

Jenkins

Introduction

CI
CD
CD

Installation

- a) Generic Way
- b) Install Installable software for respective Operating System
- c) Deploy war file into Tomcat server

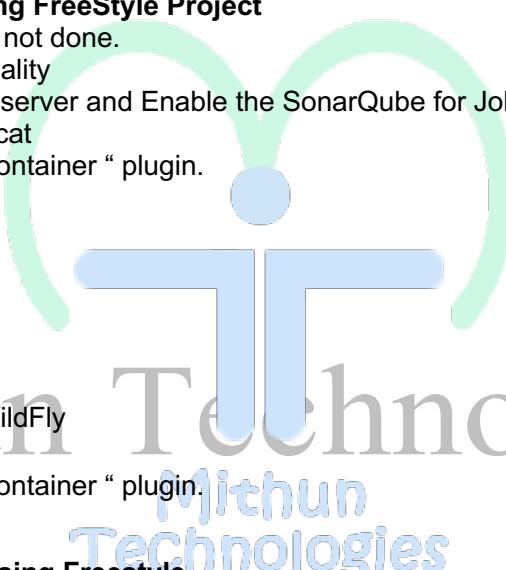
Jenkins Directory structure

Create the ANT Project using FreeStyle Project

- a) Integrate ANT software if not done.
- b) Configure EMail Functionality
- c) Integrate the SonarQube server and Enable the SonarQube for Jobs
- d) Deploy the App into Tomcat
 - 1) Through "Deploy to container" plugin.
 - 2) Through script
- e) Poll SCM
- f) Build Periodically
- g) Discard Old Build

Deploy the App into JBoss/WildFly

- a) Through script
- b) Through "Deploy to container" plugin.

The logo features a stylized blue 'T' shape with a green circular arrow above it, all set against a light gray background. The text 'Mithun Technologies' is written in a large, semi-transparent gray font across the center.

Create the Maven Project using Freestyle

Integrate Maven software if not done.
Integrate Nexus With Jenkins
Integrate SonarQube with Jenkins
Deploy the App into Tomcat

- 1) Through "Deploy to container" plugin.
- 2) Through script

Create the Maven Project using Maven Projects

Plugin Management

- a) Safe Restart Plugin : This plugin allows you to restart Jenkins safely.

Note: If you see below error while restarting

Jenkins cannot restart itself as currently configured.

Solution: Restart your command prompt in Administrator mode. So that you are having your all permission to run window as a service.

- b) Next Build Number Plugin : Sets the build number Jenkins will use for a job's next build.
- c) Email Extension Plugin :
- d) SonarQube Scanner for Jenkins :
- e) Schedule Build Plugin :
- f) Artifactory Plugin : This plugin allows deploying maven artifacts and build info to Artifactory.

- h) Cloud Foundry Plugin : Pushes a project to Cloud Foundry or a CF-based platform (e.g. IBM Cloud, Pivotal, Stackato) at the end of a build.
- i) Blue Ocean :
- k) Deploy to container : For deploying app into Tomcat server/Wildfly/Jboss.

Create Views

Jenkins Security

- a) Create Users (Default Admin)
- b) Provide the specific access Jenkins
- c) Provide the access to specific access to specific projects

Jenkins CLI

Create the PipeLine Project Jobs
Create the Mulibranch Pipeline Project Jobs

Create Master/Slave

Backup

Integrate the Urban Code Deploy server with Jenkins
Deploy the App into IBM Cloud

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.

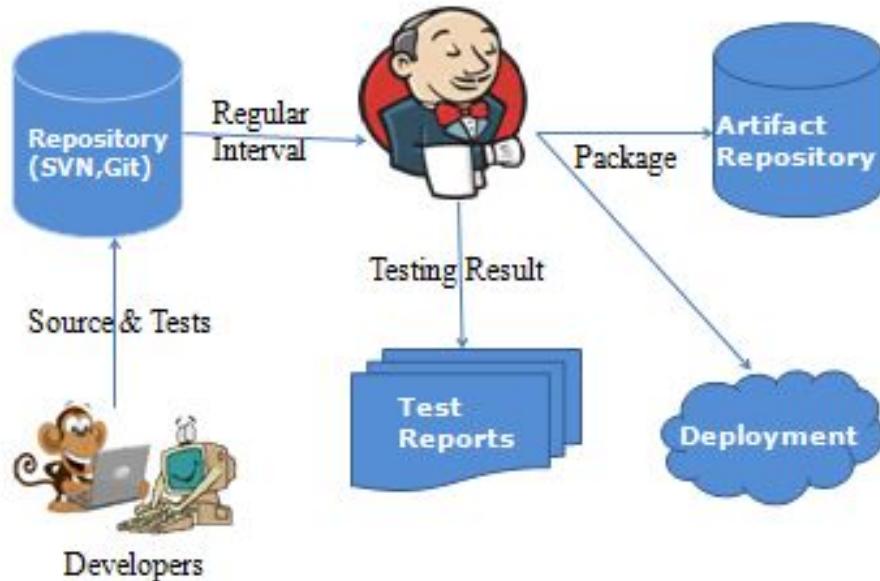
Continuous Integration: Continuous Integration (CI) is the process of automating the build and testing of code every time a team member commits changes to version control.

(OR)

Continuous Integration is a development practice where developers integrate their code into a shared remote repository frequently, preferably several times a day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible.

CI Flow

Below diagram CI flow with Jenkins as Build tool.



CI – Benefits

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

Technologies

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

Continuous Deployment: The practicing of automatically deploying every successful build directly into production without any manual steps known 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



Mithun Technologies

What Jenkins can do?

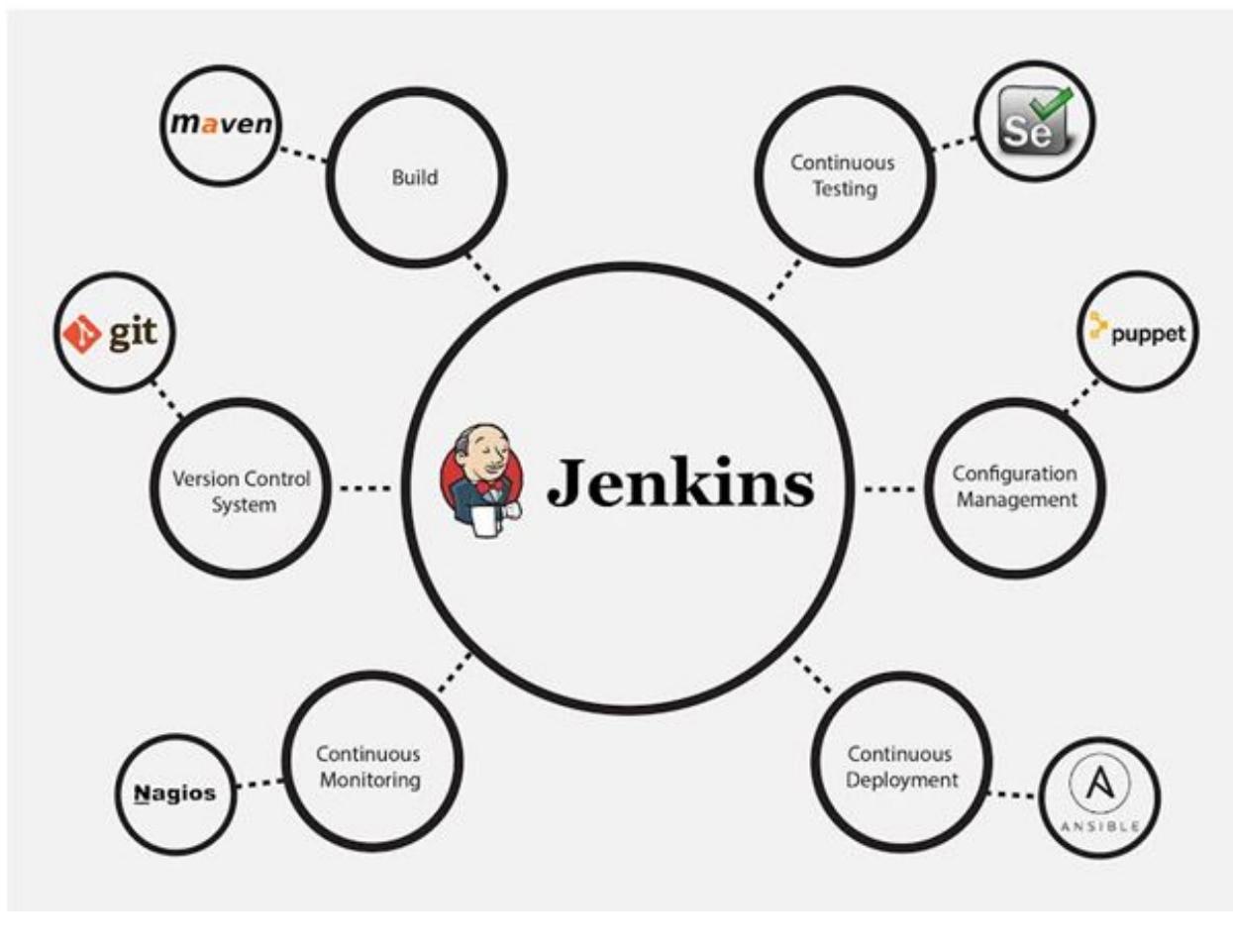
- Integrate with many different Version Control Systems (GitHub, CVS, SVN, TFS ...)
- Generate test reports (JUnit)
- Push the builds to various artifact repositories
- Deploys directly to production or test environments
- Notify stakeholders of build status (Through Email)

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 1000+ plugins to ease your work. If a plugin does not exist, just code it up and share with the community (<https://plugins.jenkins.io/>).
- ✓ 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 |

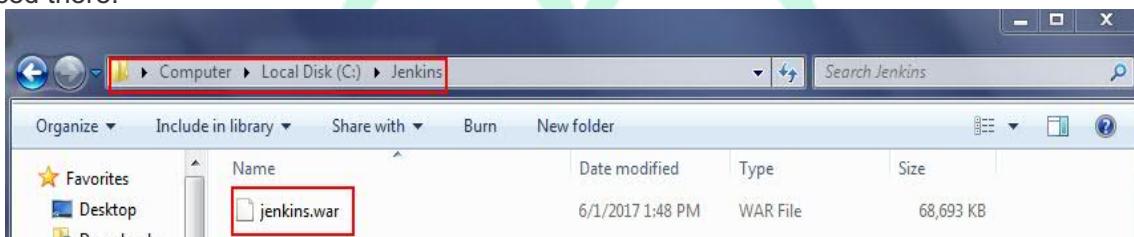
Jenkins Installation

- 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 open command prompt and go to the path, where jenkins.war is available and execute the below command.

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
```

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
Running from: C:\jenkins\jenkins.war
java -jar C:\jenkins\jenkins.war
13Jul 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
+10Jul 02, 2017 7:25:56 AM org.eclipse.jetty.util.log.JavaUtilLog info
INFO: Logging to Jenkins-InitReactorRunner$1 by GELF
Jul 02, 2017 7:25:56 AM winstone.Logger logInternal
INFO: Beginning extraction from war file
+13Jul 02, 2017 7:26:12 AM org.eclipse.jetty.util.log.JavaUtilLog warn
INFO: winstone.Logger logInternal
+10Jul 02, 2017 7:26:18 AM org.eclipse.jetty.util.log.JavaUtilLog info
INFO: Jetty-3.2.z-SNAPSHOT
Jul 02, 2017 7:26:20 AM org.eclipse.jetty.util.log.JavaUtilLog info
INFO: NO JSP Support
Jul 02, 2017 7:26:20 AM org.eclipse.jetty.util.log.JavaUtilLog 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 w/7a8f4633c/file:/C:/Users/IBM_ADMIN/.jenkins/war/[AVAILABLE]<C:/Users/IBM_ADMIN/.jenkins.war>
Jul 02, 2017 7:26:20 AM org.eclipse.jetty.util.log.JavaUtilLog info
INFO: Started ServerConnector@62435e[HTTP/1.1@0.0.0:8080]
Jul 02, 2017 7:26:29 AM org.eclipse.jetty.util.log.JavaUtilLog info
INFO: Started @34325ms
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 7:26:44 AM Jenkins.InitReactorRunner$1 onAttained
INFO: Started all plugins
Jul 02, 2017 7:26:44 AM Jenkins.InitReactorRunner$1 onAttained
INFO: Registered all extensions
Jul 02, 2017 7:26:45 AM Jenkins.InitReactorRunner$1 onAttained
INFO: Loaded all jobs
Jul 02, 2017 7:27:02 AM hudson.model.AsyncPeriodicWork$1 run
INFO: Started download 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@2610795800: display name [Root WebApplicationContext]; startup date [Sun Jul 02 07:27:05 IST 2017]; root of factory hierarchy
Jul 02, 2017 7:27:06 AM org.springframework.beans.factory.support.DefaultListableBeanFactory preInstantiateSingletons
INFO: Bean factory for application context org.springframework.web.context.support.StaticWebApplicationContext@3030795800: org.springframework.beans.factory.support.DefaultListableBeanFactory
Jul 02, 2017 7:27:06 AM org.springframework.beans.factory.support.DefaultListableBeanFactory preInstantiateSingletons
INFO: Pre-instantiating singletons in org.springframework.context.support.StaticWebApplicationContext@3030795800: defining beans [authenticationManager]; root of factory hierarchy
Jul 02, 2017 7:27:06 AM org.springframework.context.support.StaticWebApplicationContext@3030795800: 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.AbstractApplicationContext@4665f9cf: 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 singleton in org.springframework.beans.factory.support.DefaultListableBeanFactory@4a472bf: 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
*****
Jenkins is fully up and running
Jul 02, 2017 7:27:23 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 updated 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 update center 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 ms
```

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

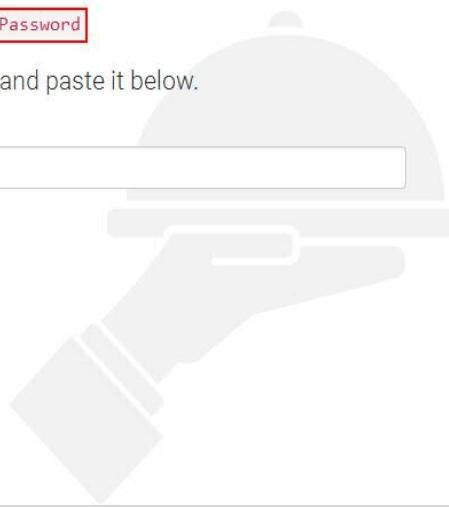
Unlock Jenkins

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

Mithun
Technologies

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

Mithun
Technologies

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

Mithun
Technologies

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

Mithun
Technologies

Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

[Start using Jenkins](#)

Jenkins 2.46.3

The screenshot shows the Jenkins dashboard at localhost:8080. The title bar says "Dashboard [Jenkins]". The main content area has a blue header "Welcome to Jenkins!" with the sub-instruction "Please [create new jobs](#) to get started." Below this, there are two sections: "Build Queue" (which is empty) and "Build Executor Status" (which shows 1 Idle and 2 Idle executors). The bottom right corner of the dashboard footer includes the text "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>.

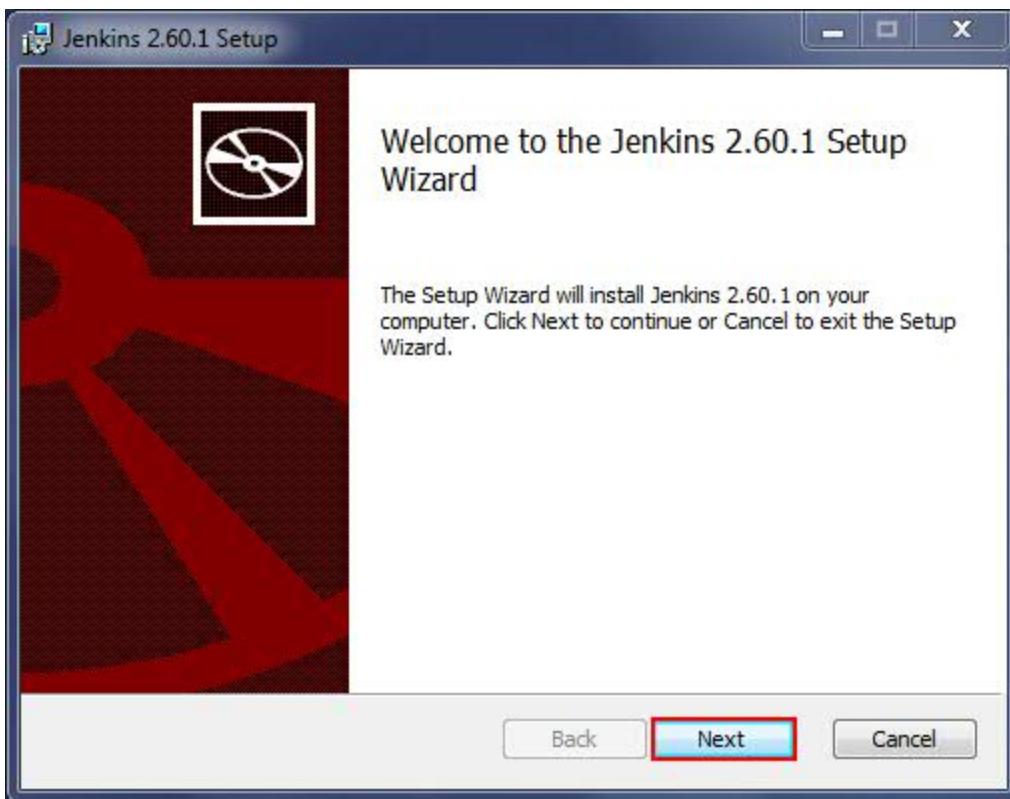
Method 3: Windows

Click on below icon.

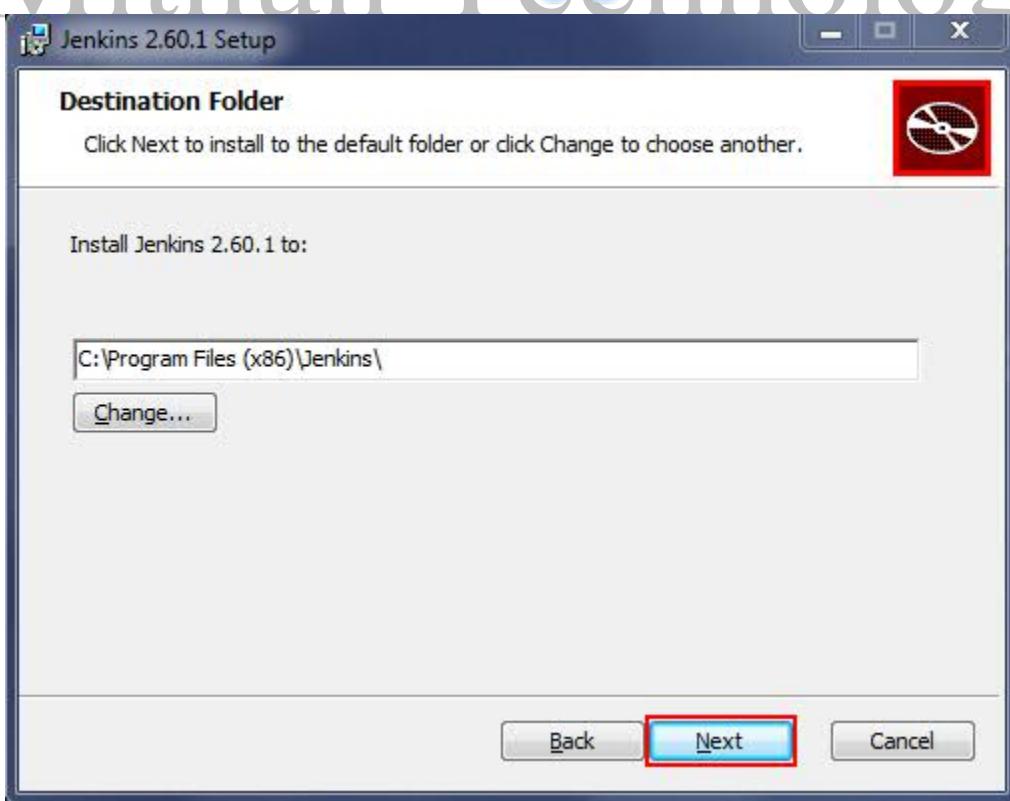


You will get the below screen





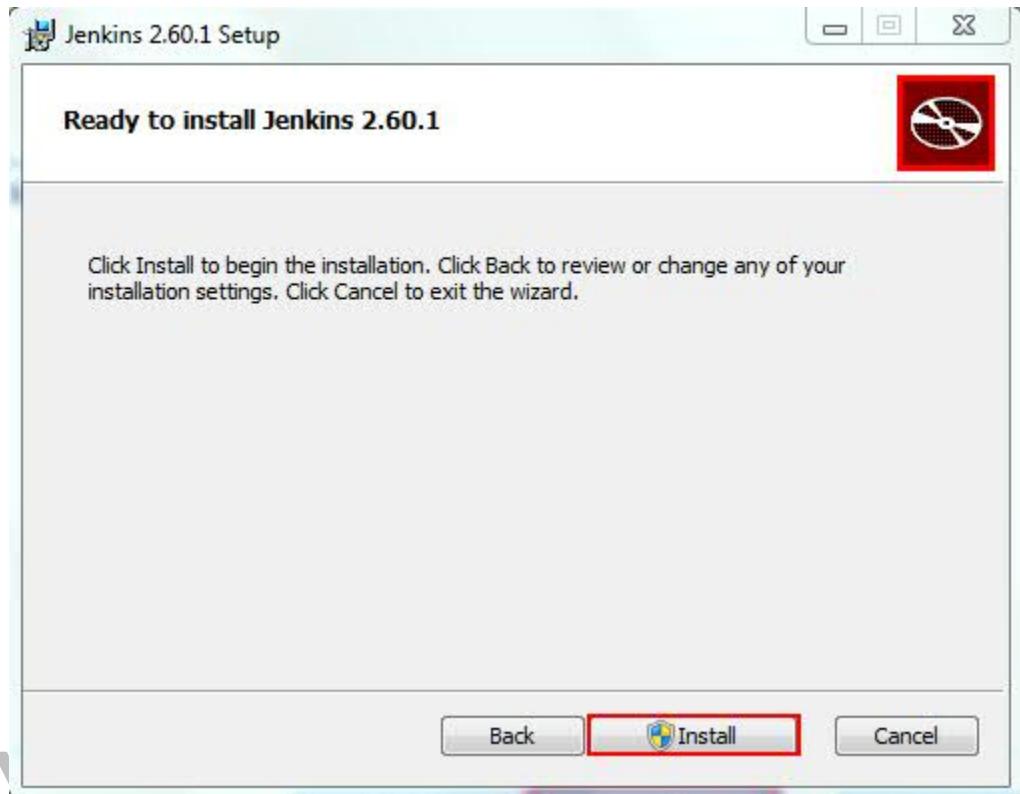
Click on Next



Mithun Reddy Lacchannagari
+919980923226

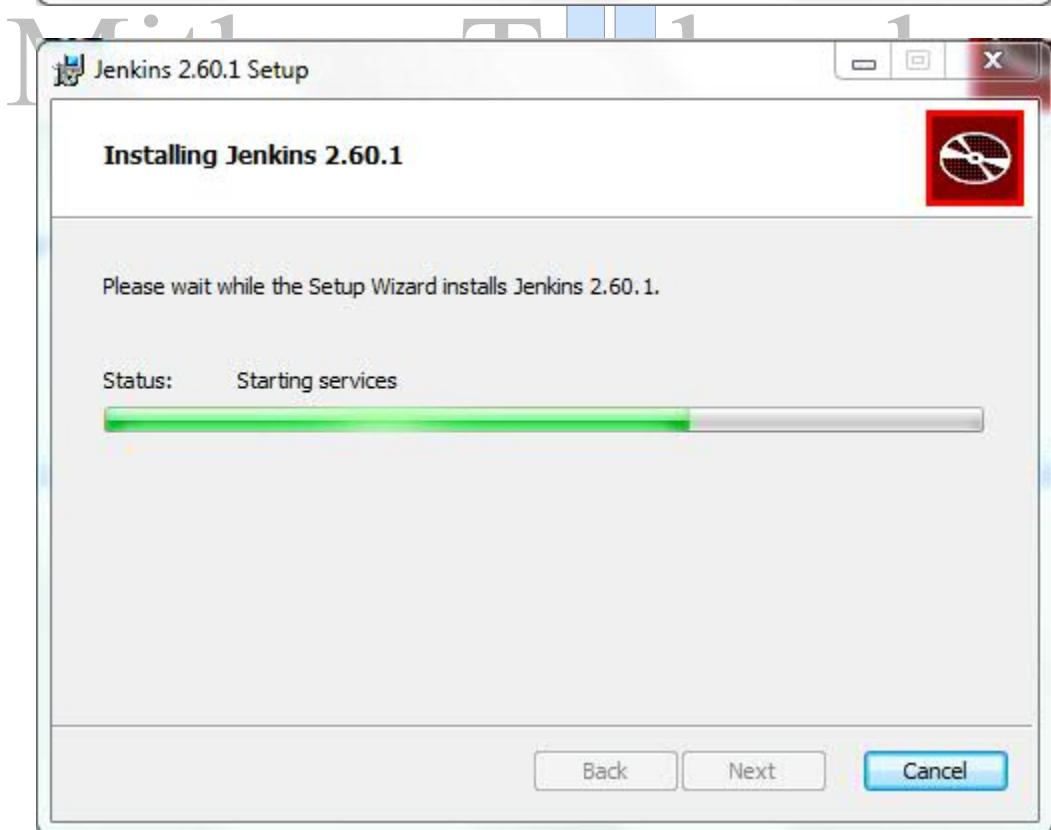
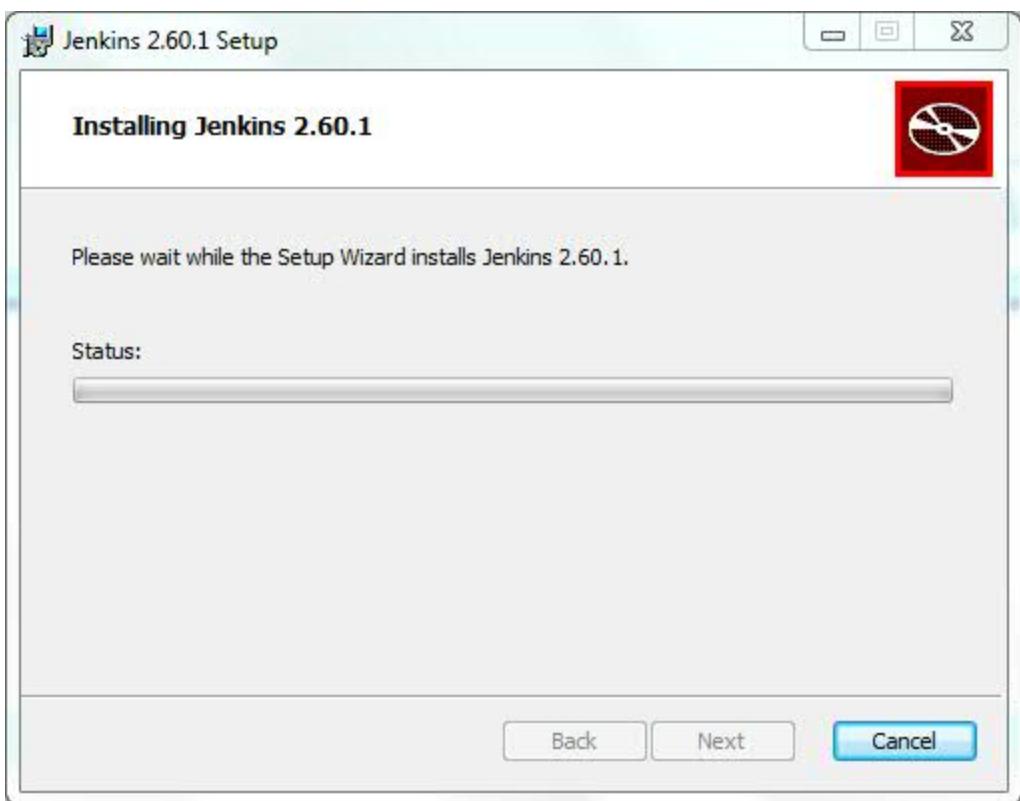
Jenkins

Click on Next

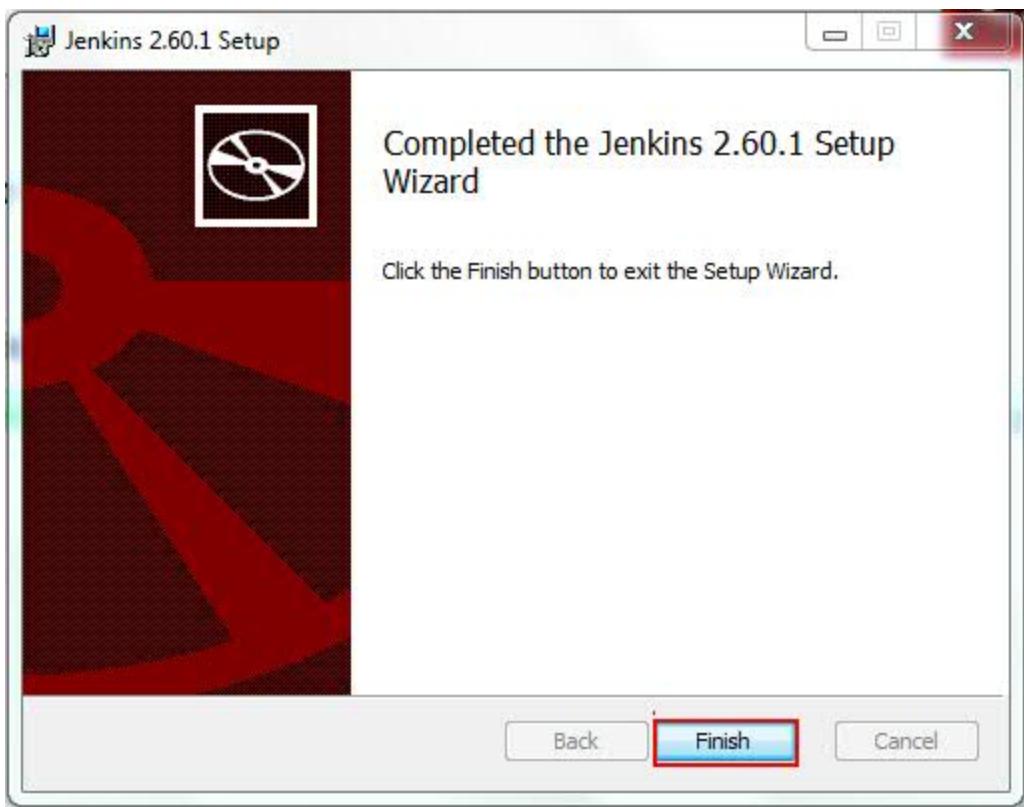


Click on Install

Mithun
Technologies



Click on Finish

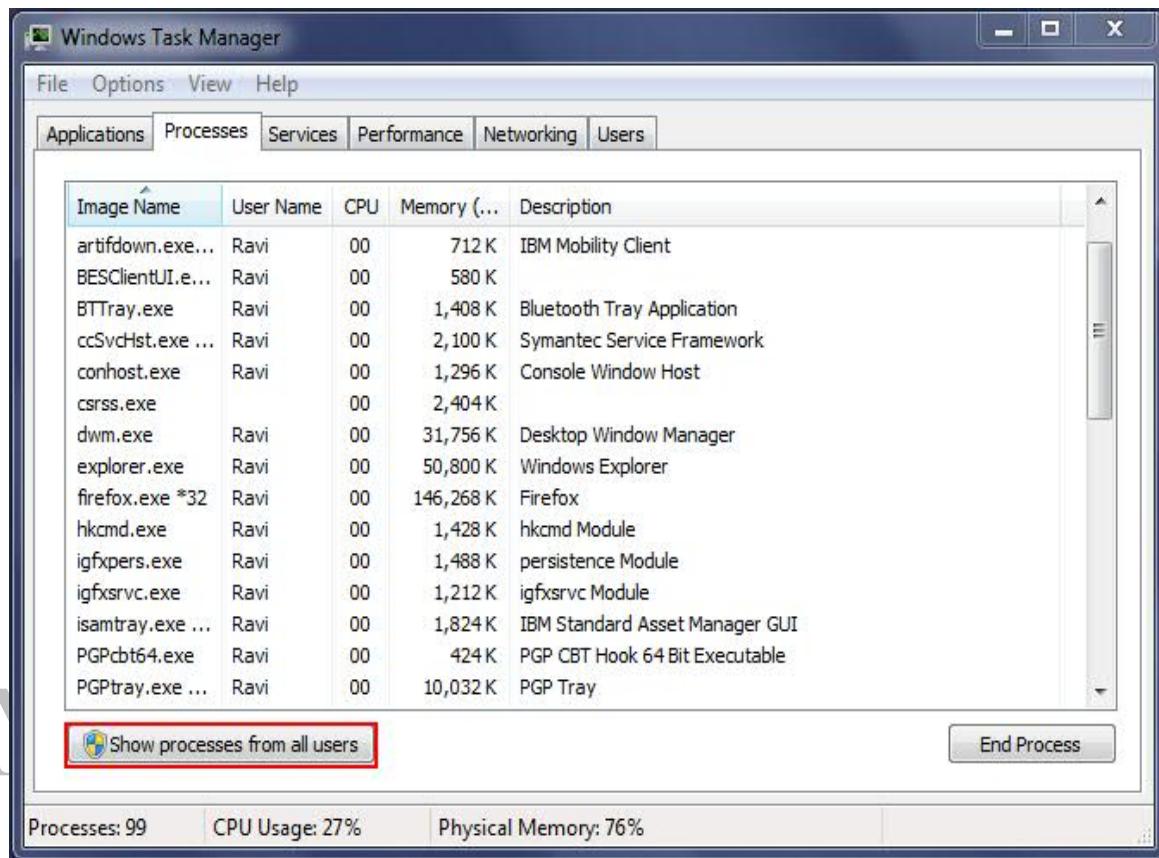


Once it successfully installed it will open by default in browser. Otherwise open the below url in browser.

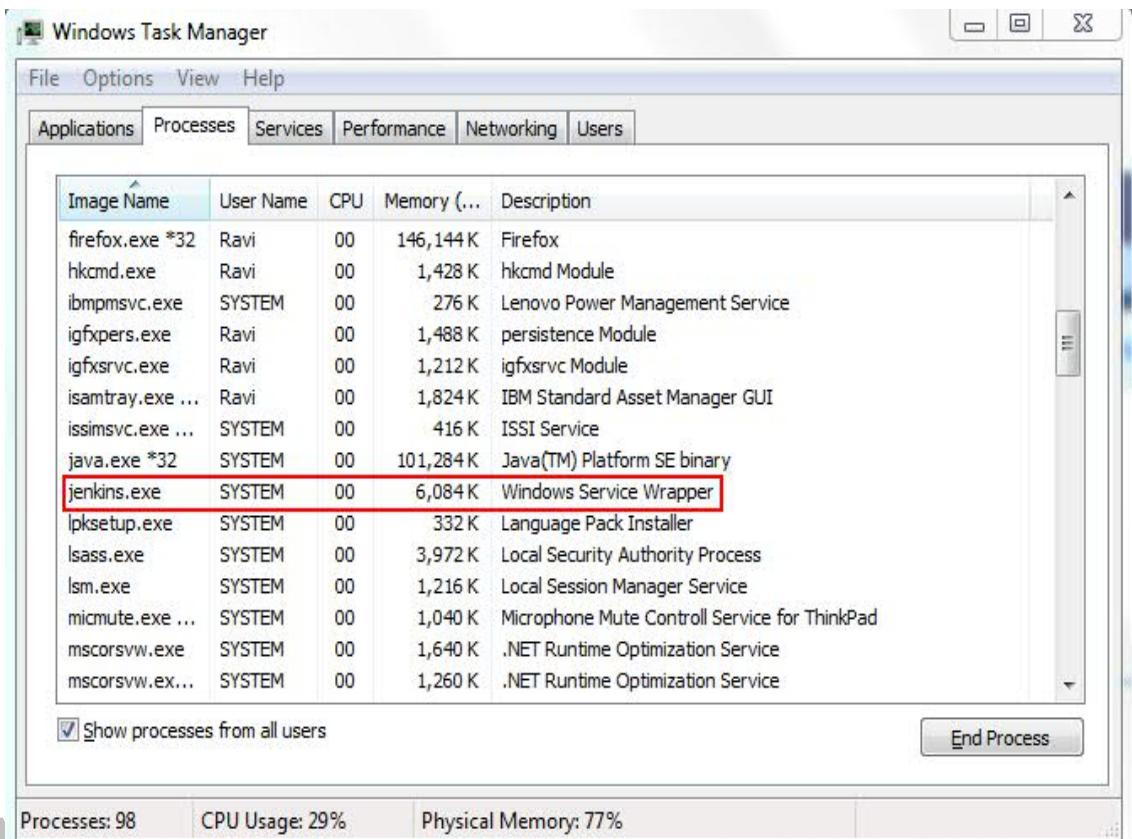
<http://localhost:8080>

Mithun
Technologies

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

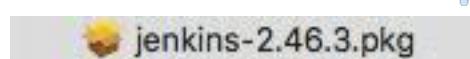


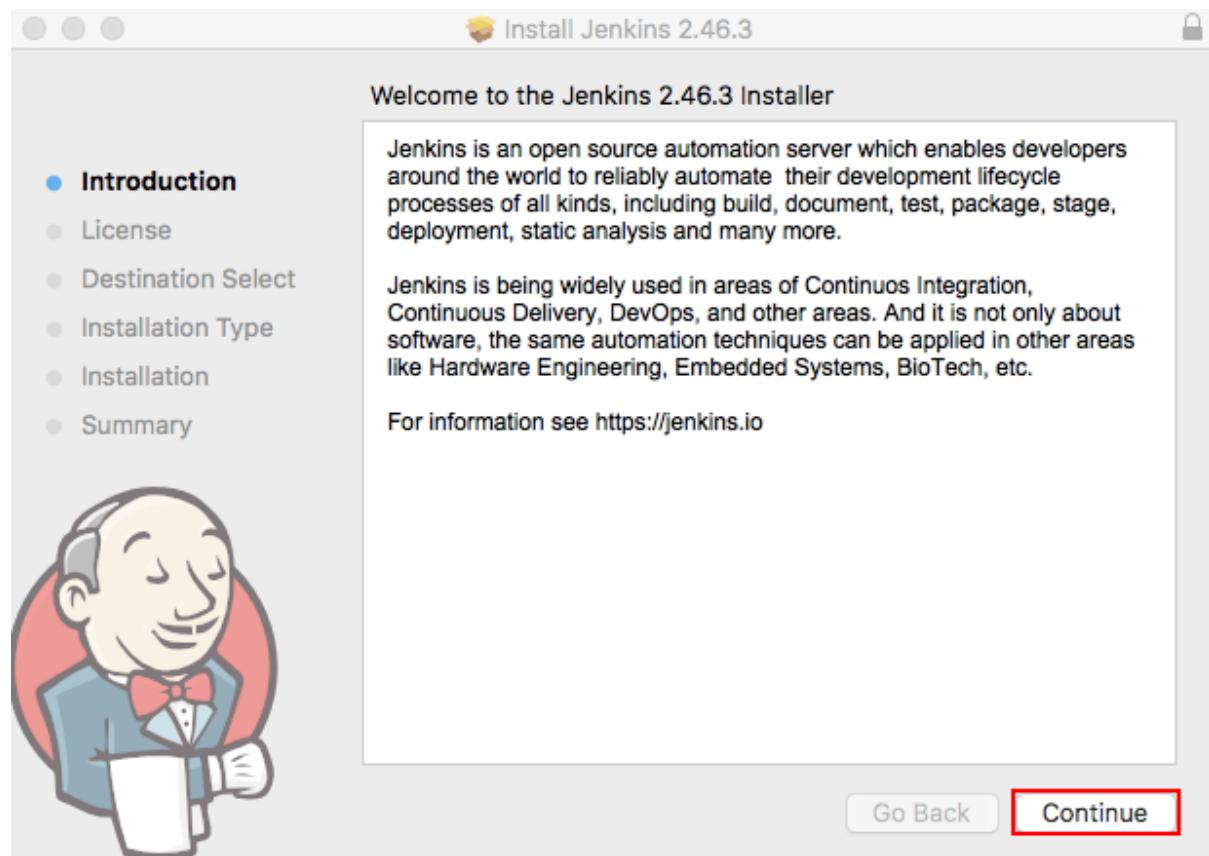
Note: Sometimes you will not see, so click on Show process from all users.



Method 3 MAC:

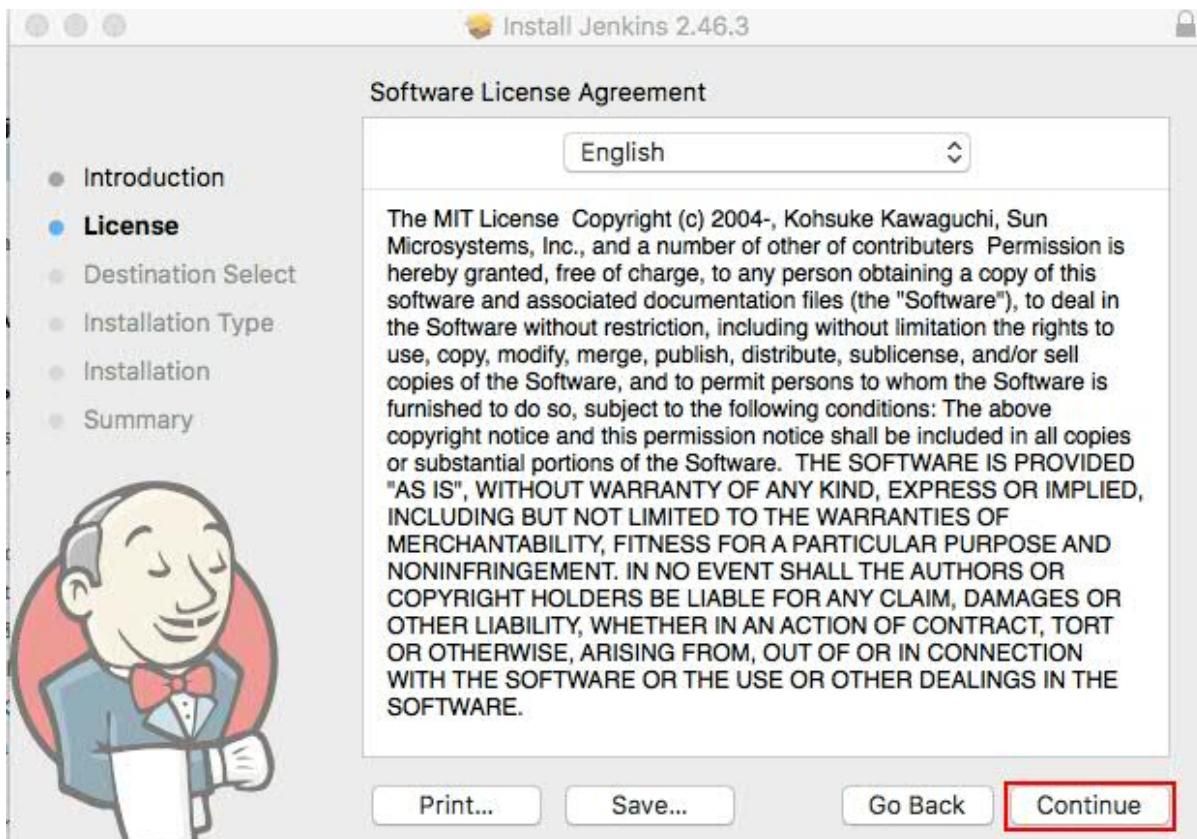
Click on below Jenkins package.





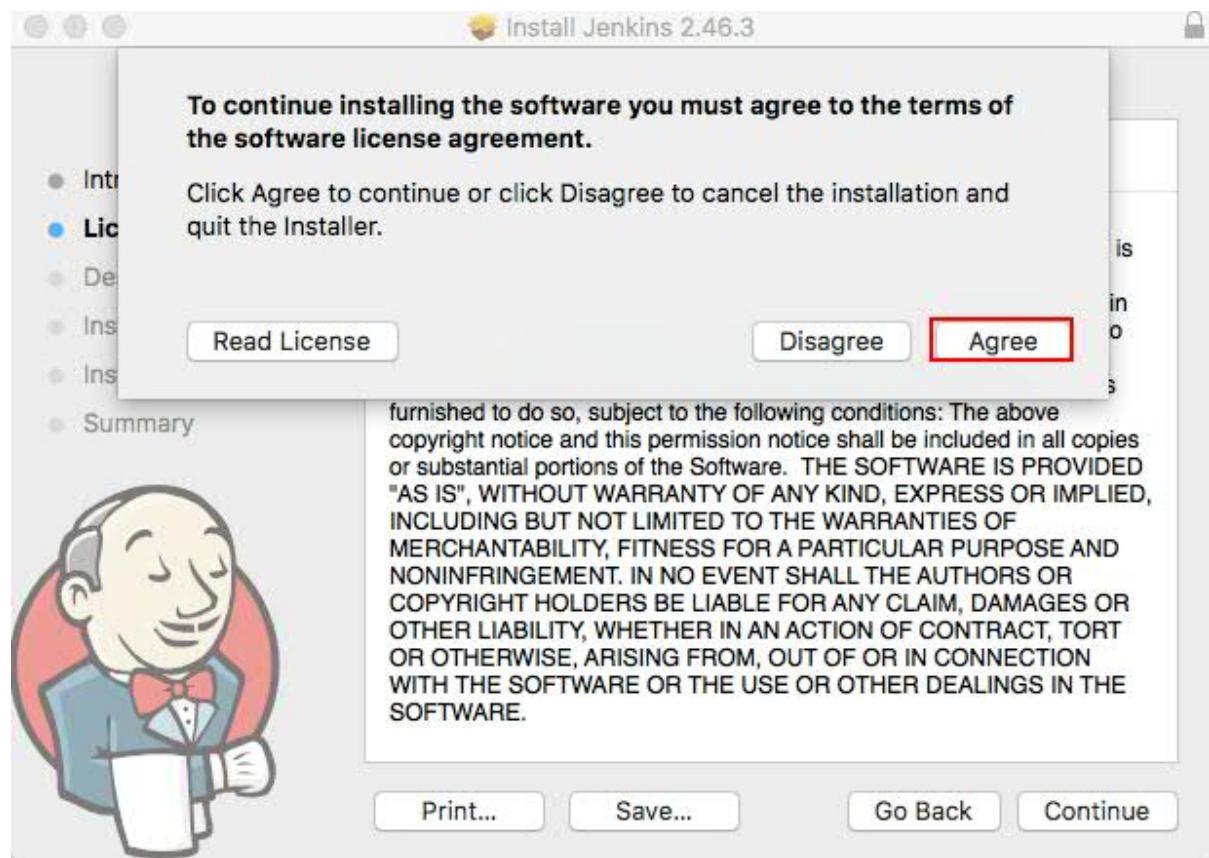
Click on Continue

Mithun Technologies
Mithun
Technologies



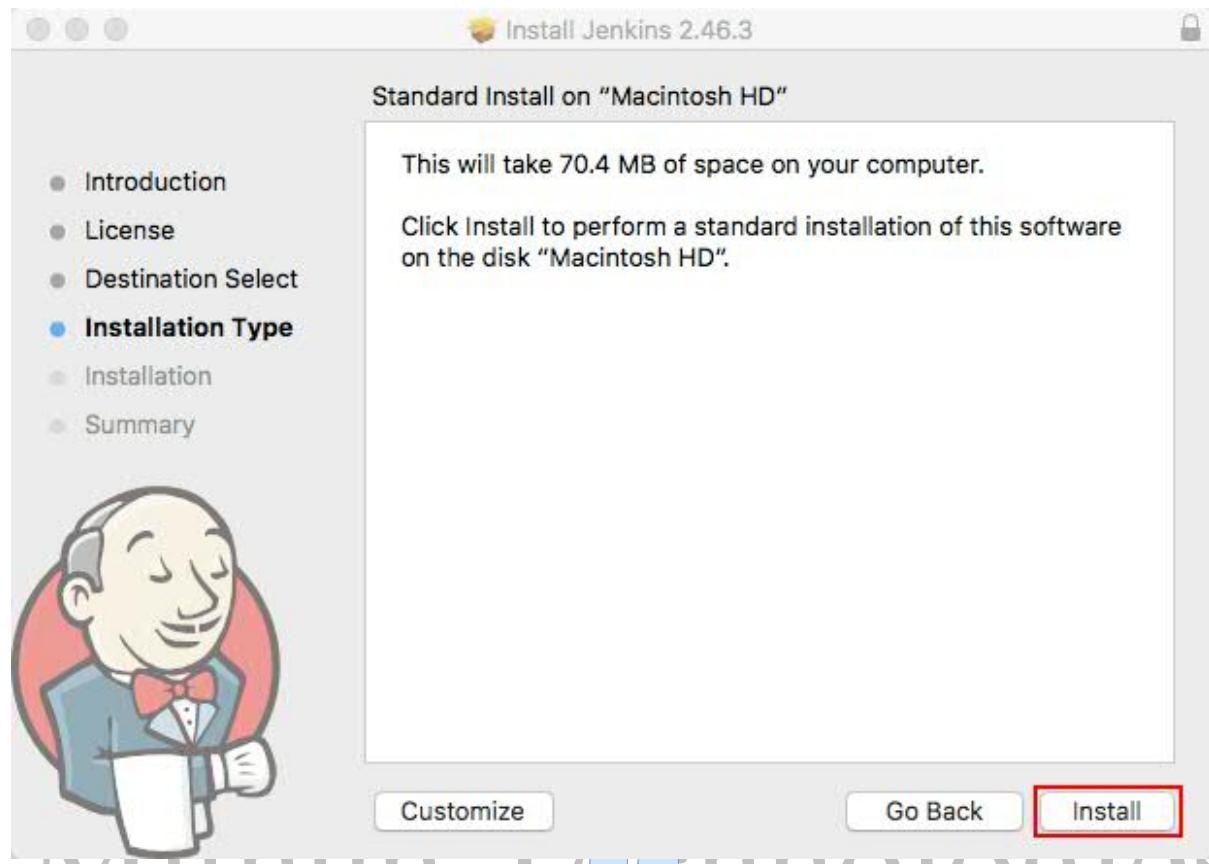
Mithun Technologies
Click on Continue

**Mithun
Technologies**

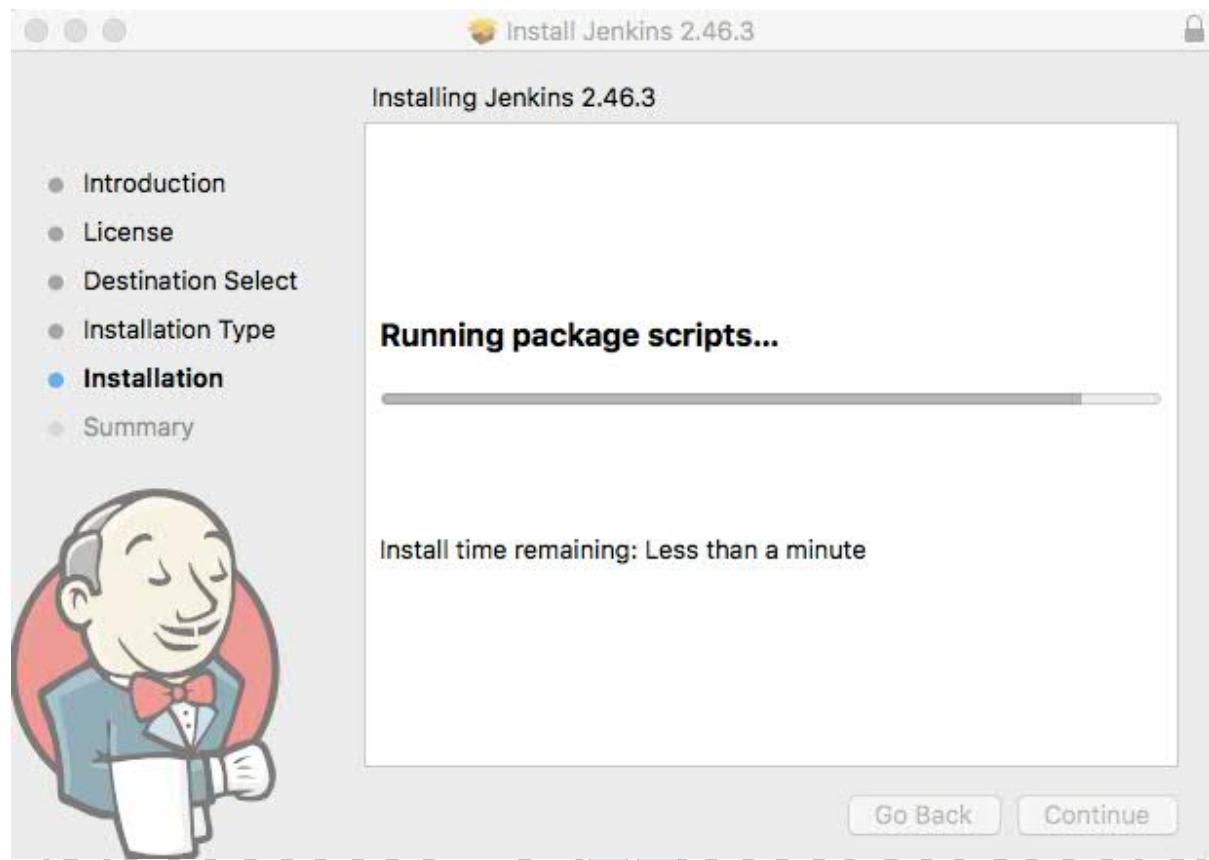


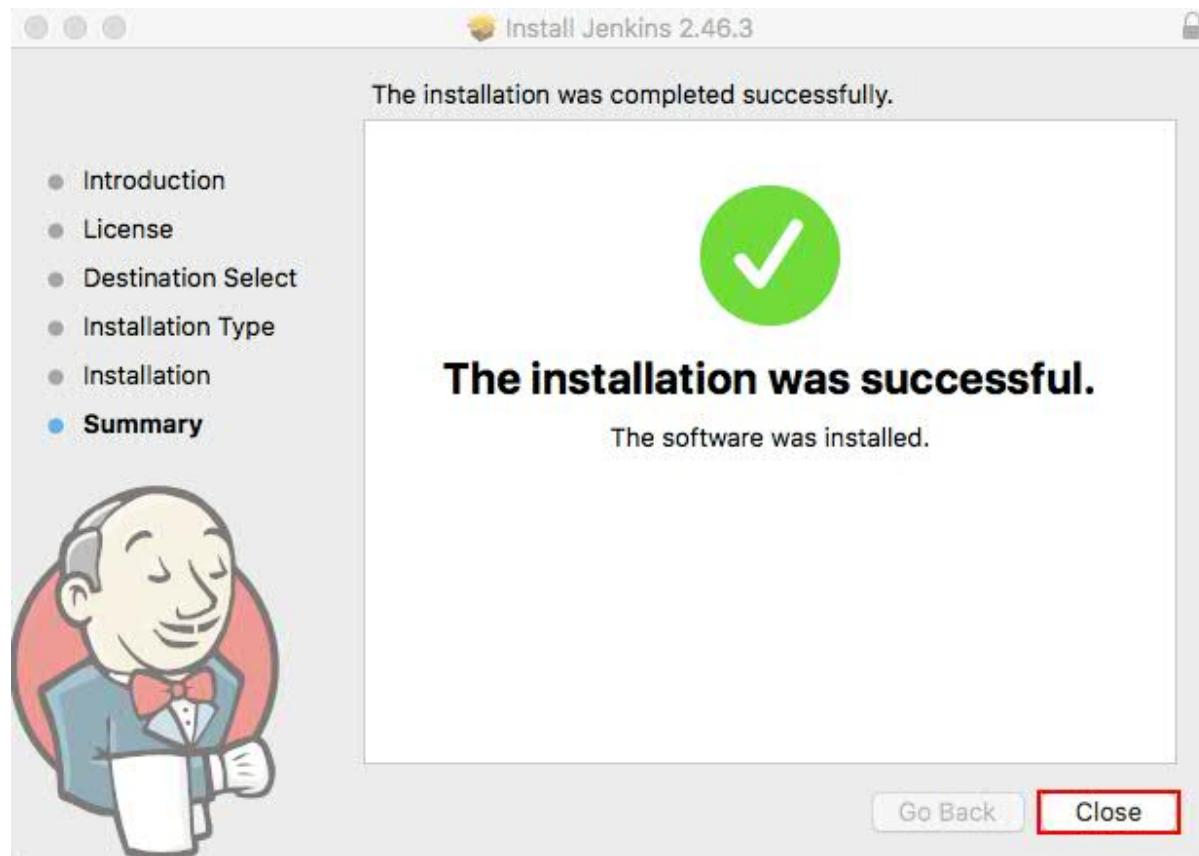
Click on Agree

Mithun
Technologies



Click on Install.
Mithun Technologies





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
```

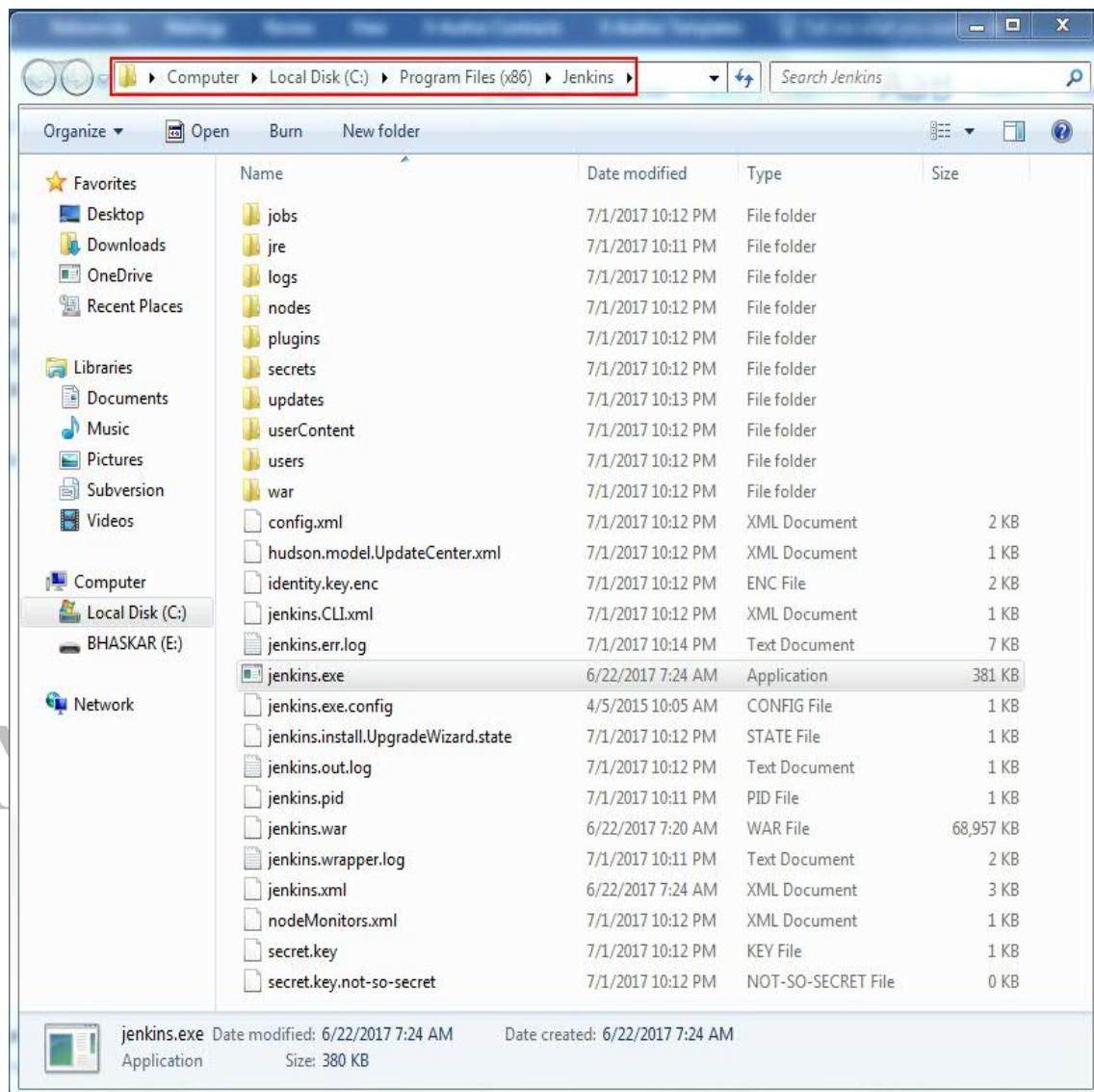
Now we can start the Jenkins Service by using systemctl as shown below

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

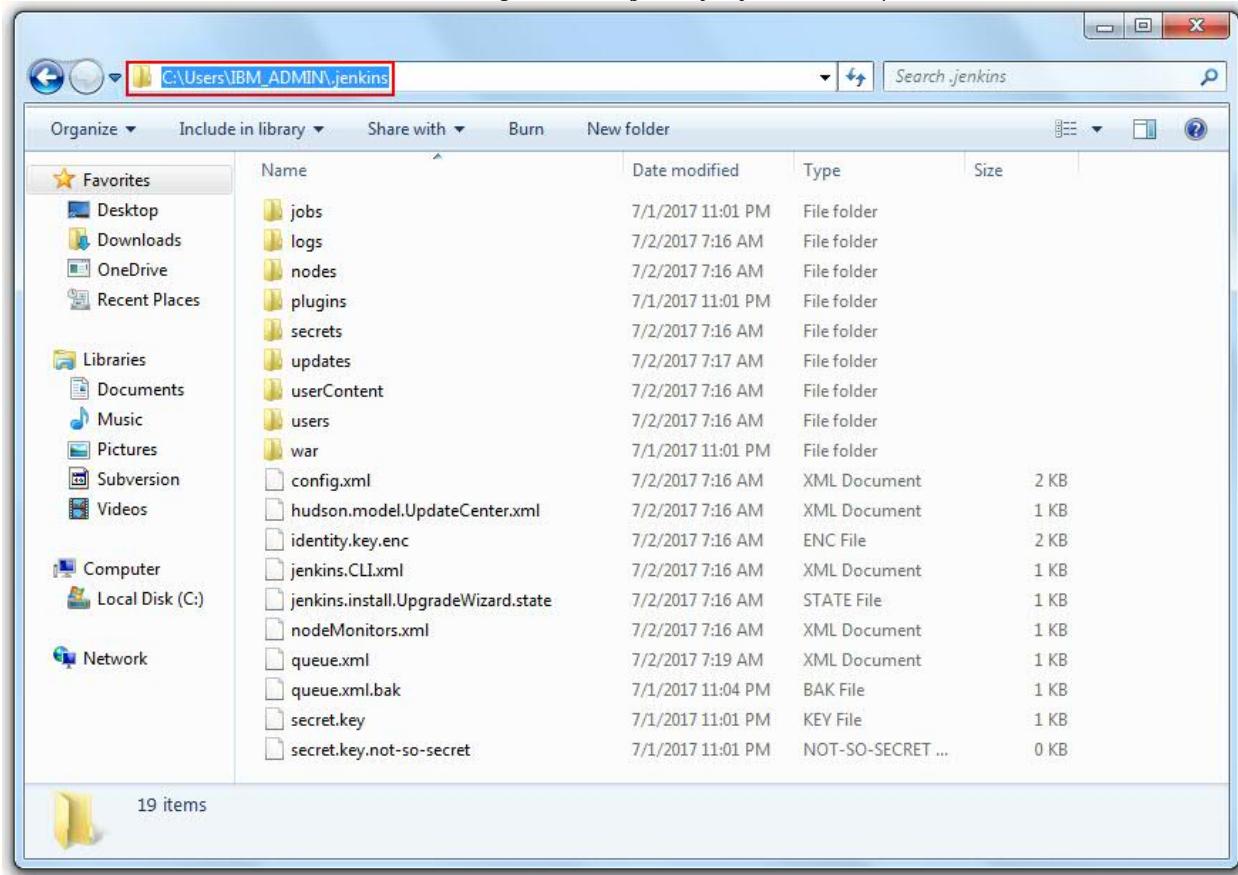
Now we can access the Jenkins webpage by using the Url: <http://ipaddress of server:8080> or <http://localhost:8080>

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\MITHUN_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 how many 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....

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's a sidebar with links: 'New Item' (highlighted with a red box), 'People', 'Build History', 'Manage Jenkins', 'My Views', and 'Credentials'. Below this are two sections: 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (showing '1 Idle' and '2 Idle'). On the right, the main area displays the 'Welcome to Jenkins!' message with 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: Add ▾

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.

The screenshot shows the Jenkins project configuration interface. The top navigation bar includes General, Source Code Management, Build Triggers (selected), Build Environment, Build, and Post-build Actions. The Subversion SCM provider is selected. The Build Triggers section shows the 'Poll SCM' option checked, with a schedule of 'H/5 * * * *'. A note indicates it would last have run at Tuesday, 27 June, 2017 6:20:22 AM IST and next run at Tuesday, 27 June, 2017 6:25:22 AM IST. The Build Environment section contains options for workspace cleanup and timestamps. The Build section shows an 'Invoke Ant' step with a 'Targets' field and an 'Advanced...' button. The page footer features the 'Technologies' logo.

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:

Add build step ▾

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

| MTTR | Last 7 Days | 21 hr |
|------|--------------|-------------|
| | 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 |

Possible Errors

The screenshot shows the Jenkins Global Tool Configuration page. Under the 'Git' section, there is a 'Repository URL' field containing 'git@github.com:devopstrainingblr/Ant-JavaProject.git'. Below it, a red error message states: 'Failed to connect to repository : Error performing command: git.exe ls-remote -h git@github.com:devopstrainingblr/Ant-JavaProject.git HEAD'. There is also a 'Credentials' dropdown set to '- none -' and an 'Add' button.

Solution

Go to the Jenkins dashboard, Click on Manage Jenkins ---> Global Tool Configuration

In Git option,

Give the Gitbash installed path in **Path to Git executable** text filed as follows.

The screenshot shows the Jenkins Global Tool Configuration page under the 'Git' section. It displays a 'Git installations' table with one entry named 'Default'. The 'Path to Git executable' field is highlighted with a red box and contains the value 'C:\Program Files\Git\bin\git.exe'. Other fields include 'Name' (Default), 'Install automatically' (unchecked), and a 'DELETE GIT' button. At the bottom are 'SAVE' and 'APPLY' buttons.

Possible Errors

```
+refs/heads/*:refs/remotes/origin/*" returned status code 128:  
stdout:  
stderr: remote: Password authentication is not available for Git operations.  
remote: You must use a personal access token or SSH key.
```

Solution

If you see this error, generate SSH or PAT and use these keys instead of password.

Disable Build:

A disabled Build will not be executed until you enable it again. This option often comes in handy to 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

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.

SSH Agent:

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

Thin Backup:

Create the Maven project/job in Jenkins

Method 1:

Install the **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.

Mithun Technologies
Mithun
Technologies

Enter an item name

» 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.



OK

GitHub Organization

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

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

Mithun
Technologies

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/>

The screenshot shows the Jenkins Global Tool Configuration page for Maven. It displays a single Maven installation named "maven". The "Install automatically" checkbox is checked. Below it, there's an "Install from Apache" section with a dropdown set to version 3.5.0. A "Delete Installer" button is located to the right. At the bottom, there are "Add Maven" and "Delete Maven" buttons.

Possible Errors

```
[ERROR] COMPILATION ERROR :  
[INFO] -----  
[ERROR] No compiler is provided in this environment. Perhaps you are running on a JRE rather than a JDK?
```

Solution1

Set the class path for Java.

Solution2

Go to the Jenkins Dashboard ---> Click on Manage Jenkins ---> Global Tool Configuration ---> in JDK section give the full path where u have installed the Java.

The screenshot shows the Jenkins Global Tool Configuration page for JDK. It displays a single JDK installation named "Java". The "JAVA_HOME" field is set to "C:\Program Files\Java\jdk1.8.0_162". A "Delete JDK" button is located to the right.

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 --->

Mithun Reddy Lacchannagari
+919980923226

Jenkins

SMTP server

Default user E-mail suffix

Use SMTP Authentication

User Name

Password

Use SSL

SMTP port

Charset

Default Content

Default Pre-send Script

Default Post-send Script

Additional groovy classpath

Enable Debug Mode
 Require Administrator for Template Testing
 Enable watching for jobs

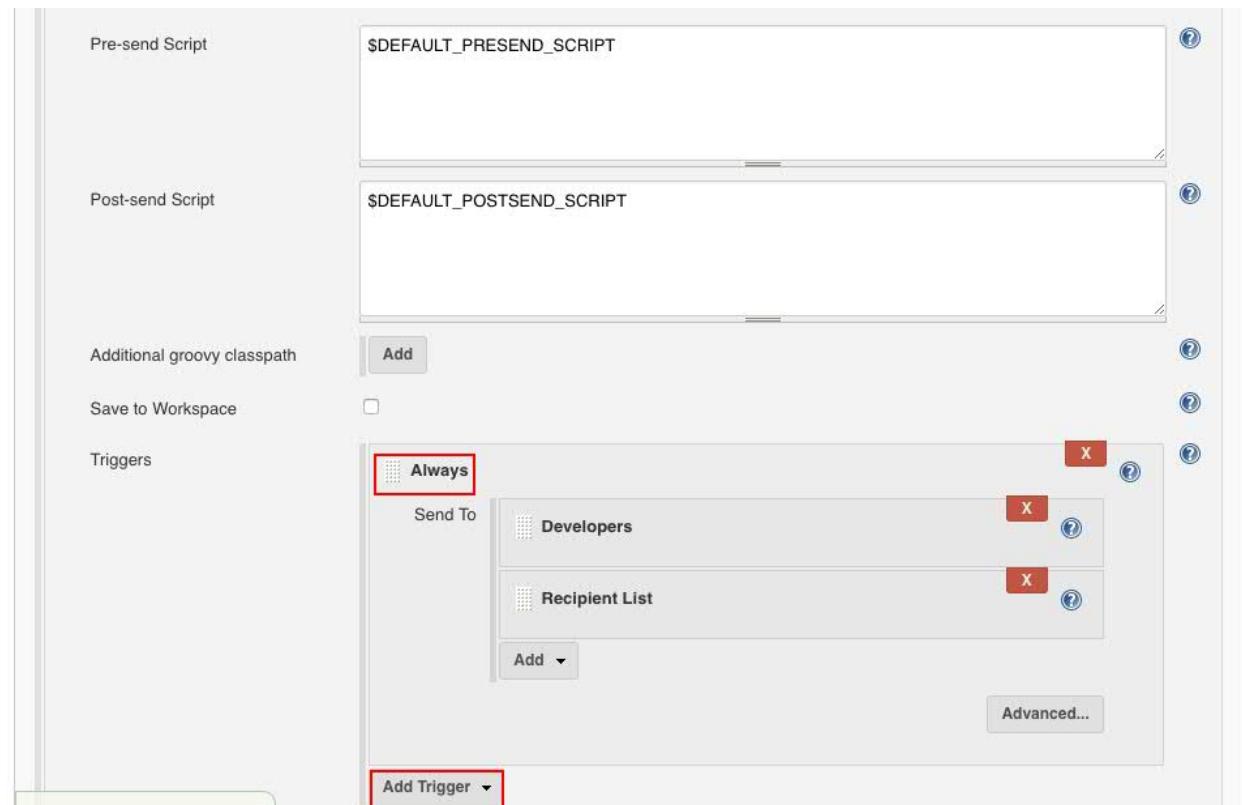
Content Token Reference

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**.

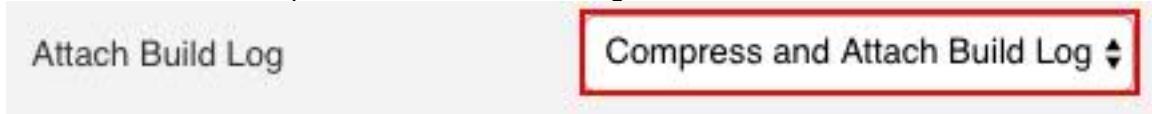


We can enable to attach the build logs while sending mail, as follows.



Output mail is like below.

We can enable to Compress and Attach Build Log to email as follows.



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.

Mithun Reddy Lacchannagari
+919980923226

Jenkins

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

Generate the SonarQube server authentication token

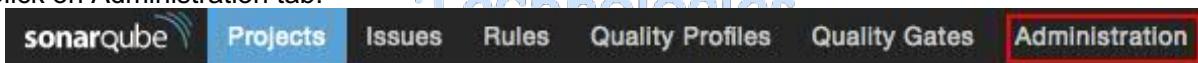
Login into SonarQube with Admin user.

<http://localhost:9000/>

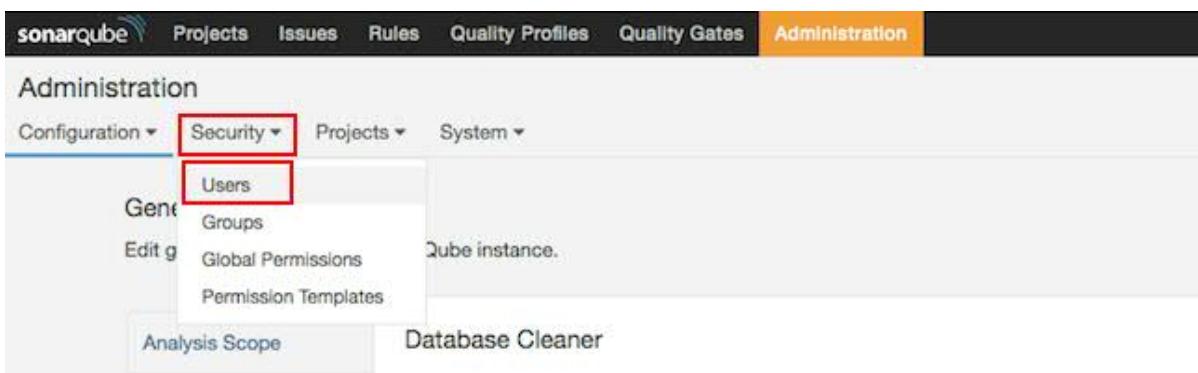
User: admin

Pwd: admin

Click on Administration tab.



Click on Security dropdown and select the Users



Click on Tokens

Mithun Reddy Lacchannagari
+919980923226

Jenkins

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

Mithun Technologies

Mithun Technologies

<http://mithuntechnologies.com> mithunreddytechnologies@gmail.com

Mithun Reddy Lacchannagari
+919980923226

Jenkins

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!

8e41f1f714a20ed8c77c54f3655ed863e5b19110

Mithun
Technologies

Configure SonarQube Scanner for Job/project

The screenshot shows the Jenkins job configuration interface. The 'Build' tab is selected. In the 'Execute SonarQube Scanner' section, the 'Analysis properties' field contains the following Java code:

```
# 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
```

Below this, there are fields for 'Additional arguments' and 'JVM Options', both currently empty.

Mithun Technologies

The screenshot shows the Jenkins job configuration interface. In the 'Execute SonarQube Scanner' section, the 'SonarQube Scanner' field contains the following error message:

**Jenkins needs to know where your SonarQube Scanner is installed.
Please do so from the global tool configuration.**

The other fields in this section are empty.

If you see above error, please do the below steps.

Go to the Jenkins Dashboard ---> Click on Manage Jenkins ---> Global Tool Configuration ---> in SonarQube Scanner section → click on SonarQube Scanner installations... and give the below details.

SonarQube Scanner

SonarQube Scanner installations

SonarQube Scanner

Name

Install automatically

Install from Maven Central

Version

Delete Installer

Add Installer

Delete SonarQube Scanner

Save Apply

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

Mithun Reddy Lacchannagari
+919980923226

Jenkins

Username:
Password:
Confirm password:
Full name:
E-mail address:
Click on Create User

The screenshot shows the Jenkins 'Create User' interface. At the top, there's a navigation bar with 'Jenkins' and 'Jenkins' own user database'. Below it, a sidebar has 'Back to Dashboard', 'Manage Jenkins', and a 'Create User' button, which is highlighted with a red box. The main area is titled 'Create User' and contains a form with the following fields:

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

At the bottom is a 'Create User' button.

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).

The screenshot shows the Jenkins 'Configure Global Security' configuration page. The 'Enable security' checkbox is checked and highlighted with a red border. The 'TCP port for JNLP agents' section has 'Disable' selected. The 'Agent protocols...' button is visible. Under 'Access Control', 'Disable remember me' is unchecked. The 'Security Realm' section is expanded, showing options for 'Delegate to servlet container', 'Github Authentication Plugin', 'Gitlab Authentication Plugin', 'HTTP Header by reverse proxy', 'Jenkins' own user database' (which is selected and highlighted with a red border), and 'Allow users to sign up' (which is checked and highlighted with a red border). Below this, 'LDAP' and 'Unix user/group database' are listed. The 'Authorization' section shows 'Anyone can do anything' selected.

Authorization

Anyone can do anything
 Legacy mode
 Logged-in users can do anything
 Matrix-based security

Project-based Matrix Authorization Strategy

| | Overall | Credentials | Agent | Job | Run | View | SCM |
|---------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|
| User/group | Administer | Read | Create | ManageDomains | Update | View | Build |
| Administrator | <input checked="" type="checkbox"/> |
| devops | <input checked="" type="checkbox"/> |
| user1 | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| user2 | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Anonymous | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

User/group to add: **Add**

Markup Formatter

Plain text

Treats all input as plain text. HTML unsafe characters like < and & are escaped to their respective character entities.

Prevent Cross Site Request Forgery exploits

Save **Apply**

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



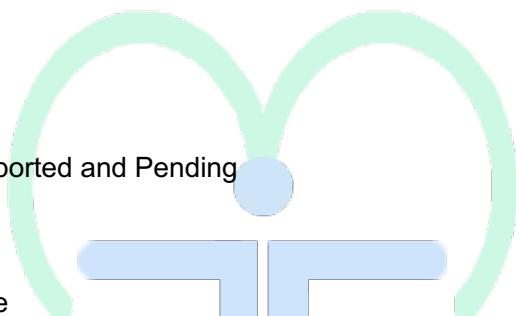
GREY: Disabled, Aborted and Pending



YELLOW: Unstable



RED: Failed



Jenkins Job Weather Status Icon Colours



Mithun Technologies



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 Poll SCM 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 */22 * * *`

⚠ 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

Deploy the application into Tomcat

Install the “Deploy to container” 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 selected. A sub-menu titled "Gradle-Artifactory Integration" is open, showing various actions like "Aggregate downstream test results" and "Deploy war/ear to a container". The "Deploy war/ear to a container" action is highlighted with a blue selection bar. Below this, other actions include 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 sub-menu is a "Save" button.

Post-build Actions

Deploy war/ear to a container

- WAR/EAR files: `**/*.war`
- Context path: `SampleAntProject`
- Containers:
 - Tomcat 8.x
 - Credentials: `admin/*****`
 - Tomcat URL: `http://mithuntechnologies.com:8083`

Deploy on failure:

Add post-build action ▾

Save **Apply**

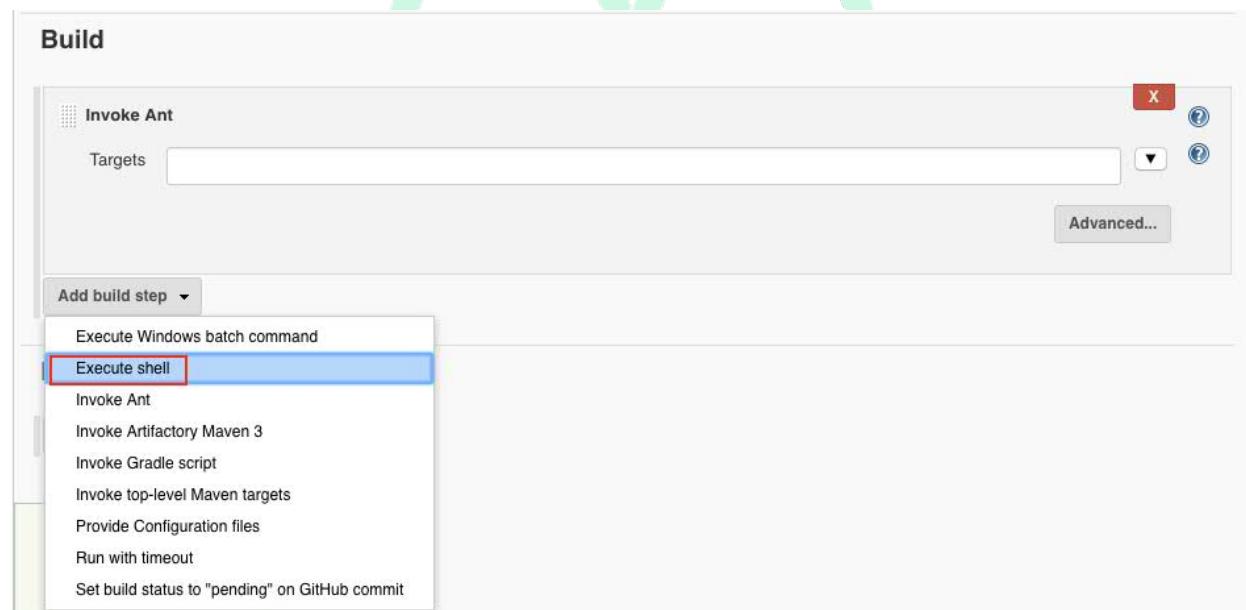
Error:

```
Caused by: org.codehaus.cargo.container.tomcat.internal.TomcatManagerException: The username you provided is
not allowed to use the text-based Tomcat Manager (error 403)
    at org.codehaus.cargo.container.tomcat.internal.TomcatManager.invoke(TomcatManager.java:704)
    at org.codehaus.cargo.container.tomcat.internal.TomcatManager.list(TomcatManager.java:876)
    at org.codehaus.cargo.container.tomcat.internal.TomcatManager.getStatus(TomcatManager.java:889)
    at
org.codehaus.cargo.container.tomcat.internal.AbstractTomcatManagerDeployer.redeploy(AbstractTomcatManagerDeployer.java:173)
    ... 17 more
Caused by: java.io.IOException: Server returned HTTP response code: 403 for URL:
http://localhost:8085/manager/text/list
    at sun.net.www.protocol.http.HttpURLConnection.getInputStream0(HttpURLConnection.java:1894)
    at sun.net.www.protocol.http.HttpURLConnection.getInputStream(HttpURLConnection.java:1492)
    at org.codehaus.cargo.container.tomcat.internal.TomcatManager.invoke(TomcatManager.java:571)
    ... 20 more
```

Solution: Need to add rule in tomcat-users.xml file as follows.

```
<user username="admin" password="passw0rd" roles="admin-gui,manager-gui,manager-script" />
```

(OR)



Add the below script in **Execute shell**

Linux/MAC for Tomcat

```
#!/bin/sh
echo "Starting to copy the build artifact"
cp $WORKSPACE/war/SampleAntProject.war
/Users/bhaskarreddyl/BhaskarReddyL/Softwares/Running/apache-tomcat-9.0.6/webapps/
echo "Deployed the build artifact into tomcat server successfully"
```

Windows

```
echo "Starting to copy the build"
copy %WORKSPACE%\war\SampleAntProject.war C:\\apache-tomcat-8.5.23\\webapps\\
echo "Copied the build to tomcat"
```

Linux/MAC for WildFly

```
#Deploy in WildFly server
```

Mithun Reddy Lacchannagari
+919980923226

Jenkins

```
#!/bin/sh
echo "Starting to copy the build"
cp $WORKSPACE/war/SampleAntProject.war
/Users/bhaskarreddyl/BhaskarReddyL/Softwares/Running/wildfly11.0.0.Final/standalone/deployment
s/
echo "Copied the build to WildFly successfully"
```

Build

The screenshot shows the Jenkins 'Build' configuration page. It contains two build steps:

- Invoke Ant**: A step with a 'Targets' field and an 'Advanced...' button.
- Execute shell**: A step with a 'Command' field containing the following script:

```
#!/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"
```

This command is highlighted with a red box.

Below the steps, there is a link to "See the list of available environment variables" and an "Advanced..." button. At the bottom left, there is a "Add build step" dropdown.

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

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

```
cp %JENKINS_HOME%\jobs\%JOB_NAME%\builds\%BUILD_NUMBER%\log
C:\Users\windows7\Downloads\newfolder\
```

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.

Artifactory

Enable Push to Bintray

Use the Credentials Plugin

Artifactory servers

Server ID: JFrog Artifactory server1

URL: http://localhost:8081/artifactory/

Default Deployer Credentials

Username: admin

Password:

Connection Timeout: 300

Number of retries: 3

Bypass HTTP Proxy

Found Artifactory 5.3.2

Test Connection

Use Different Resolver Credentials

Delete

Add Artifactory Server

Note: Once you entered all the details click on **TEST CONNECTION**. IF connection is succeeded 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 should run.
- 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/>



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
```

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)

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

2) Scripted Pipeline Syntax

Jenkins Multi Branch Pipeline Project

Required Plugins

- 1) Pipeline: Multibranch

Blue Ocean Plugin

Build Chain Executions

Create the 3 jobs like dev, stage and prod.

Configure **prod** job with as follows.

Build Triggers

Trigger builds remotely (e.g., from scripts)

Build after other projects are built

Projects to watch **stage**

Trigger only if build is stable

Trigger even if the build is unstable

Trigger even if the build fails

Build periodically

GitHub hook trigger for GITScm polling

Poll SCM

Click on **Save**.

Configure **stage** job with as follows.

Build Triggers

Trigger builds remotely (e.g., from scripts)

Build after other projects are built

Projects to watch **dev**

Trigger only if build is stable

Trigger even if the build is unstable

Trigger even if the build fails

Build periodically

GitHub hook trigger for GITScm polling

Poll SCM

Post-build Actions

Build other projects

Projects to build **prod.**

Trigger only if build is stable

Trigger even if the build is unstable

Trigger even if the build fails

Add post-build action ▾

Configure **dev** job with as follows.

Post-build Actions

Build other projects

Projects to build: stage

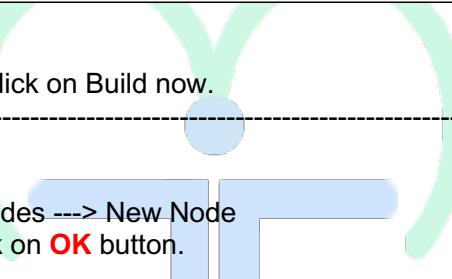
Trigger only if build is stable

Trigger even if the build is unstable

Trigger even if the build fails

Add post-build action ▾

Save **Apply**



Now you can open dev job and click on Build now.

Jenkins Master-Slave setup

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

Jenkins

Jenkins > Nodes >

Back to Dashboard Manage Jenkins New Node Configure

Node name: Node1-Mithun-Technologies

Permanent Agent

Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.

Build Queue OK

No builds in the queue.

Build Executor Status

1 Idle 2 Idle

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

Name

Description

of executors

Remote root directory

Labels

Usage

Launch method

Disable WorkDir

Custom WorkDir path

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

Fail if workspace is missing

[Advanced...](#)

Availability

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 Fixed : Random 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 |
| | | Delete Agent | ms | 6 ms | 4 ms | 16 min | 1 ms 0 ms |
| | | Configure | | | | | |
| | | | Build History | | | | |
| | | | Load Statistics | | | | |
| | | | Log | | | | |
| | | | Open Blue Ocean | | | | |

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.)

Connect agent to Jenkins one of these ways:

- Launch 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

Mark this node temporarily offline

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 agent.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 project. The 'General' tab is selected. Under the 'Restrict where this project can be run' section, the checkbox is checked and highlighted with a red box. The 'Label Expression' field contains 'Node1-Mithun-Technologies'. A yellow warning message at the bottom states: 'There's no agent/cloud that matches this assignment. Did you mean 'master' instead of 'Node'?'. There is also an 'Advanced...' button at the bottom right.

Mithun Reddy Lacchannagari
+919980923226

Jenkins

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

Installation Issues:

Issue: Offline

Offline

Offline

This Jenkins instance appears to be offline.

For information about installing Jenkins without an internet connection, see the [Offline Jenkins Installation Documentation](#).

You may choose to continue by configuring a proxy or skipping plugin installation.

[Configure Proxy](#)

[Skip Plugin Installations](#)

Solution



jenkinshomedir/hudson.model.UpdateCenter.xml and change url to use http instead of https.

Once you changed from https to http, you need to restart the Jenkins.

Issue:

While building if you see below error

```
[Test] $ ant -file build-mt.xml
ERROR: command execution failed.Maybe you need to configure the job to choose one of your Ant installations?
Finished: FAILURE
```

Solution:

Go to the Manage Jenkins ---> Global Tool Configuration ---> Ant ---> Ant Installations...

Ant

Ant installations

Add Ant

Ant

Name **ANT_HOME**

Install automatically

Install from Apache

Version **1.10.5**

Delete Installer

Add Installer

Add Ant

Delete Ant

and in Job, select the Ant Versions as follows.

Invoke Ant

Ant Version **ANT_HOME**

Targets

Advanced...

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>

<https://www.jdev.it/deploying-your-war-file-from-jenkins-to-tomcat/> ---> Deploy into Tomcat