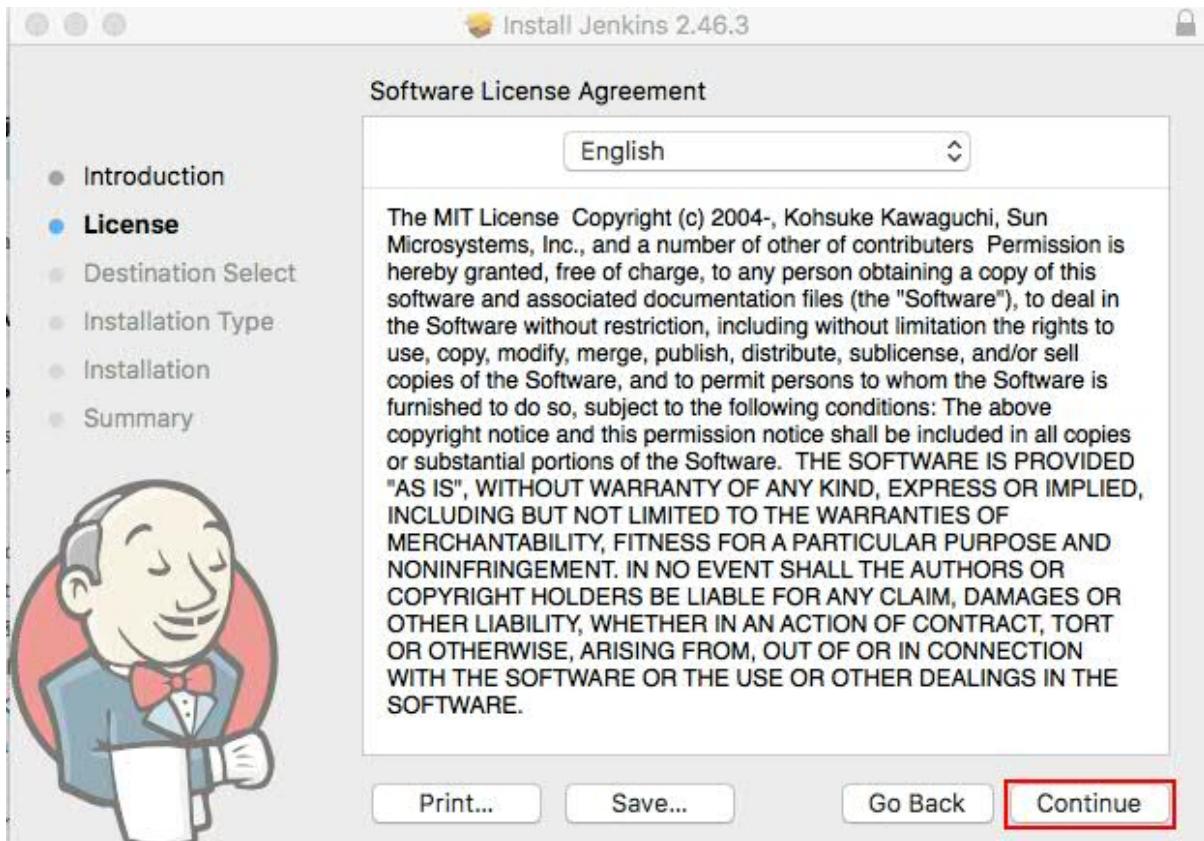
Method 3 MAC:

Click on below Jenkins package.

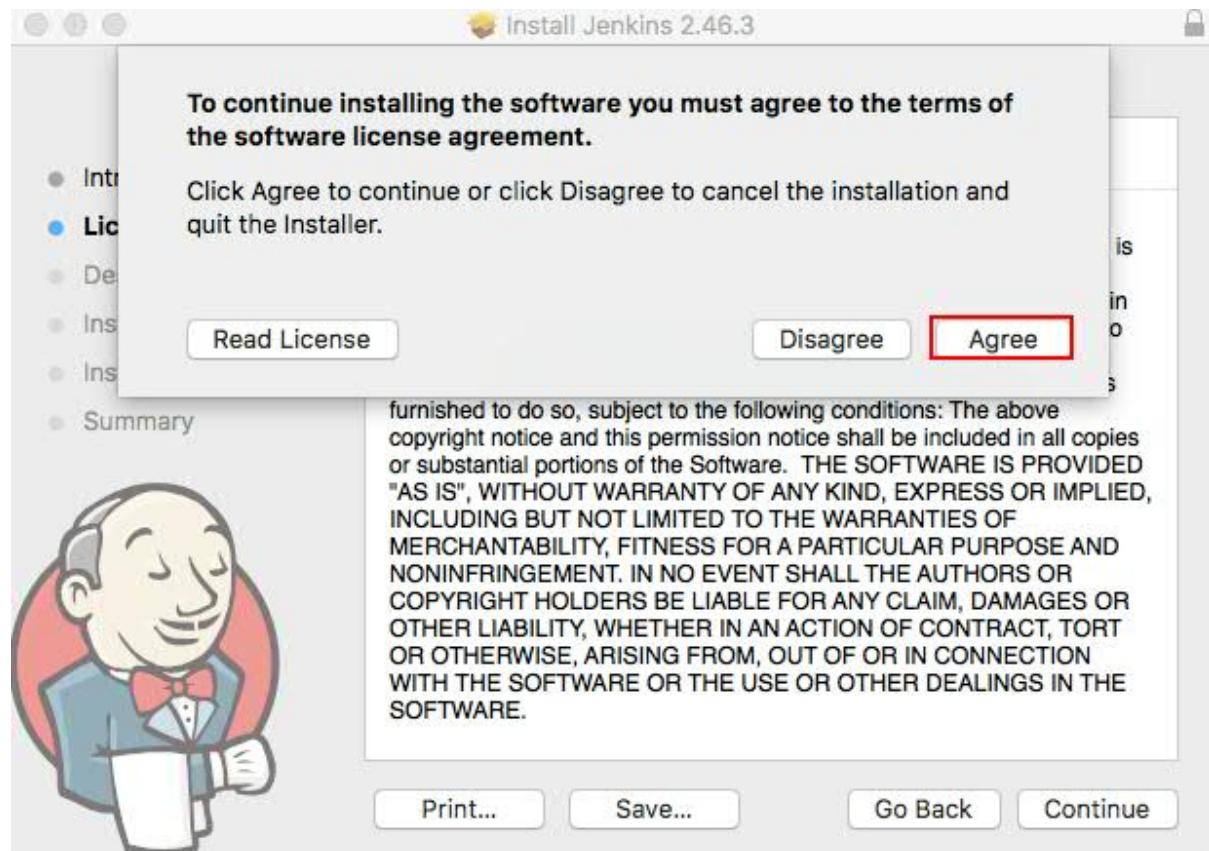




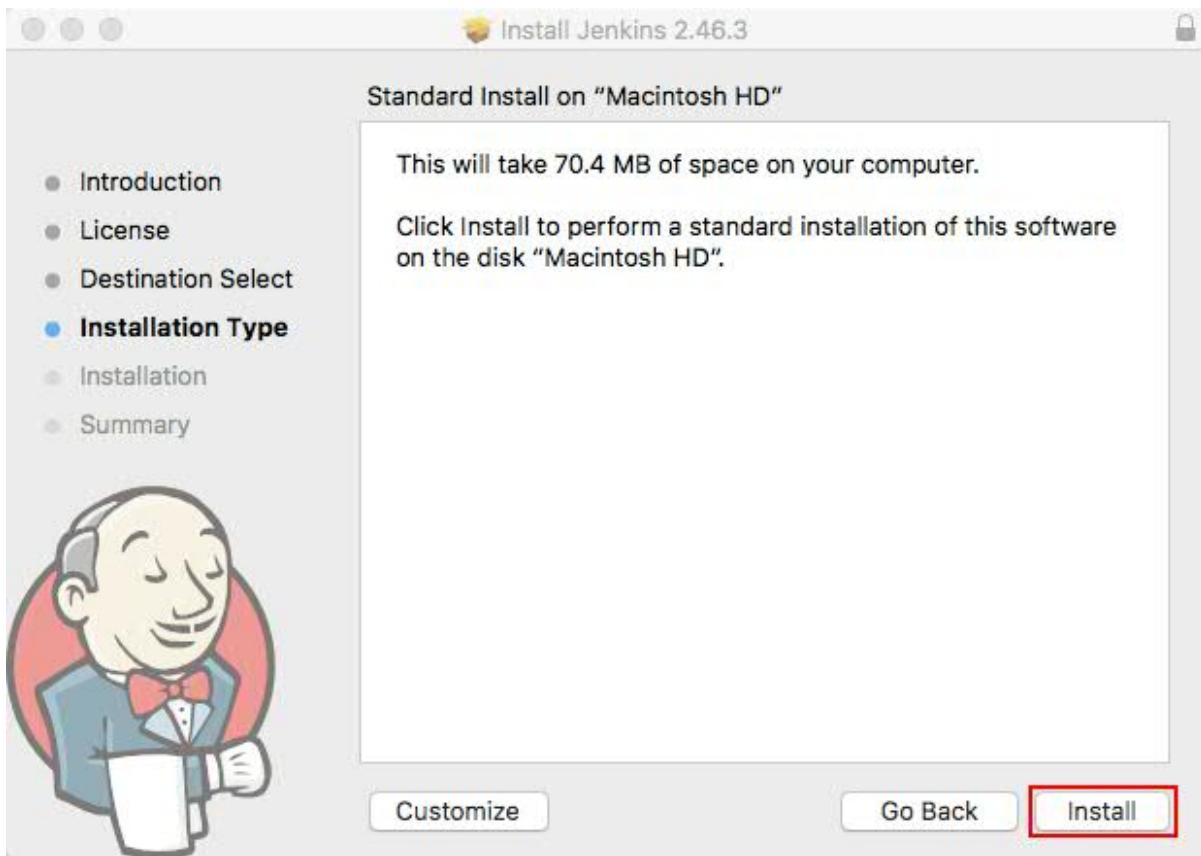
Click on Continue



Click on Continue

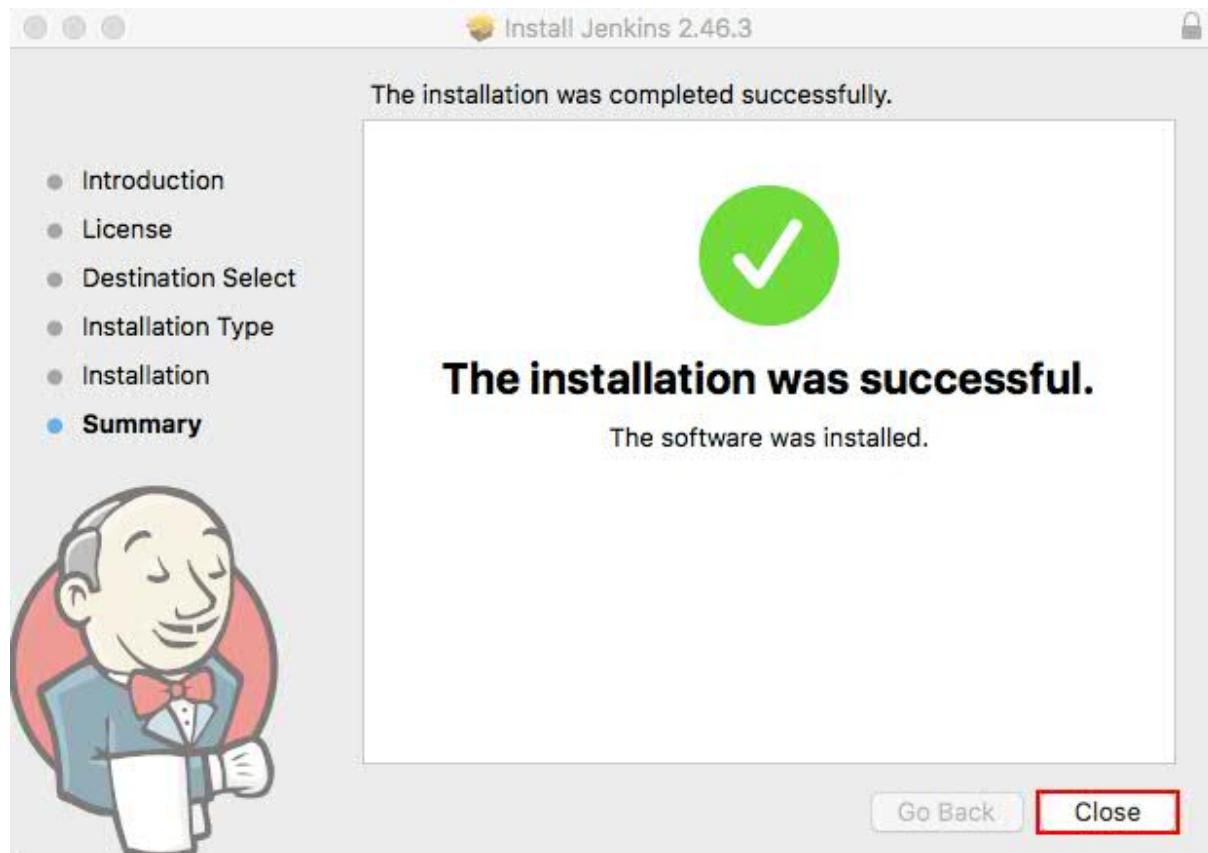


Click on Agree



Click on Install.





Click on Close.

Method 3 Linux:

```
#wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/redhat/jenkins.repo
# rpm --import https://jenkins-ci.org/redhat/jenkins-ci.org.key
# yum install Jenkins
```

Once the installation of Jenkins completed stop firewall or add the Jenkins port in Firewall in the following way.

```
# firewall-cmd --zone=public --add-port=8080/tcp --permanent
# firewall-cmd --zone=public --add-service=http --permanent
# firewall-cmd --reload
# firewall-cmd --list-all
```

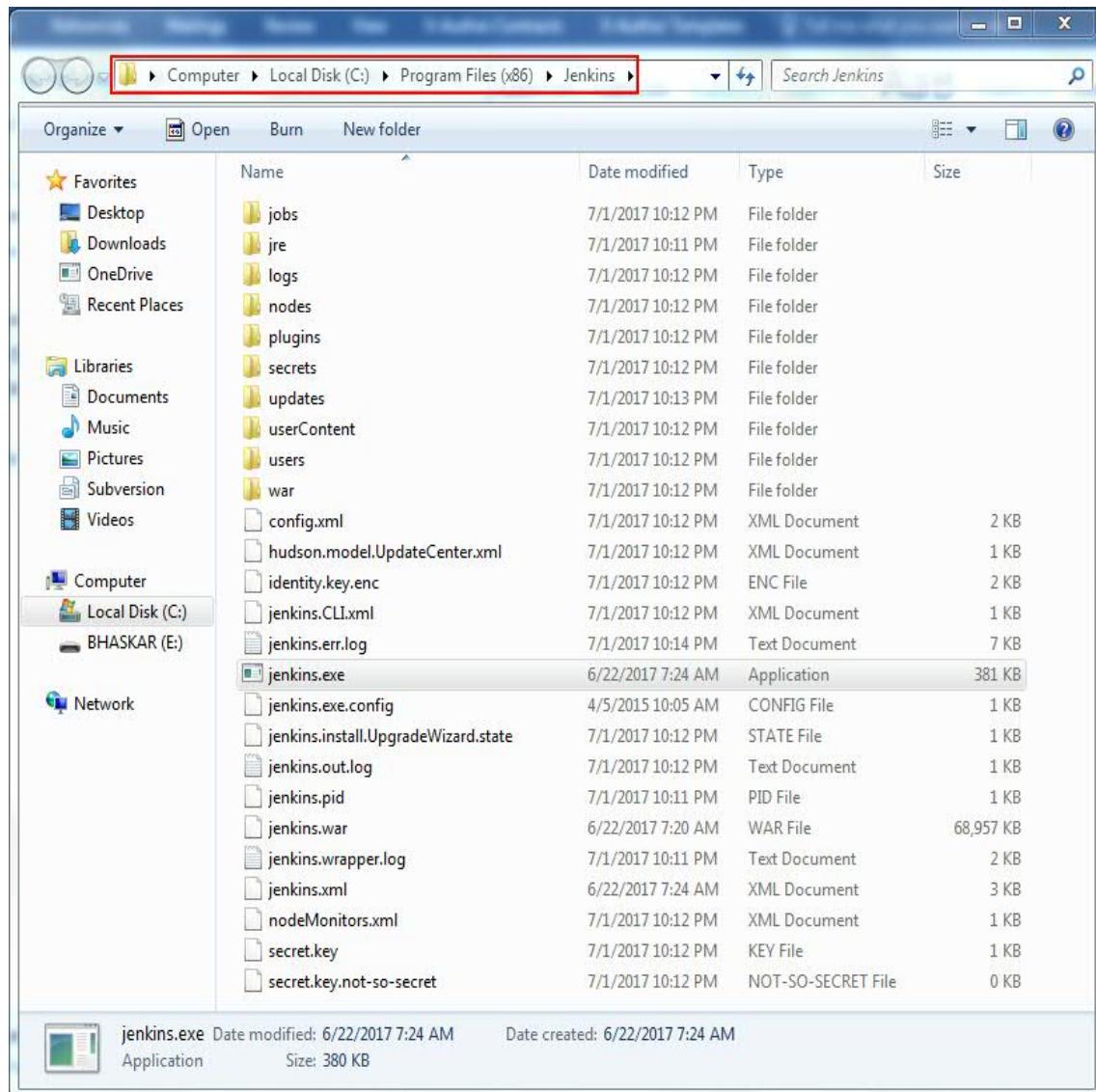
Now we can start the Jenkins Service by using systemctl as shown below

```
# systemctl start jenkins
# systemctl status Jenkins
```

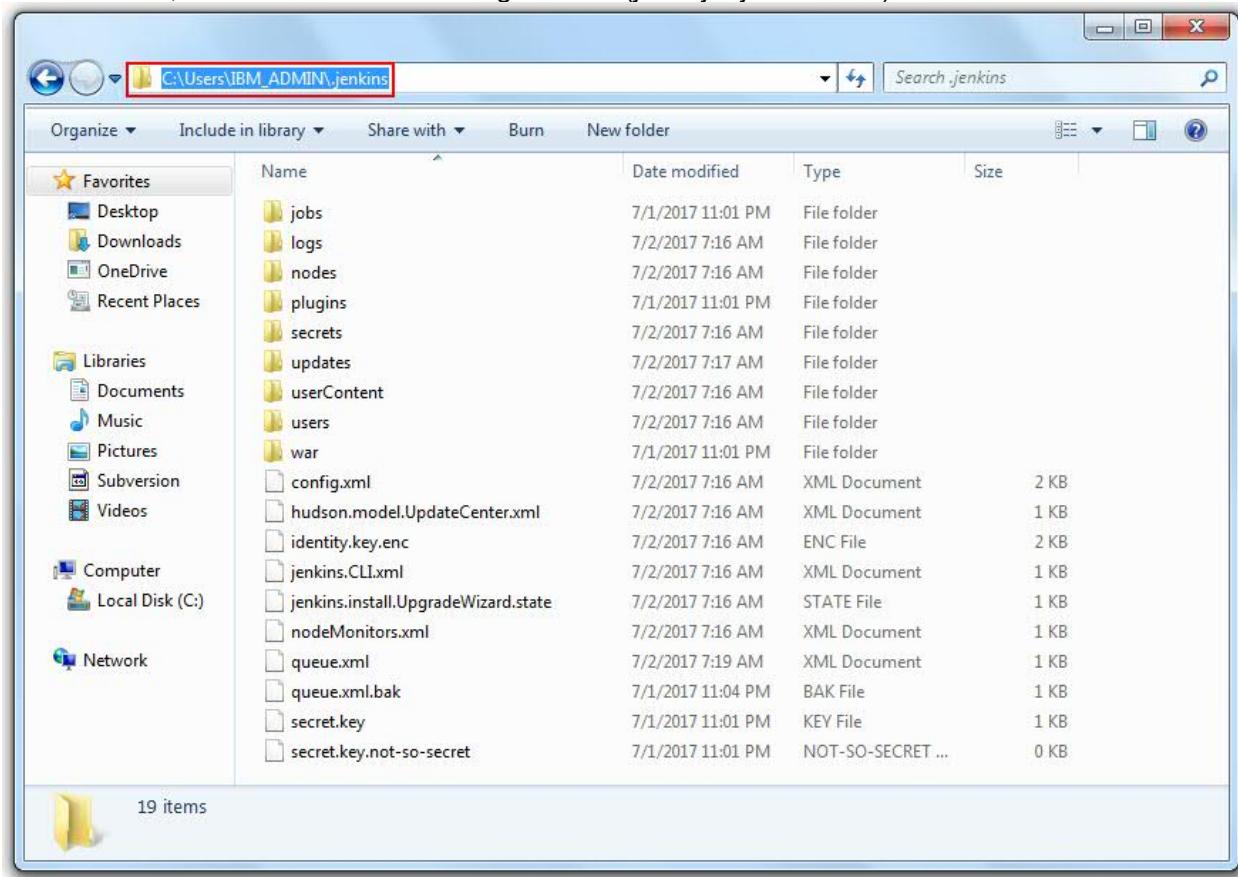
Now we can access the Jenkins webpage by using the Url: <http://ipaddress of server:8080> or <http://localhost:8080>

Reference Url: <https://wiki.jenkins.io/display/JENKINS/Installing+Jenkins+on+Red+Hat+distributions>

Below screen, installed the Jenkins using .exe file.



Below screen, installed the Jenkins using .war file (java -jar jenkins.war).



Open <http://localhost:8080/systemInfo> and check JEKNINS_HOME from browser.

.jenkins : This is the default Jenkins home directory (may be .hudson in older installations) and it will be placed in user's home directory (C:\Users\IBM_ADMIN\ ---> Windows & /Users/BhaskarReddy --> MAC).

Jenkins home directory contains the below sub directories and configuration files (.xml).

```
+- jobs
+- [JOBNAME]    :Sub directory for each job
+- config.xml   : Job configuration file
+- latest        : Symbolic link to the last successful build)
+- builds
  +- [BUILD_ID]  : for each build one build id
    +- build.xml   : build result summary
    +- log          : log file
    +- changelog.xml (change log)
+- logs          ()
+- nodes         ()
+- plugins       : This directory contains any plugins that you have installed.
+- secrets       ()
+- updates       : This is an internal directory used by Jenkins to store information
                  about available plugin updates.
```

```
+- userContent : You can use this directory to place your own custom content onto your Jenkins server. You can access files in this directory at http://localhost/jenkins/userContent (if you are running Jenkins on an application server) or http://localhost:8080/userContent (if you are running in stand-alone mode).
+- users : If you are using the native Jenkins user database, user accounts will be stored in this directory.
+- war : This directory contains the expanded web application. When you start Jenkins as a stand-alone application, it will extract the web application into this directory.
+- config.xml (jenkins root configuration)
+- *.xml (other site-wide configuration files)
+- fingerprints (stores fingerprint records)
```

<http://localhost:8080/configure>

Home directory: By default, Jenkins stores all of its data in this directory on the file system. Under the Advanced section, you can choose to store build workspaces and build records elsewhere.

There are a few ways to change the Jenkins home directory:

- Edit the JENKINS_HOME variable in your Jenkins configuration file (e.g. /etc/sysconfig/jenkins on Red Hat Linux).
- Use your web container's admin tool to set the JENKINS_HOME environment variable.
- Set the environment variable JENKINS_HOME before launching your web container, or before launching Jenkins directly from the WAR file.
- Set the JENKINS_HOME Java system property when launching your web container, or when launching Jenkins directly from the WAR file.
- Modify web.xml in jenkins.war (or its expanded image in your web container). This is not recommended.

This value cannot be changed while Jenkins is running.

It is shown here to help you ensure that your configuration is taking effect.

Ex: /Users/BhaskarReddy/.jenkins is for my Jenkins which is installed in my local MAC.

Workspace Root Directory: Specifies where Jenkins will store workspaces for builds that are executed on the master.

Build Record Root Directory: Specifies where Jenkins will store build records on the file system. This includes the console output and other metadata generated by a build.

System Message: This message will be displayed at the top of the Jenkins main page.

of executors: It shows the number of builds run at a time. E.g.: If give 2, here two builds are running.

Labels:

Usage: Controls how Jenkins schedules builds on this node.

Quiet period:

SCM checkout retry count:

Restrict project naming:

Naming Strategy

Strategy

Default ---> This is the default configuration and allows the user to choose any name they like.

Pattern ----> Define a pattern (regular expression) to check whether the job name is valid or not. Forcing the check on existing jobs, will allow you to enforce a naming convention on existing jobs - e.g. even if the user does not change the name, it will be validated with the given pattern at every submit and no updates can be made until the name confirms.

This option does not affect the execution of jobs with non-compliant names. It just controls the validation process when saving job configurations.

Global properties

Environment variables

Tool Locations

SonarQube servers

etc....

To Install any Jenkins Plugin, follow below steps

Manage Jenkins ---> Manage Plugins ---> Select the Plugin name (HTML Published plugin) ---> Install Without Restart

Package Names:

SafeRestartPlugin:

SonarQube Scanner:

JavaDoc Plugin

jQuery Plugin

Next Build Number Plugin: In UI (Set Next Build Number)

Email Extension Plugin: Is used to send the mail from Jenkins.

Delivery Pipeline Plugin: To move war/ear/jar into UCD.

Maven Integration plugin: For Maven project.

Pipeline Maven Integration Plugin: To see Maven Project option while creating job.

Run Condition Plugin

Conditional BuildStep

Parameterized Trigger plugin:

Cloud Foundry Plugin: To deploy app into Bluemix using cloud foundry.

Embeddable Build Status:

JobConfigHistory Plugin: This plugin saves a copy of the configuration file of a job (config.xml) for every change made and of the system configuration. You can also see what changes have been made by which user if you configured a security policy.

Repository browser:

Role-based Authorization Strategy:

Slack Notification Plugin:

Cobertura Plugin: In UI we will see as Coverage Trend.

Artifactory Plugin: For JFrog Or any other repositories.

Deploy to container Plugin: For deploying war/ear into Application Server or Webserver.

Build History Metrics plugin:

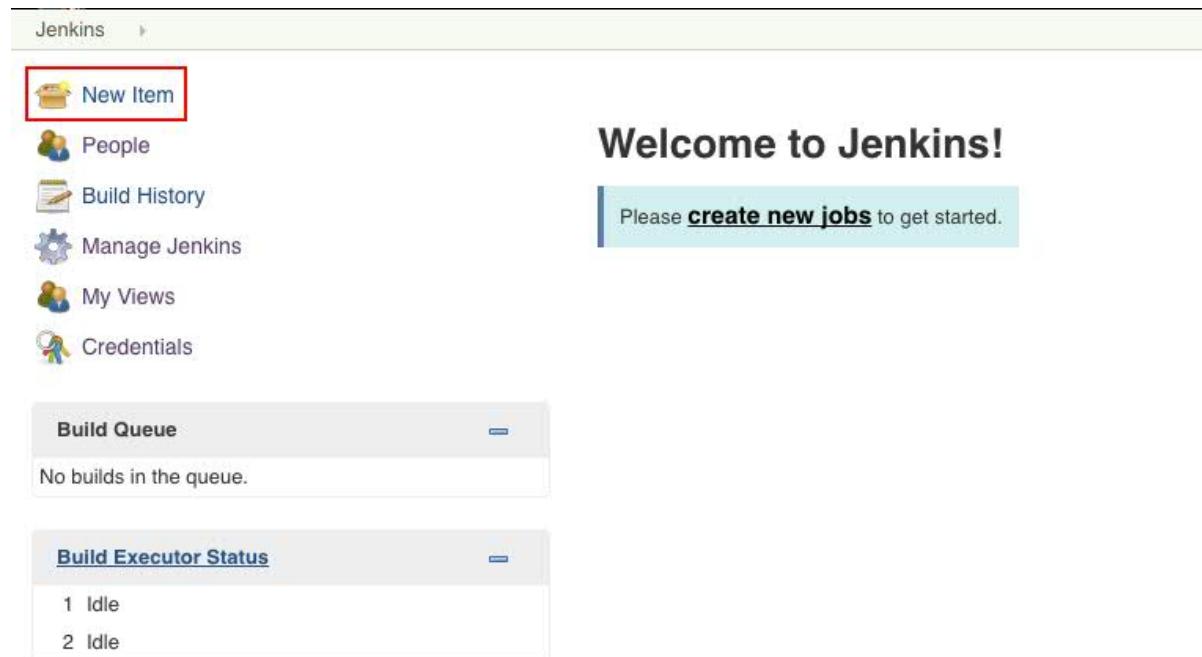
Hudson global-build-stats plugin:

Delivery Pipeline View:

Enable project-based security

Create the project/job in Jenkins

Step 1: Login into the Jenkins, go to the Jenkins dashboard left side top corner, click on **New Item**.



The screenshot shows the Jenkins dashboard. On the left, there is a sidebar with links: 'New Item' (highlighted with a red box), 'People', 'Build History', 'Manage Jenkins', 'My Views', and 'Credentials'. Below the sidebar are two sections: 'Build Queue' (which says 'No builds in the queue.') and 'Build Executor Status' (which shows '1 Idle' and '2 Idle'). On the right, the main area has a 'Welcome to Jenkins!' message and a call-to-action 'Please [create new jobs](#) to get started.'

Step 2: Enter the project name in **Enter an item name** input box and select the **Freestyle project** and click on **OK** Button.

Enter an item name

Java_Sample_ANT

» Required field

Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline

Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

External Job

This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.

Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

GitHub Organization

Scans a GitHub organization (or user account) for all repositories matching some defined markers.

Multibranch Pipeline

Creates a set of Pipeline projects according to detected branches in one SCM repository.

OK

Freestyle project: This is the central feature of Jenkins. Jenkins will build your project combining any SCM and any build system.

A Free-Style project is a project that can incorporate almost any type of build. The Free-Style project is the more "generic" form of a project. You can execute shell/dos scripts, invoke ant, and a lot more. Majority of the plugins are written to use the free-style project.

Maven project: A maven project is a project that will analyze the pom.xml file in greater detail and produce a project that's geared towards the targets that are invoked. The maven project is smart enough to incorporate build targets like the javadoc or test targets and automatically setup the reports for those targets.

Multi-configuration project: The "multiconfiguration project" (also referred to as a "matrix project") lets you run the same build job in many different configurations. This powerful feature can be useful for testing an application in many different environments, with different databases, or even on different build machines. We will be looking at how to configure multiconfiguration build jobs later on in the book.

Monitor an external job: The "Monitor an external job" build job lets you keep an eye on non-interactive processes, such as cron jobs.

General Source Code Management Build Triggers Build Environment Build Post-build Actions

Project name: Java_Sample_ANT

Description: Sample Java Web project using ANT for build.

Discard old builds

Strategy: Log Rotation

Days to keep builds: 10
if not empty, build records are only kept up to this number of days

Max # of builds to keep: 20
if not empty, only up to this number of build records are kept

Days to keep artifacts:

if not empty, artifacts from builds older than this number of days will be deleted, but the logs, history, reports, etc for the build will be kept

Max # of builds to keep with artifacts:

if not empty, only up to this number of builds have their artifacts retained

GitHub project

This project is parameterized

Throttle builds

Disable this project

Execute concurrent builds if necessary

[Advanced...](#)

General Source Code Management Build Triggers Build Environment Build Post-build Actions

Source Code Management

None
 Git

Repositories

Repository URL: <https://github.com/bhaskar0504/Ant-JavaProject.git>

Credentials: bhaskar0504/******** 

Branches to build

Branch Specifier (blank for 'any'): */master

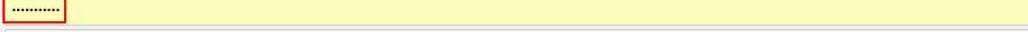
Repository browser: (Auto)

Additional Behaviours:

Subversion 

Jenkins Credentials Provider: Jenkins

Add Credentials

Domain: Global credentials (unrestricted)
Kind: Username with password
Scope: Global (Jenkins, nodes, items, all child items, etc.)
Username: bhaskar0504
Password: 
ID:
Description:

Specify when and how your build should be triggered. The following example polls the Git repository every 5 min. It triggers a build, if something has changed in the repo.

General Source Code Management **Build Triggers** Build Environment Build Post-build Actions

Subversion

Build Triggers

Trigger builds remotely (e.g., from scripts) ?
 Build after other projects are built ?
 Build periodically ?
 GitHub hook trigger for GITScm polling ?
 Poll SCM ?

Schedule H/5 * * * * ?

Would last have run at Tuesday, 27 June, 2017 6:20:22 AM IST; would next run at Tuesday, 27 June, 2017 6:25:22 AM IST.

Ignore post-commit hooks ?

Build Environment

Delete workspace before build starts ?
 Abort the build if it's stuck ?
 Add timestamps to the Console Output ?
 Use secret text(s) or file(s) ?

Build

Invoke Ant

Targets Advanced...

Add build step ▾

Once you click on Save button it will come to that particular dashboard as follows.

	Last 7 Days	21 hr
MTTR	Last 30 Days	3 days 7 hr
	All Time	3 days 7 hr
MTTF	Last 7 Days	9 hr 6 min
	Last 30 Days	1 day 7 hr
	All Time	1 day 7 hr
Standard Deviation	Last 7 Days	8.6 sec
	Last 30 Days	7.3 sec
	All Time	7.3 sec

Disable Build:

A disabled Build will not be executed until you enable it again. This option often comes in handy to temporarily suspend a build during maintenance work or major refactoring.

Once the project is configured in Jenkins then all future builds are automated. It has basic reporting features like status and weather reports (job health).

Status of the build	Description
🔴	Failed
🟡	Unstable
🟢	Success
🟠	Pending
🟤	Disabled
🟧	Aborted

Figure a: Build status

Job health	Description
☀️	No recent builds failed
🌤️	20-40% of recent builds failed
☁️	40-60% of recent builds failed
🌧️	60-80% of recent builds failed
🌩️	All recent builds failed
?	Unknown status

Figure b: Weather reports

Create the Maven project/job in Jenkins

Method 1:

Install the **Pipeline Maven Integration Plugin** and follow the below steps.

Create the Job using Freestyle project and in the Build section click on Add build step and select the Invoke Top level Maven targets.



Method 2:

Install the **Maven Integration plugin** and follow the below steps.

Create the New Item as follows.

Provide the item name and select the Maven project and click on OK.

Enter an item name

Maven-Web-ProjectName

» Required field



Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.



Maven project

Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.



Pipeline

Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.



External Job

This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.



Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.



Folder

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.



GitHub Organization

Creates a GitHub organization (or user account) for all repositories matching some defined markers.

OK

Once you click on OK, you will come to jobs configuration page as follows.

General Source Code Management Build Triggers Build Environment Pre Steps Build Post Steps Build Settings

Post-build Actions

Maven project name: **Maven-Web-ProjectName**

Description:

[Plain text] [Preview](#)

Discard old builds [?](#)

GitHub project [?](#)

This project is parameterized [?](#)

Throttle builds [?](#)

Disable this project [?](#)

Execute concurrent builds if necessary [?](#)

[Advanced...](#)

Source Code Management

None

General Source Code Management Build Triggers Build Environment **Pre Steps** Build Post Steps Build Settings

Post-build Actions

Pre Steps

Add pre-build step ▾

Build

Root POM: **pom.xml** [?](#)

Goals and options: **clean install** [?](#)

[Advanced...](#)

Post Steps

Run only if build succeeds Run only if build succeeds or is unstable Run regardless of build result

Should the post-build steps run only for successful builds, etc.

Add post-build step ▾

Once you provide all the details click on Save.

<http://localhost:8080/configureTools/>

Maven

Maven installations

Maven	Name <input type="text" value="maven"/>
<input checked="" type="checkbox"/> Install automatically	
Install from Apache	Version <input type="button" value="3.5.0"/>
Delete Installer	

[Add Installer](#)

[Delete Maven](#)

[Add Maven](#)

List of Maven installations on this system

Enable email notification

Step 1) Install Email Extension Plugin as follows.

Manage Jenkins ---> Manage Plugins ---> Install “**Email Extension Plugin**“

Step 2) Add the smtp server host as follows.

Click on Manage Jenkins ---> Configure System --->

SMTP server	<input type="text" value="smtp.gmail.com"/>
Default user E-mail suffix	<input type="text"/>
<input checked="" type="checkbox"/> Use SMTP Authentication	
User Name	<input type="text" value="devopstrainingblr@gmail.com"/>
Password	<input type="password" value="....."/>
Use SSL	<input checked="" type="checkbox"/>
SMTP port	<input type="text" value="465"/>
Charset	<input type="text" value="UTF-8"/>

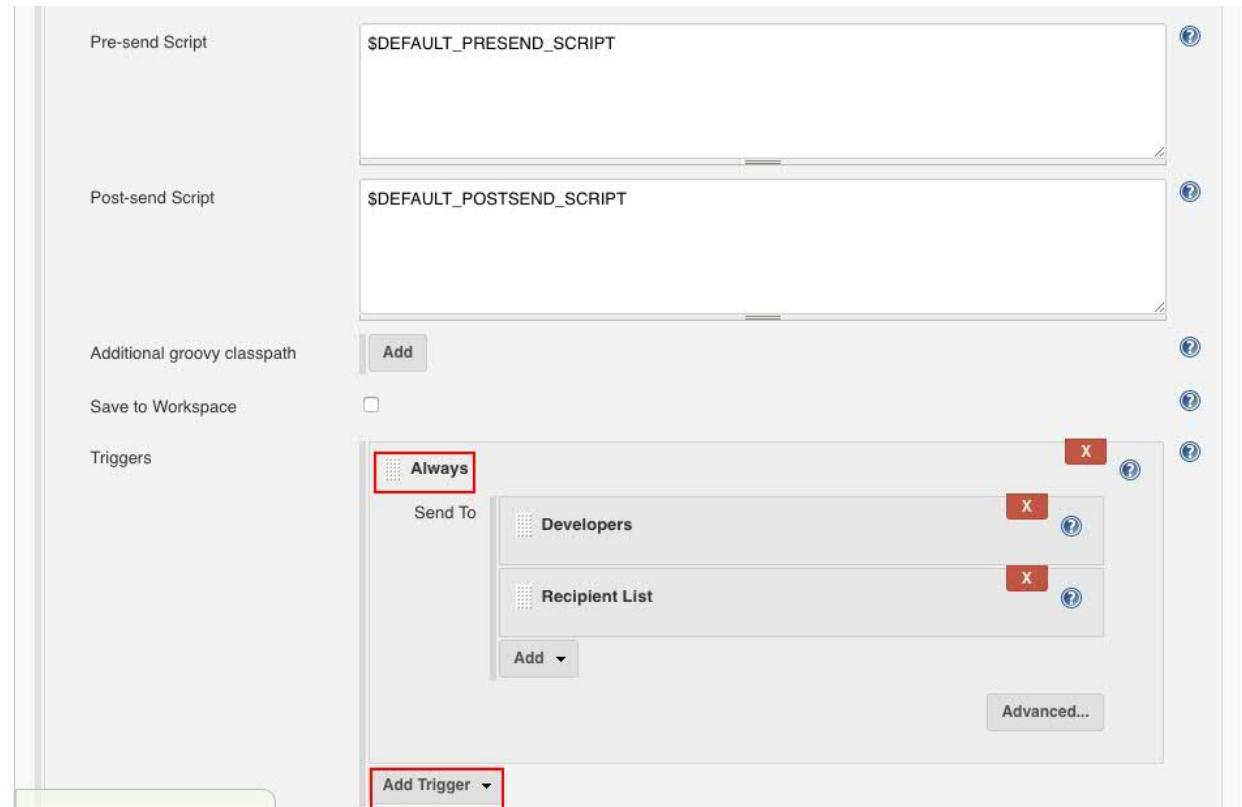
Default Content	\$PROJECT_NAME - Build # \$BUILD_NUMBER - \$BUILD_STATUS: Check console output at \$BUILD_URL to view the results.
Default Pre-send Script	
Default Post-send Script	
Additional groovy classpath	<input type="button" value="Add"/>
<input type="checkbox"/> Enable Debug Mode <input type="checkbox"/> Require Administrator for Template Testing <input type="checkbox"/> Enable watching for jobs	
<input type="button" value="Default Triggers..."/>	
Content Token Reference	

Step 3: In Job configure Editable Email as follows.

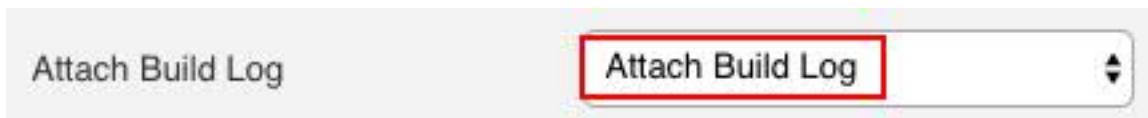
Select any Job, which we need to configure Email notification ---> Click on Configure ---> Select the **Post-build Actions** section.

Click on Advanced Settings ...

It will expand and will show more settings and click on **Add Trigger** and select the **Always**.



We can enable to attach the build logs while sending mail, as follows.



Output mail is like below.



Output mail is like below.

Configure SonarQube with Jenkins

Step 1) Install "**SonarQube Scanner for Jenkins**" for Jenkins plugin.

Step 2) Configure SonarQube as follows.

Manage Jenkins ---> Configure System ---> SonarQube servers

SonarQube servers

Environment variables

Enable injection of SonarQube server configuration as build environment variables
If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

SonarQube installations

Name	Sonarqube Server - Local
Server URL	http://localhost:9000
Server version	5.3 or higher
Server authentication token
SonarQube account login	
SonarQube account password	

SonarQube authentication token. Mandatory when anonymous access is disabled.
SonarQube account used to perform analysis. Mandatory when anonymous access is disabled. No longer used since SonarQube 5.3.
SonarQube account used to perform analysis. Mandatory when anonymous access is disabled. No longer used since SonarQube 5.3.

Advanced...

Delete SonarQube

Add SonarQube

List of SonarQube installations

Generate the SonarQube server authentication token

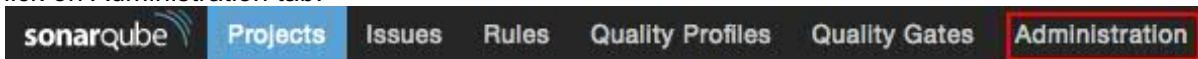
Login into SonarQube with Admin user.

<http://localhost:9000/>

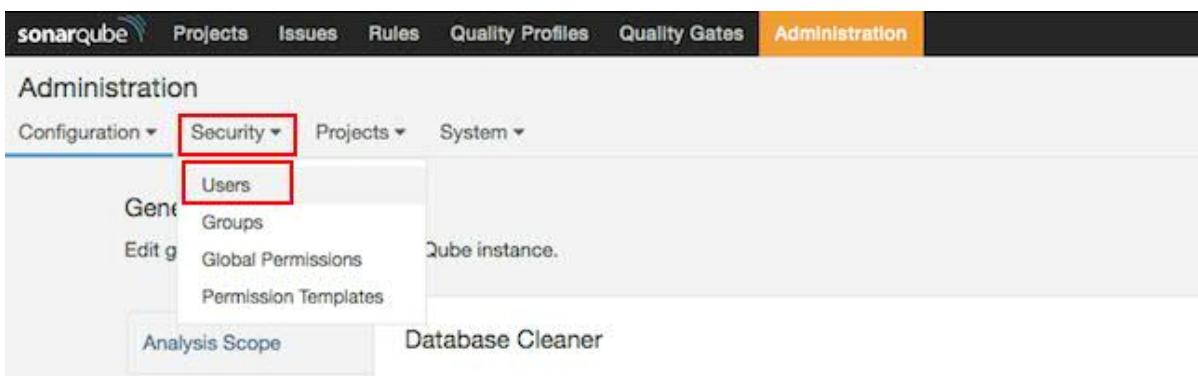
User: admin

Pwd: admin

Click on Administration tab.



Click on Security dropdown and select the Users



Click on Tokens

Q Search

SCM Accounts Groups Tokens

Administrator admin sonar-administrators sonar-users 0 Update Tokens

1/1 shown

Tokens

Name	Created
No tokens	

Generate Tokens

Jenkins - Sonarqube **Generate**

Done

Tokens

Name	Created	
Jenkins - Sonarqube	June 22, 2017	Revoke

Generate Tokens

New token "Jenkins - Sonarqube" has been created. Make sure you copy it now, you won't be able to see it again!

[Copy](#)`8e41f1f714a20ed8c77c54f3655ed863e5b19110`[Done](#)

Configure SonarQube Scanner for Job/project

General Source Code Management Build Triggers Build Environment **Build** Post-build Actions X

Execute SonarQube Scanner

Task to run

JDK (Inherit From Job)

JDK to be used for this SonarQube analysis

Path to project properties

Analysis properties

```
# must be unique in a given SonarQube instance
sonar.projectKey=Ant-Java-Project
# this is the project name displayed in the SonarQube UI
sonar.projectName=Ant-Java-Project
sonar.projectVersion=1.0
sonar.language=java
# Encoding of the source code. Default is default system encoding
sonar.sourceEncoding=UTF-8

sonar.sources=./src
```

Additional arguments

JVM Options

Add build step ▾

To restart Jenkins manually, you can use either of the following URLs:

(jenkins_url)/safeRestart - Allows all running jobs to complete. New jobs will remain in the queue to run after the restart is complete.

Ex: <http://localhost:8080/safeRestart>

(jenkins_url)/restart - Forces a restart without waiting for builds to complete.

Ex: <http://localhost:8080/restart>

(OR)

You can install one plug called **SafeRestartPlugin** , once installed it will give one option Jenkins dashboard as follows.



Jenkins - Security

How to create the users in Jenkins?

Click on Manage Jenkins ---> Manage Users ---> Create User ---> Provide the below details

Username:

Password:

Confirm password:

Full name:

E-mail address:

Click on Create User

Jenkins > Jenkins' own user database

Back to Dashboard

Manage Jenkins

Create User

Create User

Username:	devops
Password:	*****
Confirm password:	*****
Full name:	DevOps Engineer
E-mail address:	devopstrainingblr@gmail.com

How to see the list of Users in Jenkins?

Once you logged into Jenkins Dashboard

Go to Left Side Navigation Bar ---> Click on People

You will see list of users available in Jenkins.



Includes all known "users", including login identities which the current security realm can enumerate, as well as people mentioned in commit messages in recorded changelogs.

User Id	Name	Last Commit Activity	On ↓
bhaskar0504	Bhaskar Reddy L	N/A	
MANAGE_DOMAINS	MANAGE_DOMAINS	N/A	
devops	DevOps Engineer	N/A	

Icon: S M L

How to remove/delete the User in Jenkins?

Click on Manage Jenkins ---> Manage Users ---> click on below Gear icon one circle with cross symbol

It will ask Are you sure about deleting the user from Jenkins? confirmation message Click on --->
Yes

Now User is deleted successfully.

How to change the password for existing users?

Note: Need to document

Project-based Matrix Authorization Strategy is an authorization method using which we can define which user or group can do what actions on which job. This gives us a fine-grained control over user/group permissions per project.

To Enable the Project-based Matrix Authorization Strategy need to configure in Jenkins as follows.

Step 1: Click on Manage Jenkins and choose the 'Configure Global Security' option.

Step 2: Click on Enable Security option.

As an example, let's assume that we want Jenkins to maintain its own database of users, so in the Security Realm, Select the radio button of 'Jenkins' own user database'.

Step 3: Under Authorization, select "Project-based Matrix Authorization Strategy" and add 2 or 3 users, one administrator (say devops) and a regular user (say user1 and user2).



Configure Global Security

Enable security

TCP port for JNLP agents

Fixed :



Random

Disable

Agent protocols...

Disable remember me

Access Control

Security Realm

Delegate to servlet container

Github Authentication Plugin

Gitlab Authentication Plugin

HTTP Header by reverse proxy

Jenkins' own user database

Allow users to sign up

LDAP

Unix user/group database

Authorization

Anyone can do anything

The screenshot shows the Jenkins Authorization Strategy configuration page. At the top, there are four radio button options: 'Anyone can do anything', 'Legacy mode', 'Logged-in users can do anything', and 'Matrix-based security'. The fourth option is selected and highlighted with a red border. Below this is a section titled 'Project-based Matrix Authorization Strategy'.

The main area is a matrix table where rows represent 'User/group' (Administrator, devops, user1, user2, Anonymous) and columns represent various Jenkins operations (Overall, Credentials, Agent, Job, Run, View, SCM, etc.). Each cell in the matrix contains a checkbox. A red box highlights the row for 'devops' and the column for 'Read' under 'Job', indicating that 'devops' has been granted read permission for all jobs. Another red box highlights the row for 'Anonymous' and the column for 'Read' under 'Overall', indicating that 'Anonymous' has been granted read permission for all Jenkins operations.

Below the matrix table, there is a text input field labeled 'User/group to add:' and a 'Save' button. The 'Save' button is highlighted with a red border. There is also an 'Apply' button and a 'Plain text' dropdown menu.

All the checkboxes present besides users are for setting global permissions. Select all checkboxes against admin user to give admin full permissions.

For user1, we are selecting read permissions under jobs. With this, user1 would now have read permission to view all jobs which we would be creating later on.

We have to provide read permission under "Overall" category to any regular user otherwise the user won't be able to see anything after login.

All the checkboxes present besides users are for setting global permissions. Select all checkboxes against admin user to give admin full permissions. For user1, we are selecting read permissions under jobs. With this, user1 would now have read permission to view all jobs which we would be creating later on. We have to provide read permission under "Overall" category to any regular user otherwise the user won't be able to see anything after login.

Finally, you can click on Save button.

Below scenario will applicable in Matrix based security

Error : Access Denied

<<User>> is missing the Overall/Read permission

If you get this error, Please follow below steps.

Solution:

Click on Manage Jenkins ---> Configure Global Security ---> User/group to add: Enter the user Name and click on Add button and --->
Enable the appropriate feature ---> Click on Save Button.

Jenkins Build Status Icon Colours



BLUE: Success



GREY: Disabled, Aborted and Pending



YELLOW: Unstable



RED: Failed

Jenkins Job Weather Status Icon Colours



60-80 of recent builds failed.



No recent builds failed.

Note: Need to do more document on these icons.

Status of the build	Description
🔴	Failed
🟡	Unstable
🟢	Success
🟠	Pending
🟤	Disabled
🟧	Aborted

Figure a: Build status

Job health	Description
☀️	No recent builds failed
🌤️	20-40% of recent builds failed
☁️	40-60% of recent builds failed
🌧️	60-80% of recent builds failed
🌩️	All recent builds failed
❓	Unknown status

Figure b: Weather reports

How to enable the hooks in Jenkins?

Step 1: Install the “Git plugin” in Jenkins.

Step 2: Select the job which you need to enable hook and click on Configure ---> In **Build Triggers** Section enable the **Poll SCM**
And provide the values as follows.

Build Triggers

- Trigger builds remotely (e.g., from scripts)
- Build after other projects are built
- Build periodically
- GitHub hook trigger for GITScm polling
- Poll SCM

Schedule: `0 */* * * *`

⚠ Spread load evenly by using 'H */22 * * *' rather than '0 */22 * * *'.
Would last have run at Thursday, 6 July, 2017 12:00:07 AM IST; would next run at Thursday, 6 July, 2017 10:00:07 PM IST.

Ignore post-commit hooks

Step 3: Login into <https://github.com>

Step 4: Select the project/repository which you need to enable the hooks. ---> Click on Settings ---->Click on Integrations & services ---> Click on Add service and select the Jenkins (Git plugin) from Available Services ---> Provide the Jenkins url (here we are using <http://localhost:8080/>) and click on **Add service**.

The screenshot shows the GitHub 'Installed GitHub Apps' page for the repository 'devops-bangalore / Ant-WebProject'. The 'Integrations & services' tab is selected in the sidebar. A modal window titled 'Available Services' is open, showing three options: 'Jenkins', 'Jenkins (Git plugin)', and 'Jenkins (GitHub plugin)'. The 'Jenkins (Git plugin)' option is highlighted with a red border.

Installed GitHub Apps

GitHub Apps augment and extend your workflows on GitHub with commercial, open source, and homegrown tools.

Services

Services are pre-built integrations that perform certain actions when events occur.

Add service

Jenkins url

http://localhost:8080/

Active
We will run this service when an event is triggered.

Add service

Jenkins with slack integration

Install the **Slack Notification Plugin**.

Slack:

Go to the Slack App with below url

<https://devops-team-bangalore.slack.com/apps>

In the search box type Jenkins CI, you will get the Jenkins CI and select that.

Click on Install

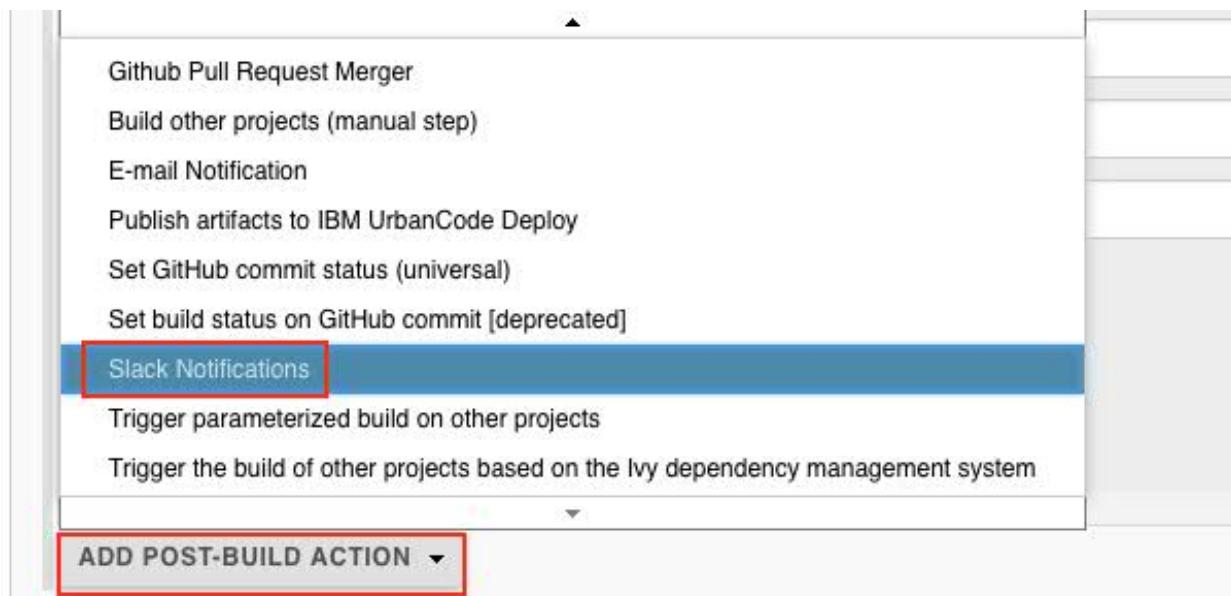
Select the channel in Post to Channel drop down.

Click on Add Jenkins CI Integration

Click on Save Settings.

Jenkins:

Open the job which you want to configure Slack Notifications and click on Configure and in **Post-build actions** tab, click on **ADD POST-BUILD ACTION** and select the **Slack Notifications** as follows.



Provide the below details and click on Save.

The screenshot shows the Jenkins Slack Notifications configuration page. At the top, there is a red 'X' button. Below it, a section titled 'Slack Notifications' contains a list of notification types, each with a checked checkbox:

- Notify Build Start
- Notify Aborted
- Notify Failure
- Notify Not Built
- Notify Success
- Notify Unstable
- Notify Regression
- Notify Back To Normal
- Notify Repeated Failure
- Include Test Summary
- Include Failed Tests

Below this, there is a checkbox labeled 'Include Custom Message' which is unchecked. A dropdown menu next to it shows the value 'nothing about commits'. A note below the dropdown says 'What commit information to include into notification message'.

Further down, there are fields for 'Base URL' (set to <https://devops-team-bangalore.slack.com/hooks/>), 'Team Subdomain' (set to devops-team-bangalore), and 'Integration Token' (set to 1QnI7MV7ISzHtCdPI1mnwBVE). A warning message '⚠ Exposing your Integration Token is a security risk. Please use the Integration Token Credential ID' is displayed above the 'Integration Token Credential ID' dropdown, which has '- none -' selected. There is also a 'TEST CONNECTION' button.

At the bottom left, there is a 'ADD POST-BUILD ACTION' button.

Deploy the application into Bluemix

Install the “**Cloud Foundry Plugin**” plugin.

Open the job which you want to configure deploy, and click on Configure and in **Post-build actions** tab, click on **ADD POST-BUILD ACTION** and select the **Push to Cloud Foundry** as follows.

Post-build Actions

- Aggregate downstream test results
- Archive the artifacts
- Build other projects
- Publish JUnit test result report
- Publish Javadoc
- Push to Cloud Foundry**
- Record fingerprints of files to track usage
- Stop Docker Containers
- Git Publisher
- Github Pull Request Merger
- Build other projects (manual step)
- E-mail Notification
- Publish artifacts to IBM UrbanCode Deploy
- Set GitHub commit status (universal)
- Set build status on GitHub commit [deprecated]
- Slack Notifications
- Trigger parameterized build on other projects
- Trigger the build of other projects based on the Ivy dependency management system

ADD POST-BUILD ACTION ▾

As soon as clicked on Push to Cloud Foundry option you will get the below screen.

General Source Code Management Build Triggers Build Environment Build Post-build Actions

Post-build Actions

Push to Cloud Foundry

Target: <https://api.w3ibm.bluemix.net>

Credentials: bhlaccha@in.ibm.com/*****

Organization: DSGPractice

Space: dev

Allow self-signed certificate:

TEST CONNECTION

Reset app if already exists:

Plugin timeout (s): 120

Create services before pushing: **ADD**

Read configuration from a manifest file

Enter configuration in Jenkins

Application Name: Ant-Web-Project

Memory (MB): 512

Hostname: Ant-Web-Project

Instances: 1

Timeout (s): 60

Custom buildpack:

Custom stack:

Environment Variables: **ADD**

Services: **ADD**

ADVANCED...

ADD POST-BUILD ACTION ▾

Click on TEST CONNECTION, If connection got succeed and click on Save button.

Deploy the application into Bluemix

Install the “**Deploy to container Plugin**” plugin.

Open the job which you want to configure deploy, and click on Configure and in **Post-build actions** tab, click on **ADD POST-BUILD ACTION** and select the **Deploy war/ear to container** as follows.

The screenshot shows the Jenkins configuration interface for a job. The top navigation bar includes General, Source Code Management, Build Triggers, Build Environment, Artifactory Configuration, Build, and Post-build Actions. The Post-build Actions tab is active. Under the 'Gradle-Artifactory Integration' section, the 'Deploy war/ear to a container' action is highlighted with a red box. Other available actions include Aggregate downstream test results, Archive the artifacts, Build other projects, Publish JUnit test result report, Publish Javadoc, Push to Cloud Foundry, Record fingerprints of files to track usage, Git Publisher, E-mail Notification, Editable Email Notification, Set GitHub commit status (universal), Set build status on GitHub commit [deprecated], Trigger the build of other projects based on the Ivy dependency management system, and Delete workspace when build is done. At the bottom of the list is an 'Add post-build action' dropdown.

This screenshot shows the detailed configuration for the 'Deploy war/ear to a container' action. It specifies WAR/EAR files as '**/*.war' and the Context path as /ant. Under Containers, it is set to Tomcat 7.x with Manager user name admin and Manager password (redacted). The Tomcat URL is http://localhost:8081/. There is an 'Add Container' button at the bottom of the container list. A 'Deploy on failure' checkbox is also present. The 'Save' and 'Apply' buttons are at the bottom of the configuration panel.

(OR)

Build

Invoke Ant Targets Advanced...

Add build step ▾ Execute Windows batch command

Execute shell (highlighted)

- Invoke Ant
- Invoke Artifactory Maven 3
- Invoke Gradle script
- Invoke top-level Maven targets
- Provide Configuration files
- Run with timeout
- Set build status to "pending" on GitHub commit

Add the below script in **Execute shell**

```
#!/bin/sh
echo "Starting to copy the build"
scp $WORKSPACE/war/SampleAntProject.war /opt/apache-tomcat-7.0.78/webapps
echo "Copied the build to tomcat"
```

Build

Invoke Ant Targets Advanced...

Execute shell

Command

```
#!/bin/sh
echo "Starting to copy the build"
scp $WORKSPACE/war/SampleAntProject.war /opt/apache-tomcat-7.0.78/webapps
echo "Copied the build to tomcat"
```

See the list of available environment variables Advanced...

Add build step ▾

Note: If we want to deploy in Tomcat, which is deployed any remote machine, use below line of code.

```
$WORKSPACE/war/SampleAntProject.war <>User Name<>@<>ServerIP<>:/opt/apache-tomcat-7.0.78/webapps
```

Integrate JFrog Artifactory with Jenkins

Install “**Artifactory Plugin**” plugin.

Got to the Manage Jenkins ---> Configure System --->
In the **Artifactory** section fill the below details and click on Save.

The screenshot shows the Jenkins 'Configure System' page with the 'Artifactory' section selected. The 'Artifactory servers' section contains one entry:

- Server ID:** JFrog Artifactory server1 (highlighted with a red box)
- URL:** http://localhost:8081/artifactory/ (highlighted with a red box)
- Default Deployer Credentials:**
 - Username:** admin (highlighted with a red box)
 - Password:** (highlighted with a red box)
- Connection Timeout:** 300
- Number of retries:** 3
- Bypass HTTP Proxy:**
- Test Connection:** A button labeled "Found Artifactory 5.3.2" (highlighted with a red box) is shown next to the "Test Connection" button.
- Use Different Resolver Credentials:**
- Delete:** A red "Delete" button.

At the bottom left is a "Add Artifactory Server" button.

Note: Once you entered all the details click on **TEST CONNECTION**. IF connection is succeed you will see the message like **Found Artifactory <<Version>>**.

Jenkins – Metrics and 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

Enable LDAP security to Jenkins

<http://www.scmgalaxy.com/tutorials/complete-guide-to-use-jenkins-cli-command-line>

Jenkins CLI

Jenkins has a built-in command line interface (CLI) that allows users and administrators to access Jenkins from a script or shell environment. This can be convenient for scripting of routine tasks, bulk updates, troubleshooting, and more.

Advantages of Jenkins CLI:

- Easier
- Faster
- Memory management
- Automation tasks.

Pre-Requisites

- a) Jenkins server is running
- b) Enable security as follows.

Go to jenkins dashboard in Home page (e.g <http://localhost:8080/>) -> Manage Jenkins

-> Configure Global Security -> Click on “Enable security” checkbox

You can also configure “Access Control” and “Authorization” option in Global Security page.

Download the Jenkins CLI jar file as follows.

Method 1

Open the below url

<http://localhost:8080/cli/>



Jenkins CLI

You can access various features in Jenkins through a command-line tool. See [the documentation](#) for more details of this feature. To get started, download [jenkins-cli.jar](#) and run it as follows:

```
java -jar jenkins-cli.jar -s http://localhost:8080/ help
```

Click on Jenkins-cli.jar.

Method 2

Click on below url, it will automatically download the jar file.

<http://<<Jenkins Server URL>>/jnlpJars/jenkins-cli.jar>

Example: <http://localhost:8080/jnlpJars/jenkins-cli.jar>

Here
Copy into any folder as follows

```
#cp jenkins-cli.jar /opt/jenkins/
```

Go to the directory where Jenkins-cli.jar is there and run the below command to get the help.

Login Jenkins using username and Password

```
# java -jar jenkins-cli.jar -s http://localhost:8080/ help --username devops --password passw0rd
```

Get the Version of Jenkins

```
#java -jar jenkins-cli.jar -s http://localhost:8080/ version --username devops --password passw0rd
```

Get all the jobs of Jenkins

```
#java -jar jenkins-cli.jar -s http://localhost:8080/ list-jobs --username devops --password passw0rd
```

Delete the Job

```
#java -jar jenkins-cli.jar -s http://localhost:8080/ delete-job ant-java-job-dev --username devops --password passw0rd
```

```
#java -jar jenkins-cli.jar -s http://localhost:8080/ disable-job ant-web-job-dev --username devops --password passw0rd
```

```
#java -jar jenkins-cli.jar -s http://localhost:8080/ enable-job ant-web-job-dev --username devops --password passw0rd
```

While executing above command if you see Enter passphrase, follow the below configuration.
(Manage Jenkins ---> Configure Global Security ---> enable the Enable Security ---> Apply and Save.)

Manage Jenkins ---> Configure System --->SSH Public Keys (Enter here any value, same value u can use in CLI)

UrbanCode Deploy with Jenkins

Download the UCD Plugin from below url.

<https://developer.ibm.com/urbancode/plugin/jenkins/> (ibm-ucdeploy-publisher-1.9.925586.hpi) --->
Old version URL

<https://developer.ibm.com/urbancode/plugin/jenkins-2-0/> (ibm-ucdeploy-build-steps-2.6.929921.hpi)
---> New version URL

Once you download the UCD plugin, install the that plugin in Jenkins as follows.

Manage Jenkins ---> Manage Plugins ---> Got to the "**Advanced**" tab ---> Go to the "**Upload Plugin**" section and click on **Choose file** and select the UCD plugin (ibm-ucdeploy-publisher-1.9.925586.hpi OR ibm-ucdeploy-build-steps-2.6.929921.hpi) ---> Click on "**Upload**"

Upload Plugin

You can upload a .hpi file to install a plugin from outside the central plugin repository.

File: Choose file **ibm-ucdeploy-publisher-1.9.925586.hpi**

Upload

Installing Plugins/Upgrades

Preparation

ibm-ucdeploy-publisher  Success

 [Go back to the top page](#)

(you can start using the installed plugins right away)

 Restart Jenkins when installation is complete and no jobs are running

Integrate the UrbanCode Deploy Server as follows.

Manage Jenkins ---> Configure System ---> Got the "IBM UrbanCode Deploy Pipeline Plugin Configuration" section and click on **Add** button and provide the UCD details as follows, and click on **Save** button.

IBM UrbanCode Deploy Pipeline Plugin Configuration

UCD Servers

Add

IBM UrbanCode Deploy Pipeline Plugin Configuration

UCD Servers	Profile Name	UCD Server -1
	IBM UrbanCode Deploy URL	https://ucd.mithuntechnologies.com:8443
	User Name	mithunreddy@mithuntechnologies.com
	Password
	Administrative User	<input type="checkbox"/>
	Trust All Certificates	<input type="checkbox"/>
	Success	Delete
		Test Connection

If you provide wrong UCD server, you will get the below error message.

 ucd.mithuntechnologies.com: nodename nor servname provided, or not known

If you provide the wrong Credentials, you will get the below error message.

 Error connecting to IBM UrbanCode Deploy: Invalid user and/or password

Once you have integrated UCD with Jenkins, you can enable UCD for each job as follows.
Go to the **Post-build Actions** section and select the **Publish artifacts to IBM UrbanCode Deploy**.

Note: If you have install **ibm-ucdeploy-publisher-1.9.925586.hpi** Plugin, you will see the **Publish artifacts to IBM UrbanCode Deploy** option in **Post-build Actions** section.

If you have install **ibm-ucdeploy-build-steps-2.6.929921.hpi** Plugin, you will see the **Publish artifacts to IBM UrbanCode Deploy** option in Build section.

General Source Code Management Build Triggers Build Environment Build **Post-build Actions**

Post-build Actions

Add post-build action ▾

- Publish Checkstyle analysis results
- Aggregate downstream test results
- Archive the artifacts
- Build other projects
- Publish JUnit test result report
- Publish Javadoc
- Record fingerprints of files to track usage
- Sonargraph Integration Report Generation & Analysis
- Git Publisher
- SonarQube analysis with Maven
- E-mail Notification
- Editable Email Notification
- Publish artifacts to IBM UrbanCode Deploy**
- Set GitHub commit status (universal)
- Set build status on GitHub commit [deprecated]
- Trigger the build of other projects based on the Ivy dependency management system
- Delete workspace when build is done

As soon as you select the **Publish artifacts to IBM UrbanCode Deploy** option you will get the below screen.

Provide the UCD Details.

The screenshot shows the Jenkins build configuration for a project. The 'Build' tab is selected. Under 'Publish Artifacts to IBM UrbanCode Deploy', the 'Urban Code Deploy Server' is set to 'Urban Code Deploy Server1'. The 'Create Component Version' checkbox is checked. The 'Component Name' is 'Demo App Comp'. Under 'Delivery Mechanism', the 'Push Files' section includes 'Version Name' as '\${BUILD_NUMBER}', 'Base Artifact Directory' as '\${WORKSPACE}/dist', and 'Include' as '**/*'. There is also an 'Exclude' section. Below these, there are 'Version Properties' and 'Version Description' fields, and an 'Incremental Version' checkbox. A 'Deploy' button is present. At the bottom, there are 'Save' and 'Apply' buttons. The 'Post-build Actions' section is empty.

Jenkins Pipeline Project

Required Plugins

- 1) Pipeline Maven Integration Plugin
- 2) JUnit Attachments Plugin'
- 3) Task Scanner Plugin

In Jenkins Pipeline project, we will use one file called Jenkinsfile, in this file we will write groovy code to build process.

We will write Jenkinsfile in 2 ways.

- 1) Declarative way
- 2) Scripted way.

1) Declarative Pipeline Syntax

Jenkinsfile

```
pipeline {
    agent any

    stages {
        stage('Compile Stage') {
            steps {
                echo 'Compile Stage starts...'
                withMaven(maven : 'maven_3_5_2'){
                    sh 'mvn clean compile'
                    echo 'Compile Stage ends...'
                }
            }
        }
        stage('Testing Stage') {
            steps {
                echo 'Testing Stage starts...'
                withMaven(maven : 'maven_3_5_2'){
                    sh 'mvn test'
                    echo 'Testing Stage ends...'
                }
            }
        }
        stage('Install Stage') {
            steps {
                echo 'Install Stage starts...'
                withMaven(maven : 'maven_3_5_2'){
                    sh 'mvn install'
                    echo 'Install Stage ends...'
                }
            }
        }
        stage('Deploy Stage') {
            steps {
                echo 'Install Stage starts...'
                withMaven(maven : 'maven_3_5_2'){
                    sh 'mvn deploy'
                    echo 'Deploy Stage ends...'
                }
            }
        }
        stage('SonarQube Scanner Stage') {
            steps {
                echo 'Deployment Stage starts...'
                sh 'mvn sonar:sonar'
                echo 'Deployment Stage ends...'
            }
        }
    }
}
```

}

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-
v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.bt</groupId>
  <artifactId>maven-web-project</artifactId>
  <packaging>war</packaging>
  <version>0.0.1-SNAPSHOT</version>
  <name>maven-web-project Maven Webapp</name>
  <url>http://maven.apache.org</url>

  <properties>
    <!-- Spring test requires 1.7 onwards. don't down grade version -->
    <!-- <jdk.version>1.6</jdk.version> -->
    <spring.version>4.1.4.RELEASE</spring.version>
    <junit.version>4.11</junit.version>
    <log4j.version>1.2.17</log4j.version>
  </properties>

  <dependencies>

    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>

    <dependency>
      <groupId>org.json</groupId>
      <artifactId>json</artifactId>
      <version>20160212</version>
    </dependency>

    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-core</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-web</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-webmvc</artifactId>
      <version>${spring.version}</version>
    </dependency>
```

```
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>${spring.version}</version>
</dependency>
<dependency>
    <groupId>javax</groupId>
    <artifactId>javaee-web-api</artifactId>
    <version>7.0</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>3.1.0</version>
</dependency>
<!-- http://mvnrepository.com/artifact/org.springframework/spring-jdbc --&gt;
&lt;dependency&gt;
    &lt;groupId&gt;org.springframework&lt;/groupId&gt;
    &lt;artifactId&gt;spring-jdbc&lt;/artifactId&gt;
    &lt;version&gt;${spring.version}&lt;/version&gt;
&lt;/dependency&gt;

&lt;/dependencies&gt;

&lt;repositories&gt;
    &lt;repository&gt;
        &lt;id&gt;myMavenRepo.read&lt;/id&gt;
        &lt;url&gt;https://mymavenrepo.com/repo/Kc9khPUGgDzV9gruoyls/&lt;/url&gt;
    &lt;/repository&gt;
&lt;/repositories&gt;

&lt;distributionManagement&gt;
    &lt;repository&gt;
        &lt;id&gt;myMavenRepo.write&lt;/id&gt;
        &lt;url&gt;https://mymavenrepo.com/repo/xtzDXsryp1OKYd3hZ4iE/&lt;/url&gt;
    &lt;/repository&gt;
&lt;/distributionManagement&gt;

&lt;build&gt;
    &lt;plugins&gt;
        &lt;plugin&gt;
            &lt;groupId&gt;org.sonarsource.scanner.maven&lt;/groupId&gt;
            &lt;artifactId&gt;sonar-maven-plugin&lt;/artifactId&gt;
            &lt;!--&lt;version&gt;3.3.0.603&lt;/version&gt;--&gt;
        &lt;/plugin&gt;
    &lt;/plugins&gt;
&lt;/build&gt;

&lt;profiles&gt;</pre>
```

```
<profile>
    <id>sonar</id>
    <activation>
        <activeByDefault>true</activeByDefault>
    </activation>
    <properties>
        <sonar.host.url>http://localhost:9001/</sonar.host.url>
    </properties>
</profile>
</profiles>
</project>
```

2) Scripted Pipeline Syntax

```
#!groovy

node {
    currentBuild.result = "SUCCESS"

    try {

        stage('Checkout'){
            checkout scm
        }

        stage('Compiling'){
            sh 'mvn deploy'
        }

        stage('Sonar') {
            //add stage sonar
            sh 'mvn sonar:sonar'
        }
        stage('mail'){

            mail body: 'project build successful',
            from: 'devopstrainingblr@gmail.com',
            replyTo: 'mithunreddytechnologies@gmail.com',
            subject: 'project build successful',
            to: 'mithunreddytechnologies@gmail.com'
        }
    }

    catch (err) {

        currentBuild.result = "FAILURE"

        mail body: "project build error is here: ${env.BUILD_URL}" ,
        from: 'devopstrainingblr@gmail.com',
        replyTo: 'mithunreddytechnologies@gmail.com',
    }
}
```

subject: 'project build failed',
to: 'mithunreddytechnologies@gmail.com'

```
    throw err
}
}
```

Jenkins Multi Branch Pipeline Project

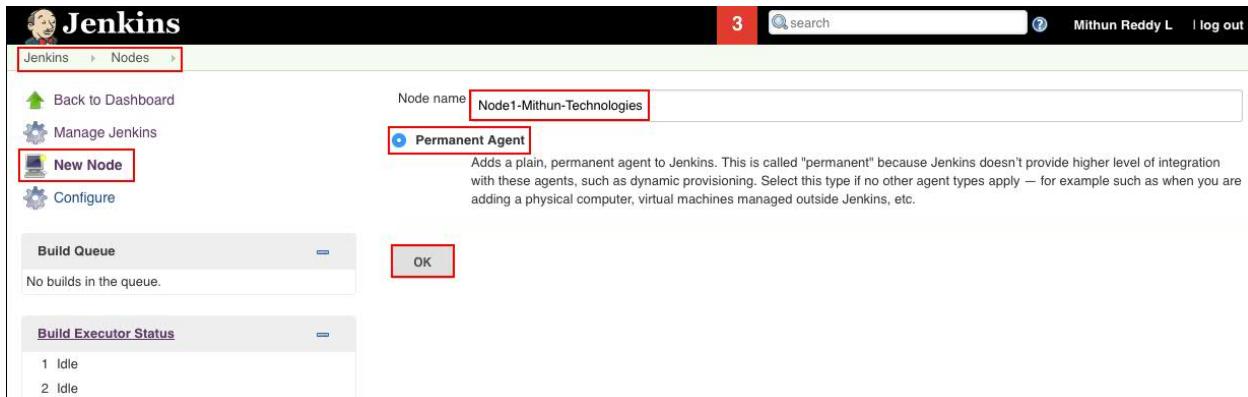
Required Plugins

- 1) Pipeline: Multibranch
-

Blue Ocean

Jenkins Master-Slave setup

Manage Jenkins ---> Manage Nodes ---> New Node
Provide the Node name and click on **OK** button.



The screenshot shows the Jenkins 'New Node' configuration page. The 'Node name' field contains 'Node1-Mithun-Technologies'. The 'Permanent Agent' radio button is selected. The 'OK' button is highlighted with a red box. Other fields include 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle).

Provide all the details as follows and click on **Save** button.

Name	Node1-Mithun-Techologies
Description	This Node is used to build for Java Projects.
# of executors	1
Remote root directory	/Users/bhaskarreddyl/BhaskarReddyL/Softwares/Running/jenkins/node1/
Labels	Node1-Mithun-Techologies
Usage	Use this node as much as possible
Launch method	Launch agent via Java Web Start
Disable WorkDir	<input type="checkbox"/>
Custom WorkDir path	/Users/bhaskarreddyl/BhaskarReddyL/Softwares/Running/jenkins/node1/workdirectory
If defined, a custom Remoting work directory will be used instead of the Agent Root Directory. This option has no environment variable resolution so far, it is recommended to use only absolute paths.	
Internal data directory	remoting
Fail if workspace is missing	<input type="checkbox"/>
Advanced...	
Availability	Keep this agent online as much as possible

Node Properties

Enable node-based security
 Environment variables
 Tool Locations

Save

Note: Suppose if you don't see "Launch agent via Java Web Start" option, do the below configurations.

Manage Jenkins ---> Configure Global Security ---> enable the TCP port for JNLP agents (by default, it is Disabled.)

Agents

TCP port for JNLP agents	<input checked="" type="radio"/> Fixed : 50000	<input type="radio"/> Random	<input type="radio"/> Disable
Agent protocols...			

Once you click on Save you will see the Nodes and Master detail, and select the Node which we have created and click on configure.

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	master	Mac OS X (x86_64)	In sync	144.97 GB	1.36 GB	144.97 GB	0ms
	Node1-Mithun-Technologies		N/A	N/A	N/A	N/A	N/A
			ms	6 ms	4 ms	16 min	1 ms
							0 ms

You will see below screen and click download the slave.jar file.

Agent Node1-Mithun-Technologies (This Node is used to build for Java Projects.)

Mark this node temporarily offline

Connect agent to Jenkins one of these ways:

- Launch agent from browser
- Run from agent command line:

```
java -jar slave.jar -jnlpUrl http://localhost:8080/computer/Node1-Mithun-Technologies/slave-agent.jnlp -secret 8e6c24c3e977342073d2184d051b1fb87f30d57acd0c63ae0a913008e65ad86f -workDir "/Users/bhaskarreddyl/BhaskarReddyL/Softwares/Running/jenkins/node1/workdirectory"
```

Projects tied to Node1-Mithun-Technologies

None

Copy slave.jar file into any directory
(/Users/bhaskarreddyl/BhaskarReddyL/Softwares/Running/jenkins/node1)

Go to the path where slave.jar copied and run the below command.

```
java -jar slave.jar -jnlpUrl http://localhost:8080/computer/Node1-Mithun-Technologies/slave-agent.jnlp -secret 8e6c24c3e977342073d2184d051b1fb87f30d57acd0c63ae0a913008e65ad86f -workDir "/Users/bhaskarreddyl/BhaskarReddyL/Softwares/Running/jenkins/node1/workdirectory"
```

```
Bhaskars-MacBook-Air:node1 bhaskarreddyl$ java -jar slave.jar -jnlpUrl http://localhost:8080/computer/  
Node1-Mithun-Technologies/slave-agent.jnlp -secret 8e6c24c3e977342073d2184d051b1fb87f30d57acd0c63ae0a9  
13008e65ad86f -workDir "/Users/bhaskarreddyl/BhaskarReddyL/Softwares/Running/jenkins/node1/workdirectory"  
Nov 26, 2017 9:48:30 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir  
INFO: Using /Users/bhaskarreddyl/BhaskarReddyL/Softwares/Running/jenkins/node1/workdirectory/remoting  
as a remoting work directory  
Both error and output logs will be printed to /Users/bhaskarreddyl/BhaskarReddyL/Softwares/Running/jenkins/node1/workdirectory/remoting  
Nov 26, 2017 9:48:31 PM hudson.remoting.jnlp.Main createEngine  
INFO: Setting up slave: Node1-Mithun-Technologies  
Nov 26, 2017 9:48:31 PM hudson.remoting.jnlp.Main$CuiListener <init>  
INFO: Jenkins agent is running in headless mode.  
Nov 26, 2017 9:48:31 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir  
INFO: Using /Users/bhaskarreddyl/BhaskarReddyL/Softwares/Running/jenkins/node1/workdirectory/remoting  
as a remoting work directory  
Nov 26, 2017 9:48:31 PM hudson.remoting.jnlp.Main$CuiListener status  
INFO: Locating server among [http://localhost:8080]  
Nov 26, 2017 9:48:31 PM org.jenkinsci.remoting.engine.JnlpAgentEndpointResolver resolve  
INFO: Remoting server accepts the following protocols: [JNLP4-connect, JNLP-connect, Ping, JNLP2-conne  
ct]  
Nov 26, 2017 9:48:31 PM hudson.remoting.jnlp.Main$CuiListener status  
INFO: Agent discovery successful  
Agent address: localhost  
Agent port: 50000  
Identity: 96:6e:10:60:c1:c4:f2:e8:7e:4c:d9:c7:01:b3:e1:a3  
Nov 26, 2017 9:48:31 PM hudson.remoting.jnlp.Main$CuiListener status  
INFO: Handshaking  
Nov 26, 2017 9:48:31 PM hudson.remoting.jnlp.Main$CuiListener status  
INFO: Connecting to localhost:50000  
Nov 26, 2017 9:48:31 PM hudson.remoting.jnlp.Main$CuiListener status  
INFO: Trying protocol: JNLP4-connect  
Nov 26, 2017 9:48:31 PM hudson.remoting.jnlp.Main$CuiListener status  
INFO: Remote identity confirmed: 96:6e:10:60:c1:c4:f2:e8:7e:4c:d9:c7:01:b3:e1:a3  
Nov 26, 2017 9:48:32 PM hudson.remoting.jnlp.Main$CuiListener status  
INFO: Connected
```

Now slave become communicating to node and it is live.

Now you can use this slave for job creation.

Create one Freestyle project/any kind of project and select the Restrict where this project can be run and select the Node which you have created.

The screenshot shows the Jenkins 'General' configuration page for a new project. The 'Restrict where this project can be run' checkbox is checked and highlighted with a red box. The 'Label Expression' field contains 'Node1-Mithun-Technologies' and displays a warning message: 'There's no agent/cloud that matches this assignment. Did you mean 'master' instead of 'Node'?'. The 'Advanced...' button is visible at the bottom right.

Provide the Git url and click on **Save** button.

Resources:

<https://jenkins.io/> ---> Download software

<https://wiki.jenkins-ci.org/display/JENKINS/Installing+Jenkins+as+a+Windows+service>

<http://www.tothenew.com/blog/jenkins-implementing-project-based-matrix-authorization-strategy/> --->
User Access

<https://support.cloudbees.com/hc/en-us/articles/216118748-How-to-Start-Stop-or-Restart-your-Instance>

Mithun Reddy Lacchannagari
mithunreddytechnologies@gmail.com

Jenkins