

JENKINS

Introduction to Jenkins:

Jenkins is an open source automation web server written in Java. Jenkins helps to automate the non-human part of the software development process.

Jenkins primarily known for Continuous Integration(CI) tool, however we can do continuous deployments(CD)/Delivery also.

Jenkins is open source and freeware, Previously jenkins was called as Hudson, initially this tool was maintained by Oracle, few developers had differences with Oracle and they came out and continued this tool with the name Jenkins.

Similar tools to Jenkins:

1. Hudson
2. Bamboo
3. Circle CI
4. Codepipeline (AWS)

Continuous Integration(CI): Continuous Integration (CI) is a development practice where developers integrate code into a shared repository frequently(like Git), preferably several times a day. Each integration can then be verified by an automated build, tests and automated build package also.

Continuous Deployment: It is next level of continuous Integration, Continuous Deployment is a software development practice in which every code change goes through the entire pipeline and is put into production, automatically, resulting in many production deployments every day.

Installation of jenkins in Amazon Linux/Redhat Linux:

Step-1:

Jenkins is dependent of java 1.8.0 so we want to install java1.8.0 before installing Jenkins. Use following command to install java-1.8.0

```
sudo yum install java-1.8.0-openjdk-devel -y
```

If you had multiple versions of java , if you want 1.8.0 as default use following command to change.

```
sudo update-alternatives --config java
```

Choose the number beside java-1.8.0

```
sudo update-alternatives --config javac
```

Choose the number beside java-1.8.0

Step-2:

To perform git operations in jenkins we want to install git on jenkins server. So we want git on jenkins server.

JENKINS

```
sudo yum install git -y
```

Step-3:

To perform Maven operations in Jenkins we want to install Maven on Jenkins server. So to install Maven use following commands.

1. `sudo wget http://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo`
2. `sudo sed -i s/$releasever/6/g /etc/yum.repos.d/epel-apache-maven.repo`
3. `sudo yum install -y apache-maven`
4. `mvn --version`

Step-4:

To install Jenkins on Redhat Linux use following commands.

1. `sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo`
2. `sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key`
3. `sudo yum install jenkins -y`

Step-5:

To start Jenkins use following command.

```
sudo service jenkins start
```

Step-6:

To enable Jenkins start on reboot use following command.

```
sudo chkconfig jenkins on
```

Configuring Jenkins:

Note: Jenkins default runs on port number **8080**.

1. To access Jenkins via web browser use public IP of Jenkins server and use it.
<http://ip-address:8080>
2. To unlock Jenkins take initial admin password from following path, enter it and continue.

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

3. *Install suggested plugins* → *Continue*
4. *Create user with username, password, name and email* → *Save and Continue*.
5. Start using Jenkins

Creating Jenkins New Build Job:

From Jenkins Dashboard → *New Item* → *Enter an item name* → *Freestyle project*.

JENKINS

Step-1 In Source Code Management Section Select GIT → Enter Repository URL → If that is Private Repo → Create New Credentials and select Credentials

The screenshot shows the 'Source Code Management' configuration page in Jenkins. On the left, there are radio buttons for 'None' and 'Git', with 'Git' selected. Below this is a sidebar with 'Repositories', 'Branches to build', and 'Repository browser'. The 'Repositories' section contains a 'Repository URL' field with the value 'https://github.com/devops-mpotech/sample-app' and a 'Credentials' dropdown menu showing 'devop-mpotech/***** (Github)'. There are 'Advanced...' and 'Add Repository' buttons. The 'Branches to build' section has a 'Branch Specifier (blank for 'any')' field with the value '*/master' and an 'Add Branch' button. The 'Repository browser' field is set to '(Auto)'.

Step-2: Add Build Step

Take Invoke Top-Level Maven Targets

Build

The screenshot shows the 'Build' configuration page in Jenkins. It features a section titled 'Invoke top-level Maven targets' with a red 'X' icon and a help icon. Below this is a 'Goals' field containing the text 'clean package'. There is an 'Advanced...' button and an 'Add build step' button at the bottom left.

Step-3: Apply and Save

Step-4: Build Now

- Back to Dashboard
- Status
- Changes
- Workspace
- Build Now
- Delete Project
- Configure
- Rename

JENKINS

Jenkins Job to Build a war file and Deploy to Tomcat:

Before going to deploy you want to download the **“Deploy to Container”** Plugin from Jenkins Store

From Jenkins Dashboard → Manage Jenkins → Manage Plugins → From Available Section → Search for “Deploy to Container” plugin → Select it → Install without restart

Note: If you want to deploy war file via this Plugin you must be enable “Tomcat Manager app” and create at least one user.

Enter into Job → Configure → Add Post-build Actions → Select Deploy War/Ear to container → Enter war file location → Add Container → Select Tomcat 8.x → and configure tomcat credentials and url of tomcat → Apply and Save → Build Now.

Post-build Actions

The screenshot shows the 'Post-build Actions' configuration page in Jenkins. The 'Deploy war/ear to a container' action is selected. The 'WAR/EAR files' field contains 'target/myapp-0.0.7.war'. The 'Context path' field is empty. Under 'Containers', 'Tomcat 8.x' is selected. The 'Credentials' dropdown shows 'tomcat/***** (Tomcat)' with an 'Add' button next to it. The 'Tomcat URL' field contains 'http://34.227.10.60:8080/'. There is an 'Add Container' button below the container list. At the bottom, there is a checkbox for 'Deploy on failure' which is unchecked, and an 'Add post-build action' button.

Suppose if you don't want to change the Version of the project every time in url follow below steps:

In Build step → Add Build Step → Execute Shell → Paste the following Commands

```
cd target/  
mv *.war myapp.war
```

Discarding Old Builds:

Discarding old builds enables to free the jenkins Server Volume otherwise unnecessary Jenkins Volume will full of old builds .

From your Job → In general Section → Select Discard Old Builds → There you may select → Days to keep build or Max number of Builds to Keep

JENKINS

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

Description

[Plain text] [Preview](#)

☒ Discard old builds

Strategy Log Rotation

Days to keep builds

if not empty, build records are only kept up to this number of days

Max # of builds to keep 5

if not empty, only up to this number of build records are kept

Advanced...

Build Triggers:

Build Triggers enables you to Trigger the the job automatically if there are new changes in GIT

From your job → Select Build triggers → Select Poll SCM → Keep all 5 stars (Means Jenkins poll every minute your GIT) → Save

This will unnecessarily waste your jenkins RAM by polling your GIT account every minute. So better approach is Select poll SCM and don't fill any thing in box.

From your Github account → Open your Project → Settings → Integration and Services → Add Service → Search Jenkins(Git) Plugin → Enter the Jenkins URL

devops-mpotech / sample-app
forked from rcp-suresh/sample-app

Watch 0 Star 0 Fork 11

Code Pull requests 0 Projects 0 Wiki Insights Settings

sample-app
Add topics

59 commits 1 branch 0 releases 3 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

JENKINS

The screenshot shows the Jenkins 'Integrations & services' page. On the left is a sidebar with navigation links: Options, Collaborators, Branches, Webhooks, Integrations & services (highlighted), Deploy keys, Moderation, and Interaction limits. The main content area is titled 'Installed GitHub Apps' and includes a description: 'GitHub Apps augment and extend your workflows on GitHub with commercial, open source, and homegrown tools.' Below this is a 'Services' section with a yellow note: 'Note: GitHub Services are being deprecated. Please contact your integrator to migrate or replace a service with webhooks or GitHub Apps.' A list of services is shown, with 'Jenkins (Git plugin)' highlighted. An 'Add service' button is in the top right. A modal window titled 'Available Services' is open, showing a search bar with 'git' and a list of services including Gitter, GitHub Auto-Deployment, Jenkins (Git plugin) (highlighted), Jenkins (GitHub plugin), GetLocalization, Packagist, and Hostedgraphite. The footer contains copyright information for GitHub, Inc. and various links like Terms, Privacy, Security, Status, and Help.

From next time onwards whenever the code changes in git it automatically triggers the job in Jenkins.

Email Configuration to Jenkins:

This will enable whenever your Jenkins job will fail Jenkins will send email automatically to configured email addresses. To enable email configuration follow the steps.

From Jenkins Dashboard → Manage Jenkins → Configure System → Email Notification → Configure Gmail SMTP Server Details.

The screenshot shows the 'E-mail Notification' configuration page in Jenkins. It contains several input fields and checkboxes for configuring email notifications. The fields are: SMTP server (smtp.gmail.com), Default user e-mail suffix (empty), User Name (devops.mpotech@gmail.com), Password (masked with dots), SMTP Port (465), Reply-To Address (devops.mpotech@gmail.com), and Charset (UTF-8). There are checkboxes for 'Use SMTP Authentication' (checked), 'Use SSL' (checked), and 'Test configuration by sending test e-mail' (checked). A 'Test e-mail recipient' field contains devops.mpotech@gmail.com. A 'Test configuration' button is at the bottom right.

Address: MP Techno Solutions, Above Titan ShowRoom, Marathahalli, Bangalore, Phone: 9606955170/1/2/3

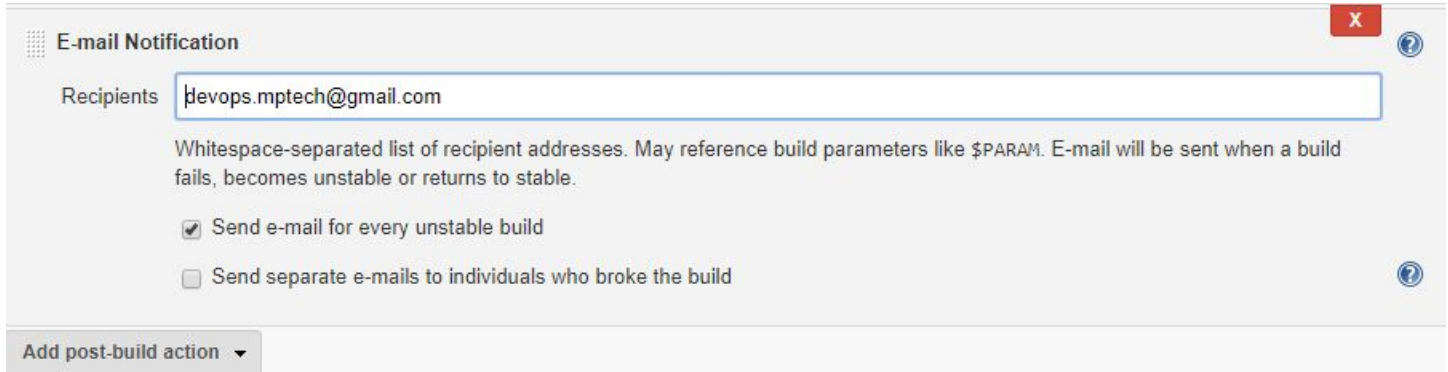
JENKINS

SMTP server(smtp.gmail.com) → Select Use SMTP Authentication → Put your gmail id → Put your gmail password → Select SSL → SMTP port 465 → Save

Note: In Real time you don't want to configure Gmail server settings, you want to configure your company smtp server settings.

After configuring you want to assign this to your Job follow the steps.

From your Job → Configure → Post Build Actions → Email Notification → Enter who wants to receive the emails for this job → And select Send email for every unstable build → Save



The screenshot shows the 'E-mail Notification' configuration panel in Jenkins. At the top, there's a title bar with a grid icon, the text 'E-mail Notification', a red close button with an 'X', and a help icon. Below the title bar, there's a 'Recipients' label followed by a text input field containing 'devops.mpotech@gmail.com'. Underneath the input field, a descriptive text reads: 'Whitespace-separated list of recipient addresses. May reference build parameters like \$PARAM. E-mail will be sent when a build fails, becomes unstable or returns to stable.' Below this text are two checkboxes: the first is checked and labeled 'Send e-mail for every unstable build', and the second is unchecked and labeled 'Send separate e-mails to individuals who broke the build'. A help icon is located to the right of the second checkbox. At the bottom of the panel, there's a button labeled 'Add post-build action' with a dropdown arrow.