Jenkins

What is Jenkins?

Jenkins is an open source automation tool written in Java with plugins built for Continuous Integration purpose. Jenkins is used to build and test your software projects continuously making it easier for developers to integrate changes to the project and making it easier for users to obtain a fresh build.

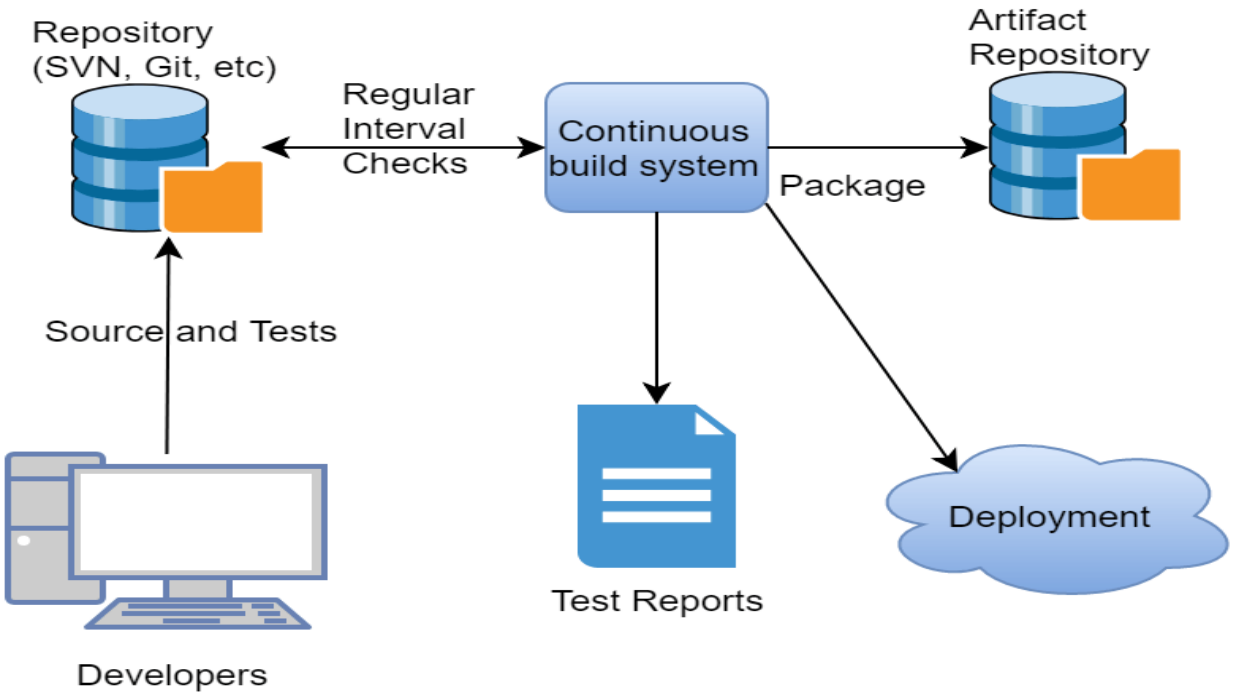
Jenkins installation:

Jenkins can be installed in 3 ways:

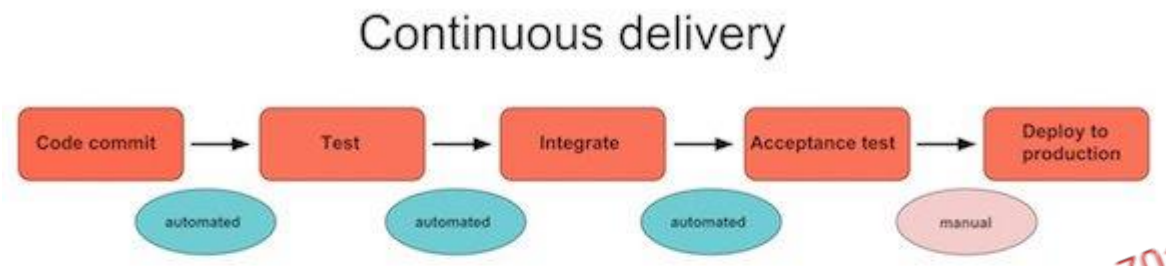
1) Generic way
2) Installing software on respective Operating System.
3) Deploying Jenkins war into Tomcat webapps.

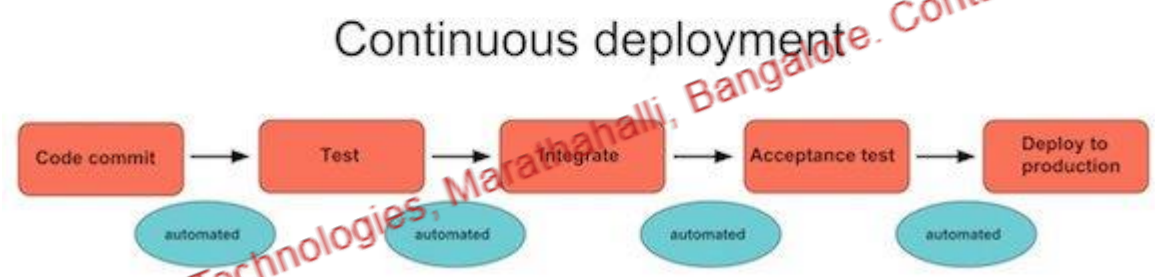
Jenkins works based on Plugins. To integrate any tool with Jenkins, we need to install related plugin to Jenkins.

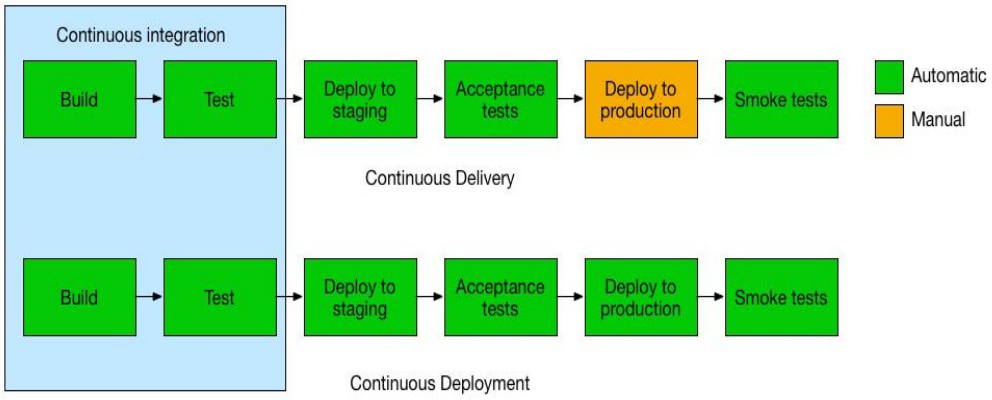
Continuous Integration Flow:



Difference between Continuous Delivery and Continuous Deployment:







Installing Jenkins:

Creating free style job in Jenkins:

Login to Jenkins, click on new item.

Enter the project name followed by environment name, then select the type of project. For most common projects, we select free style project option, then click on Ok at the bottom.

Then go to Source Code management, Select Git, give the git repo url, then add the git credentials, then select those added credentials.

Then give the branch name at: Branches to build. Ex:*/master

Then go to Build Environment: Select "With Ant"(only for Ant projects).

Go to Post-build Actions: Select Editable Email Notification, go to advanced settings, Select Add trigger, select the option as Always-> which sends email for all failure, success cases.

We have different trigger options to send emails.

Before that, install **Email Extension plugin** and then go to Configure System and add SMTP server details.

Difference between Build periodically and Poll SCM?

- Build periodically will trigger the builds as per the schedule given in the Cron expressions, even there are no changes/commits to git repo.
- Poll SCM will trigger the builds as per the schedule given in the Cron expressions, only when there are changes/commits to git repo only.

SMTP configurations:

Login to Jenkins, click on Jenkins on left top corner.

Go to Manage Jenkins. Go to Configuration System, go to E-mail Notification, then click on Advanced.

Enter the SMTP server name: ex: smtp.gmail.com

Then, Enable Use SMTP Authentication option.

Enter username: abcd@gmail.com, and gmail password

Then, Enable Use SSL option.

SMTP Port: 465, default port is: 25(without SSL)

Then click on save.

Jenkins integration with SonarCube for freestype projects:

To generate the token to connect from jenkins to Sonar scanner,

Login to SonarQube, then go to Administration, then go to security---> select token, then give token name, generate token and enter the sonarqube generated token in Jenkins.

Ex:094be29463837bfe66ac106c2bf5505549608628

Then go to Jenkins, Build and in pre-builds select---> Execute SonarQube Scanner and enter the details:

ex:

must be unique in a given SonarQube instance

sonar.projectKey=com.kctech.java

this is the name and version displayed in the SonarQube UI. Was mandatory prior to SonarQube 6.1.

sonar.projectName=AntJavaProject

sonar.projectVersion=2.0

Path is relative to the sonar-project.properties file. Replace "\" by "/" on Windows.

This property is optional if sonar.modules is set.

sonar.sources=src

When accessing multiple sonar scanner for different projects, then the above lines needs to be repeated with different projectKey.

Otherwise it will override.

To run the maven goal in Jenkins for sonar scanner, then in the Build option, select Execute batch command for windows or Execute Shell option and give below goal:
mvn clean sonar:sonar

When we forget password in jenkins:

Go to jenkins home directory(.jenkins), go to config.xml file and change <denyAnonymousReadAccess> element value from true to false

Ex:C:\Users\KCTech\.jenkins

<denyAnonymousReadAccess>true</denyAnonymousReadAccess>
change this to
<denyAnonymousReadAccess>false</denyAnonymousReadAccess>

Plug In to add Tomcat plugin to jenkins:

q) What is the PlugIn to add Tomcat plugin to jenkins:

Ans: Deploy to container

Adding plugins to Jenkins(SonarQube):

Go to manage jenkins, then Manage Plugins, go to available, search for sonarqube, then install without restart, then restart Jenkins. That's it.

Adding Tomcat plugins to Jenkins:

Go to manage jenkins, then Manage Plugins, go to available, search for Deploy to container, then install without restart, then restart Jenkins. That's it.

Starting Jenkins:

Go to jenkins war folder. Ex:C:\DevOps\Jenkins\Jenkins_War

Type the command: java -jar jenkins.war

Verify by logging from jenkins console: http://localhost:8080

By default, Jenkins starts on 8080 port number

To start with different port number, use below command.

java -jar jenkins.war –httpPort=8081

To start Tomcat server:

Go to tomcat server path: C:\KCTech\DevOps\Tomcat\apache-tomcat-8.0.46\bin

Start the server with this command: catalina.bat start

Verify tomcat is up or not by logging in from console: http://localhost:8080/

To start Sonar server:

Go to sonar server path. Ex:C:\ KCTech\DevOps\SonarCube\sonarqube-6.3.1\bin\windows-x86-64

Then, type this command: SonarStart start

Verify by logging in from console: http://localhost:9000

We can do Deployment to tomcat in 2 ways:

1st way) By copying war to webapps folder: shellscript:

Go to project-->Configure--->Go to Builds, Select "Execute Windows batch command" and paste the below 3 lines.

command to paste:

echo "Starting to copy the build"

copy %WORKSPACE%\war\SampleAntProject.war C:\\apache-tomcat-8.5.23\webapps\

echo "Copied the build to tomcat"

Change the tomcat webapps folder as per your path in the above lines.

Again in Build, go to Add build steps--> select Invoke Ant--> Then give the target(location where build.xml exists, root location is project location),

If the build.xml is under project root directory, then leave the Target as empty.

Then give the build file name ex: build.xml or build-kc.xml (custom file)

Then save and run the build.

Note:**)** Use **Execute Shell option** for Linux systems

2nd Way) Using Deploy to container plugin.

Jenkins integration specific to Maven Style projects:

Go to Manage Plugins option, then search for the plugin name "Maven Integration" plugin, then add and install it.

Then create a job with Maven style Project instead of free style project.

The benefit of selecting Maven project is You will get much options and you will get options which are only related to Maven, so that will be easy to choose the options.

Few more important points:

In build result email, we can attach log file in the email.

In the Post-build actions, select Attach build file, then it will attach the log file and sends the email.

To compress and attach the log file,

In the same Post-build actions, select the compress and attach build log option, then it will compress the log file and adds as an attachment and sends the email.

To Restart jenkins from Console:

Login to console,

In the url, change the context root like, localhost:8080/safeRestart Then confirm to restart jenkins.

Or

in the console, click on "Restart Safely" option and confirm yes.

So Jenkins will restart once all the running jobs are finished.

To force restart, without depending on running jobs status, then change the context root as

http://localhost:8080/restart

Difference between /safeRestart and /restart is:

safeRestart: Will wait until the running jobs to complete.

restart: Will not wait until the running jobs to complete. kills the current running jobs and restart Jenkins.

Adding Open Blue Ocean plugin:

In the Mange Plugins, search for blue ocean plug in and then download and install. After adding the plugin, you will see Open Blue Ocean in UI console.

If you click on that, then the screen will change

UI screen of Jenkins will change.

Blue ocean will create a good User Interface for Jenkins pipelines and jobs.

To go back from Blue Ocean Ui screen to Normal Jenkins UI screen:

Just click on exit button on right top corner.

Or just log out and login

To create our own custom view:

Click on Jenkins dropdown on top left corner, then select all, then You will see All, + options,

Click on + symbol, then give the view name and add what all the jobs needs to be listed under that view and all you can select different display option like lastDuration, lastFailure, etc etc.

For example, you can create a View with name JAVA, and then add only java related jobs to that view.

Create one more View with the name PYTHON, and then you can add only Python related jobs to that view.

Now, All option will show all jobs. JAVA view will show only java projects, etc...

Under 1 View ex:java, if you want to create a nested view, then you have to download **nested view plug-in** and then create nested view.

Creating Users in Jenkins:

Go to Manage Jenkins → Manage Users, then create user and then give users details.

By default, this will create as an Admin user.

To restrict admin access to the user:

Go to **Manage Jenkins**, go to **Configure Global Security**,

Under, Logged-in users can do anything, just enable the "Allow anonymous read access" checkbox.

This will show the Jenkins console options except manage Jenkins, even if you are not logged-in and just hit console url: http://localhost:8080

To disable admin access to users on global level (To all projects):

Go to Manage Jenkins, Configure Global Security, then Anyone can do anything, then will show all the options, without login also.

Go to Manage Jenkins, Configure Global Security, then select "Project-based Matrix Authorization Strategy" option:

Then, add the user, whom you want to restrict access, add the user, then enable the check boxes what you want to provide the access, ex: Read (Admin), Read (Job), then save.

So, the added user can see only the restricted access with the options what you enabled with checkboxes.

Using Groovy script for pipeline in Jenkins:

To run the build using customized script instead of UI configurations, we use pipeline with groovy script.

Jenkinsfile is the file we use for pipeline. DevOps engineer will right it.

For normal pipeline, groovy script is mandatory.

We can write Jenkinsfile in 2 forms: declarative and scripted formats.

We follow scripted format.

While creating the pipeline project, while creating job select pipeline project option.

Then nothing to select, just go to Advance project options, then select Definition as: Pipeline script from SCM, then give GIT repo url and select credentials. Then click on build now.

Note: In the groovy script, change sh to bat for windows systems.

Jenkins Master Slave Configuration:

To reduce the work load to jenkins and maintain load balance, we create slaves or Nodes in jenkins.

Nodes exists in different Linux servers, so CPU memory of Server where Jenkins exists will be saved.

Configurations will be on master only. Slaves will just run the jobs.

For practice, we create Salves(nodes) in same server.

To create Nodes:

Go to manage jenkins, then go to manage nodes, -->New Node, give name Node name, give some description.

No of executors: No.of jobs needs to be run parallel when all the jobs are triggered at once.

Remote root directory: While running the jobs, Jenkins may generate some temp files, those temp files will be created here.

Labels: Is logical group of Nodes. Multiple nodes can link to same label.

Based on the label configured and based on links to label, label can balance and share the builds between different nodes.

To get this advantage, while configuring the job, we can give the label name instead on slave node name.

Now you can see, "Launch agent via Java Web start option under **Launch Method**.

To get that,

Go to Manage Jenkins, then go to Configure Global Security, then go to agents,

There, if you select fixed, till 1024 are fixed ports, so give above 1024.

Random don't need port number, so you can select that.

Then select Java Web Start Agent Protocol/1 (deprecated, unencrypted)

Then you can see **"Launch agent via Java Web start**" option. under launch method.

Then click on node.

Then click on agent.jar to download. Put it in the location where jenkins.war exists.

Go to the downloaded file path (C:\ KCTech\DevOps\Jenkins\Jenkins_War): and enter this command;

java -jar agent.jar -jnlpUrl http://localhost:8080/computer/slave-node1/slave-agent.jnlp -secret 4b82500d8dd124505f6eda2a7a1e2851c52fc52cb2875e05437eeb7853930ace -workDir "C:\ KCTech\DevOps\JenkinsTempFiles"

In real time, we use launch method. we select launch slave agents via SSh, and then we give the host name and other credentials.

With this we no need to download agent.jar and no need to run below commands.

This command you can get in Jenkins below the Launch button;

java -jar agent.jar -jnlpUrl http://localhost:8080/computer/slave-node1/slave-agent.jnlp -secret 4b82500d8dd124505f6eda2a7a1e2851c52fc52cb2875e05437eeb7853930ace -workDir "C:\ KCTech\DevOps\JenkinsTempFiles"

To inform each to run on specific node:

Go to project configure, then General, then enable, Restrict where this project can run.

Give the Label name, so it will run on the nodes which are links to this label. So, it can share the nodes to run the job.

If you give node name, it will always run on same node.

Java code coverage plug-in:

For java code coverage, we have to add **jacoco plugin** to Jenkins.

To check the code coverage, in the post build actions, we can restrict the percentage of code coverage, if the code coverage is less than the mentioned percentage, then it will fail the build.

Multi Branch pipeline in jenkins:

To create build jobs at a time for multiple branches present in same git repository, then we go for multi branch pipeline.

It will find the branches based on indexes and creates and triggers the build.

When a branch is deleted, then it automatically deletes the job as well.

To create multi branch pipeline, create New Item, then give branch name, then select Multi Branch pipeline style project,

You have to write groovy script to use multi branch pipeline, if you don't want to use UI provided options.

In the Add source, select Git, give git repo and credentials. You no need to select each branch. It will create jobs for all the branches under that project repo url.

If you add, one more branch after creating multi branch pipeline, then If you enable the "Periodically if not otherwise run" option in the configure, then it will find even the new branches and creates new jobs and triggers builds.

Even if you delete a branch, then it will detect automatically and stops triggering build for that project.

Sample groovy script to use in Jenkins:

```
node {
  git url: 'https://github.com/jglick/simple-maven-project-with-tests.git'
  def mvnHome = tool 'M3'
  sh "${mvnHome}/bin/mvn -B verify"
}
```

Dependency Jobs:

In this case one job will run only the other job runs i.e a child job will depend on Parent job and the Child jobs runs only when the Parent Job completes execution.

Create 2 Jobs with the names Job1 and Job2.

Then go to child job (Job2) configuration. Go to Post build actions. Select Build other projects.

Then give the parent job (Job1) and select 1 option among the 3 options.

Note: Here you have dependency chain.

Then click on Build now for Parent i.e Job1. Then the child Job (Job2) will be triggered automatically only when the Parent job completes, based on the select option among 3.

Job2 will show Job1 as upstream project. Job1 will show Job2 as downstream project.

Build Triggers:

One build Job will be triggered after other projects are build:

Give the parent job name and click on Apply and save

Difference between **Dependency Jobs and Build after other projects are build** options.

Clone workspace plugin:

To copy a workspace of one job to other location by archiving or as it is.

Next Build number plugin:

To change the build numbers as required.

Build metrics plugin:

Will give different options to search for builds like, how many days builds, Job filters like failure and success builds, jobs ran on each node.

Build Pipeline plugin:

This plugin will show the dependency jobs (Child job depends on Parent job) in a separate view.

If Job1 depends on Job2 and Job2 on Job3, 3 on 4. Then it will show the Jobs execution flow one by one.

For this we need to create a View with some name and then Select Pipeline view option and there if you select the top parent job(Job1), then it will show the dependency projects in a pipeline view with parent to downstream projects.

It shows the job execution status with colors. green: success, Blue: In-progress, red: Failure

Backup plugin:

To take backup of .jenkins file periodically in archived or normal format.

role based authorization strategy plugin:

Download and install this plugin from Manage Plugins option, then create few users.

Then go to Manage Jenkins-->Manage and assign roles.

Then create Global, Project and Slave related roles like admin, employee. Then enable the check boxes for required permissions for each role.

Then, go to Assign Roles,

then assign roles to each user.

Catlight plugin:

Catlight is the app we download and install on our local system.
We configure our Jenkins to catlight. So that catlight will keep getting desktop notifications about Jenkins build status like success, failure, stable etc.

We can do configurations in catlight as we require.