**Jenkins**

* java application, since it is java app its platform independent and can be used in any OS
* it can be used in windows, mac or any other OS

Continuous integration & continuous delivery tool From latest commit and build it.

We can also add post build actions to check whether the commit worked fine. Once everything worked well, report can be sent back to the server.

1. Download Jenkins – place .war into any location on machine.
2. Go to cmd. Move to the directory where war file is stored & run command.

**java –jar Jenkins.war**

1. By Default Jenkins will run on 8080. Verify whether its running or not on browser.

# Localhost:8080 (ip of local m/c can also be used instead of localhost)

1. Jenkins do have its own server (**winstone/jetty**).

## Setting Jenkins on Tomcat

But why? As Jenkins do have its own servlet container i.e., Jetty.

Reason – u can start all the applications on one server, if u have other applications deployed there and keep Jenkins also.

There are other containers where Jenkins can be deployed – tomcat, glassfish,ibm websphere,jboss , weblogic and so on.

## Requirements for setting up:

## Apache tomcat 5 or above

Java 7 or above

* Once the tomcat is installed, go to tomcat folder ->webapps->keep the Jenkins.war file over there.
* Now go to cmd prompt. Go to tomcat bin directory.
* U will find ./startup.sh & ./shutdown.sh. Use these for starting and shutting down tomcat.
* Once tomcat is up , your Jenkins will be running as well.
* Localhost:8080- tomcat and localhost:8080/Jenkins
* To run Jenkins on a different port. Run command : java –jar Jenkins.war --httpPort=9090

**Changing Home Directory (.Jenkins)**

This directory stores job, configuration and log information.

Why to change : because of project requirements. And to move Jenkins home dire to a location that has

Enough disk space.

**Steps:**

1. Create a new directory. Copy everything from .jenkins and paste it into new directory.
2. Change environment variables.
3. Set JENKINS\_HOME to new directory.(Just the path).
4. Restart Jenkins.
5. You can verify the home directory. Manage Jenkins- Configure System

## Using CLI for Jenkins

Why? As Jenkins have a clear UI, why there is a need to use CLI.

Using CLI is Easier , faster, better memory management(efficient), Continuous integration.

**Steps:**

1. Start Jenkins
2. Manage Jenkins->configure global Security -> Enable Security should be checked.
3. Apply and save.
4. Localhost:8080/Cli -- check the commands over there, download Jenkins-cli.jar file from the page.
5. Place it anywhere in your machine. Test Whether Jenkins –cli is working or not.
6. **java -jar** Jenkins-cli.jar **-s http://localhost:8080/ help**
7. it will show some help, if it ask for sha passwords, then go to Jenkins and on the right corner

click on dropdown of your username and configure, Check the passwords added there, or you can add new password according to you. Pass the same password on CLI. It will work.

### Creating Users and assigning roles

## Steps:

1. Manage Jenkins -> manage users.
2. Create user.
3. To configure user, click on the username dropdown u logged in with and click on configure option.
4. **Manage User roles**- download **Role based Strategy Plugin** from Wiki or from Manage Plugins option in Jenkins.
5. Restart Jenkins.
6. Now go to Configure global Security , check Enable Security is checked and in authorization section, select role based strategy.
7. Now login with admin and go to **manage and assign roles section.**
8. You can see Global Roles , project roles and slave roles there.
9. Add new roles in each section and assign the permissions you want to give to the user.
10. Like in **project roles**, you can add developer, Tester and assign the permissions.
11. Ex- Global roles I added employee, and project roles I added developer and tester.
12. Now in Global roles u will see two users/group – admin and employee.
13. You can add new users and give them the permissions as given to admin and employee.
14. Same can be done in project roles. Lets say I added 2 users where user1 is a dev. And user2 is a tester. Assign the permissions accordingly.
15. We can specify patterns if we want developer to access only **devprojects** (dev.\*).here developer can only access projects starting with “dev”.

## Trigger the job remotely

In build Triggers(configure project), there is an option for Trigger builds. Authentication token can be anything(ex-1234). Now run this url into any browser and the job will be triggered.

Ex - Localhost:8080/job/Test1/build?Token=1234 where Test1 is the project name and token is 1234 that we have set.

This is used to trigger the job from outside.

## Jenkins Integration with GIT

Add new project and run in Jenkins to check if its working fine or not.

Now to commit code on git.

Go to your project location and type command **git init**.

**Then git status**

**git add .**

**git status – to check the status**

**git commit –m “comment”**

**git remote add origin** [**https://github---project**](https://github---project) **location**

**git push –u origin master**

**check on github, the project will be commited.**

Now to **integrate Jenkins with Git**, go to manage Jenkins -> manage plugins - > search for git plugin and install.

Now once it is installed u can find **git** available under **SCM** section.

Give the path of repository that we created on git.

Now in **Build Triggers -> Poll Scm** and set the cron pattern to take the code from Git and poll as soon as new code changes are done in repository.

Using **Catlight** with Jenkins.

Catlight is a tool which is used as a status notifier for developers. It is used when we have multiple projects to monitor. It will give us the information on the screen itself regarding the builds whether its failed or passed.

Download the zip -> unzip it.

Then there will be option to configure Jenkins or any other CI server.

Once configured you can see the projects as we have on Jenkins.

**Automated Deployment -**  is the process of automating the deployment process in a continuous delivery system.

**How to do it:**

**Steps:**

* Download Deploy to container plugin and restart Jenkins.
* Now create a project. So we basically deploy EAR/WAR file to a container on deployment.
* Once the EAR/WAR file is generated from Deliverables. We will deploy them to containers.
* In post Build Actions you can find an option to deploy EAR/WAR to container.
* Give the path of your war file where it will be stored. Normally it will be stored in Jenkins workspace location which u can find out in Configure Jenkins section.
* Now give the context path of your war file which is basically the name of war generated.
* Select the containers where you want to deploy the wars.
* Give the username & password for the tomcat. Same can be configured in tomcat-user.xml which is stored under conf folder inside tomcat. New user can be added.
* Build the project. Once done successful you can verify it by going to localhost:8080/name of your war

### Sending Emails from Jenkins

**Steps:**

* Manage Jenkins-> Configure System ->E-mail Notification
* Ex – gmail need to be configured.
* Set SMTP server as **smtp.gmail.com**
* Go to advanced section -> check Use SMTP Authentication.
* Give the username & password of the account from where mails will be triggered.
* Check Use SSL.
* SMTP Port -465,or 587 or 25(use any of them)
* Check test email and give the recipient name over there in the text field.
* Configure your projects , in the postbuild actions you can set email triggering if the build is failed or unstable.
* There are some other plugins present which can send the email in different formats like **Notification plugin, extreme notification plugin & Email ext plugin.(to send the email in json. Webhook and other advanced formats).**