**JENKINS**

Jenkins is an open source automation tool written in Java programming language that allows continuous integration.

**Jenkins Architecture**

Here's how Jenkins elements are put together and interact:

* Developers commit changes to the source code, found in the repository.
* The Jenkins CI server checks the repository at regular intervals and pulls any newly available code.
* The Build Server builds the code into an executable file. In case the build fails, feedback is sent to the developers.
* Jenkins deploys the build application to the test server. If the test fails, the developers are alerted.
* If the code is error-free, the tested application is deployed on the production server.

The files can contain different code and be very large, requiring multiple builds. However, a single Jenkins server cannot handle multiple files and builds simultaneously; for that, a distributed Jenkins architecture is necessary.

## Jenkins Applications

Jenkins helps to automate and accelerate the software development process. Here are some of the most common applications of Jenkins:

### **1. Increased Code Coverage**

Code coverage is determined by the number of lines of code a component has and how many of them get executed. Jenkins increases code coverage which ultimately promotes a transparent development process among the team members.

### **2. No Broken Code**

Jenkins ensures that the code is good and tested well through continuous integration. The final code is merged only when all the tests are successful. This makes sure that no broken code is shipped into production.

**CI &CD With Jenkins-Gradle-Artifactory**

Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly."

**CI / CD step by step**

• Developers check out code into their local machines. • Developer commit changes to the source code repository. • The CI server monitors the source code repository and starts build process. • The CI server builds the system and runs unit and integration tests. • The CI server releases deployable artifacts for testing. • The CI server assigns a build label to the version of the code it just built. • The CI server informs the team of the successful build. • If the build or tests fail, the CI server alerts the team. • The team fix the issue at the earliest opportunity. • Continue to continually integrate and test throughout the project • On Approval, CI server deploys to test environments. • CI Server runs functional / performance tests on Test environment. • CI Server stage / Deploys to Production Environments (Can set up email based approval).

Jenkins Workflow Each Jenkins job executes one of the commands available via Gradle gradle compile gradle build gradle test

**Build CI & CD Pipeline with Jenkins**

Introduction

In software development life-cycle, DevOps is a set of practices which joins Development and IT Operations together and form an ideal set of practices which can be followed for smooth software development. When these practices are followed a set of tools used and referred as ‘toolchain’ which includes:

Coding — It involves coding and review and uses Source Code Management (SCM) tools like Git, GitHub, BitBucket, Subversion

Building — It is called Continuous Integration (CI) and involves CI tools and Build tools like maven, gradle, ant

Testing — It is called Continuous Testing (CT) and involves testing tools like Junit, JaCoCo etc. It evaluates the risks and report to owner based upon test cases. Mostly, it is considered part of CI from automation perspective

Packaging — It involves assembling of code and creating artifacts, it also comes under CI umbrella

Releasing — This release portion involves configuration management, release approvals and deployment. It is often considered under Continuous Deployment (CD)

Configuring — It involves infrastructure configuration and management

Monitoring — It is the last tool of toolset which involves liveness and performance monitoring

DevOps always encourage for maximum automation. In above toolset, CI and CD portion are mostly targeted for fully-automation. This automation of CI & CD enables, development environment very smooth, developers and product owner focus on actual business features instead of general aspects and CI & CD process.

This CI & CD involves various tools and steps which needs to be automated and the result is often termed as CI&CD Automation Pipeline, reason these steps are interlinked with each other and each step must be performed after success of previous steps.

To automated CI & CD pipeline, there are several tools and applications available from different vendors like Jenkins, JFrong Pipelines, Atlassian Bamboo, AWS CodePipeline, Azure DevOps, JenkinsX etc.

Jenkins is one of the most popular and widely used tool for CI & CD pipeline automation. Let’s understand Jenkins in following sections.

About Jenkins

Jenkins is most popular automation server used for setting up development environment with Continuous Integration & Continuous Deployment (CI & CD) automation which covers development processes like building, testing, vulnerability analysis and deployment etc.

It provides a complete web-based platform which includes a web application, work agents, build tools, Cloud, source code management and so on.

It was originally developed with name Hudson on the work by Kohsuke Kawaguchi in Sun Microsystems which later acquired by Oracle and it becomes licensed. Later on, a fork of Hudson created named Jenkins as open source project.

Below diagram depicting Jenkins components:

Figure 1: Jenkins internal components and design

The Jenkins has web-based user interface which running into the web server as part of Jenkins application. It serves various tools and configurations as required. Jenkins provides hundreds of plugins to be used which can be installed in Jenkins for different kinds of support and integrations. Also, these plugins can be developed by anyone based upon need to enhance Jenkins capability.

By default, Jenkins runs on Embedded H2 database, which can be changed to other database. It has its own user and access management backed with database and it also supports LDAP based authentication.

The Jenkins application runs with a node master node. Apart from master nodes, other nodes can be created and connected with Jenkins and jobs are delegated to worker nodes instead of running on master node. The nodes can be created with different infrastructure e.g. web-socket based node (it connects with an executable which is referred as Agent) with Java agent, docker container agent and SSH based agent.

A new version of Jenkins which is specific for Cloud and Kubernetes integration released with name JenkinsX.

Jenkins automation is completely based upon Jobs. A Job can be defined as setup activities declared as Job body and will be executed to achieve the automation.

Jenkins supports wide range of applications which are used while development, testing and deployment. Some of them are:

Build tools: Ant, Maven, Gradle

Test & Report tools: Junit, JaCoCo, Cobertura, SonarQube

OS & Platform: Windows, Linux, Mac, Docker, Kubernetes

SCM tools: Git, GitHub, BitBucket, Subversion

Cloud: Digital Ocean, AWS, Azure, OCI etc.

Let’s understand a Continuous Integration (CI) automation for a Java & Maven based project:

Java based project requires steps to be executed to perform as part of CI automation like: Code Checkout, Compile, Build, Test, Package, Publish artifacts.

These steps defined, must be executed sequentially at each time when code change happens

If any step fails, the sub-sequent steps will be aborted e.g. if compilation fails then build, test, package etc. will be aborted and reported to owner about compilation failure

These steps form a Job in Jenkins and whenever code change happen / required Job can be triggered to perform the steps. One execution of steps / Job trigger referred as a Build in Jenkins.

Apart from Job creation, there are several ancillary features are provided to support the Job management like:

User and access management to control the Jenkins and Jobs access

Folder and View creation to arrange Jobs

Build, Job Definition and execution history

Jenkins plugins to enhance capabilities of Jenkins

These Job purpose, scope, features and other capabilities depends upon the Plugins and its version. These Job have some common features like:

Job Name, Description, Display Name fields

Enable project-based security — It depends upon security strategy plugins

Discard old builds — Defines policy how to discard the old build e.g. discard build older than 5 days, keep data only for last 10 builds etc.

GitHub project — Configure GitHub for SCM purpose

This project is parameterized — Parameters can be defined which further used in Job build steps

Throttle builds — Define policy for throttle the builds e.g. parallel build can be trigger

Disable this project — Disable the project

Restrict where this project can be run — Define where build can be executed based upon labels which mapped with Node label

Quiet period — Define quite period in seconds

Retry Count — Retry count for SCM

Block build when upstream project is building — Block builds when upstream project is building

Block build when downstream project is building — Block builds when downstream project is building

Use custom workspace — Custom directory can be defined as workspace

Trigger builds remotely (e.g., from scripts) — It creates a link which can be used to trigger build

Build after other projects are built — It defines build dependency with other Job in Jenkins

Build periodically — Trigger build based upon expression like cron

Poll SCM — Define the polling of SCM to check for update which helps detecting changes at SCM and build can be triggered

Provide Configuration files — Profile configuration file

Generate Release Notes — A link with JIRA for release note

Apart from these common features, based-on Job type ‘Build’ and ‘Post Build Actions’ are available. There are many types of Job based upon the nature and usability of Job e.g.

Freestyle Project

This type can be used to create a free-style project and multiple build steps can be created by adding steps from UI from defined list of activities e.g. Execute Shellscript, Gradle, Ant, Maven, Windows Batch Command etc. It can be used for any type of project where steps can be defined with available execution steps.

Maven Project

Maven project is recommended for Java based project which uses Maven as build tool. In Maven based Job, there is provision to define pom.xml and build goals like clean, compile, test, package, install, deploy etc.

Pipeline Projects

This type of Job is based upon pipeline script (earlier known as workflow) which is used create complex pipeline using groovy sandbox. From 2016, this has been enabled by default from Jenkins for creating pipeline.

It supports two types of script structures based upon programming approach:

Scripted — It is based upon imperative programming model. It begins with node keywords and use groovy code and references for Jenkins pipeline. It does not need steps with the stages.

Declarative — It is based upon declarative programming model, where stages breaks down into steps and starts with pipeline keyword.

From ability and performance angle, there is no different only difference is way of writing. If there is question to choose approach, declarative must be preferred as it provides much declarative ways and option to add multiple steps.

Additionally, it supports two ways providing pipeline:

Provide pipeline code in Jenkins Job

Create and file named Jenkins and add pipeline code and commit in SCM

Multi-configuration Project

It is based upon freestyle project with additional features like ‘Configuration Matrix’ where user defined axis can be created.

Multibranch Pipeline

It is again based upon groovy based pipeline code. Additional feature is, it connects with SCM, get list of branches and run build on each branch based change detected.

Follow the steps given to setup Jenkins on local from ‘Setup Jenkins’ section.

Setup Jenkins

Jenkins binary and deployment file available in formats supported for a range of operating systems and platforms. Some or the deployable deliveries are:

Generic Java based Web Application Archive (WAR) — It can be run on any platform where compatible JRE is available

Ubuntu/Debian

CentOS/Fedora/RedHat

Windows

openSUSE

FreeBSD

Gentoo

macOS

OpenBSD

Cloud Versions e.g. Jenkins Azure, Jenkins AWS, Jenkins on GKE, Jenkins IBM Cloud, Jenkins on OCI, Jenkins One-Click on Civo Kubernetes, Jenkins Bitnami Image

All Jenkins deployments are very simple and easy to install. For testing purpose, let’s try generic Java based WAR file.

Prerequisites:

Install JDK 11 (Oracle or OpenJDK)

Maven 3.6.x — [Optional] if pipeline has to use

Git Console — [Optional] if pipeline has to use git manually or from SCM management

Ant — [Optional] if pipeline uses Ant

Gradle — [Optional] if pipeline uses Gradle

NodeJS and Npm — [Optional] if pipeline uses

Angular CLI — [Optional] if pipeline uses

DotNet SDK — [Optional] if pipeline uses

Docker CLI — [Optional] if pipeline use

Kubernetes CLI — [Optional] if pipeline uses

Based upon different project and pipeline need, additional SDKs, Build Tools, Integrators can be installed on machine.

Steps to setup on local:

Download the war binary from link: https://www.jenkins.io/download/. Once downloaded, a war file will be available

Create a directory which will be used has Jenkins home and set path as

set JENKINS\_HOME=c:/JenkinsHomeDirectory

Note: This is optional, if home path not setup, it will create a .jenkins directory in user’s home directory.

Start war file by running below command

java -jar jenkins.war.java -httpPort=8080

It will start Jenkins on port 9090

Launch browser (e.g. Chrome) and open URL as http://localhost:8080/ . It will take couple of minutes to start. After starting it will prompt for default password which is generated in home directory.

Open Jenkins home directory -> secret and open file named ‘initialAdminPassword’ for initial password. Copy the content and paste into input box in browser which seeks for initial password.

It will open Customize Jenkins, click ‘Select plugins to install’

It will open window to select the plugins and click on Install button

Configure Jenkins

Jenkins configuration is deep and subject to the plugins installed. Configurations are categorized into different categories and based upon need it can be customized. Open Jenkins Home -> Manage Jenkins, it will list configuration categories as:

There are some important configurations which are often used e.g.

Managing user: To create/manage users, open Jenkins Home -> Manage Jenkins -> Manage Users. It has ‘create user’ option for creating users. Existing users can be modified by clicking on User and Configure button:

Configure Global Security: To configure user repository, user and access mapping and other security configurations. Open Jenkins Home -> Manage Jenkins -> Configure Global Security and configure as:

Manage Credential: To manage credentials which are used for integration with other applications e.g. SonarQube, GitHub etc. a credential manager is provided by Jenkins:

Configure Nodes: It is recommended to keep worker nodes separately from Jenkins web application and therefore need to configured multiple worker nodes and link with Jenkins. To create and link nodes, need to perform below steps.

Open Jenkins Home -> Manage Jenkins -> Manage Nodes and Clouds. It provides menus for creating Node, Configure Clouds and Monitor Nodes.

To create a Node, click on ‘New Node’

Provide necessary details like remote directory, work directory, launch method and other details as per screen grab and save it:

Now, Node with name ‘WinNode1’ created.

Open ‘WinNode1’ by clicking on it but it will show offline (notice cross mark in red color in front of WinNode1).

Click on agent.jar, it will download a Java based executable.

Place this Jar file into the working node (another machine which has to be used as worker). And copy run command from screen and use it to execute worker node agent.

Note: Change the URL so that it can target correctly to the machine which runs the Jenkins. Mostly hostname or IP is used.

Start the node with command like:

Now the red icon has been removed in Jenkins Node:

Manage Plugins: Mange Plugins is used to install / uninstall the plugins. It downloads plugins from Internet hosted plugin repository.

Use Cases

Java and Maven Project CI

Use sample project from GitHub: https://github.com/siddhivinayak-sk/jwt-openid-oauth2.0-keycloak-kerberos-ntlm.git

Login in Jenkins and click on New Items, provide name and select ‘Maven Project’. It will draw a page for the Job pipeline with multiple tabs and fields in it which need to be configured for creating build pipeline.

On General Tab, provide description

Click on ‘Source Code Management’ tab and provide information of SCM. In this case, let’s use Git URL [Note: Credential is optional as it is public repository]. Define the branch as well

Click on Build tab and provide root POM and Goal option

Click on Save button. At this step, a Maven based pipeline for Java Project has been created and listed on Jenkins Home. [Note: JDK and Maven must be installed and configured in Manage Jenkins -> Global Tool Configuration]

Invoke Build Pipeline

Open Job by clicking on it. It will show available menus where there is button ‘Build Now’. Click on ‘Build Now’, it will start execution of pipeline.

Once, execution started, it will be visualized under the build history as running process for build number

The build process and console log can also be visualized. For this, click on Build number it will open build details where is button named ‘Console Output’. Click on it, it will open console output for the running build

Java Project with Groovy based Pipeline

Use sample project from GitHub: https://github.com/siddhivinayak-sk/jwt-openid-oauth2.0-keycloak-kerberos-ntlm.git

Open Jenkins and click on New Item, provide the name, select Pipeline and click on OK. It will open page for Job definition with multiple tabs

In General tab, provide description for the Job

Click on Pipeline tab, in definition select ‘Pipeline script from SCM’. Select Git as SCM and provide repository URL and branch name for the SCM. In Script-path provide path of Jenkins file form SCM.

Save the pipeline. It will create pipeline based upon groovy code. [Note: The pipeline code has been written into a file named Jenkins and stored in SCM. The script path field specifies the file which contains pipeline code. Also, this pipeline contains stages for docker based image creation along with Java tech stack therefore JDK, Maven, Docker must be available on machine where build is running.]

Node-Angular Project with Groovy based Pipeline

Use sample project from GitHub: https://github.com/siddhivinayak-sk/node-angular-front-end.git

To create CI and CD pipeline in Jenkins for Node project, need to follow the same steps which has been mentioned into ‘Java Project with Groovy based Pipeline’ as other portions must remain same only pipeline code changed which already changed and committed in SCM.

Additionally, for Node and Angular project, need to install the nodejs and ng for code compilation and build.

DotNet C# Project with Groovy based Pipeline

Use sample project from GitHub: https://github.com/siddhivinayak-sk/dotnetmicroservice.git

To create CI and CD pipeline in Jenkins for DotNet C# project, need to follow the same steps which has been mentioned into ‘Java Project with Groovy based Pipeline’ as other portions must remain same only pipeline code changed which already changed and committed in SCM.

Additionally, for Dotnet project compilation and build, docker platform has been used where a temporary container will be created and installed all required tools like SDK, build tool etc. and then the same environment is used to build the code.

Conclusion

CI & CD pipeline automation is essential part of ideal development environment which provides automation of DevOps processes to achieve effortless and faster development.

Jenkins is one of the most popular and widely used tool for CI & CD automation. It supports hundreds of plugins which enriches the features of Jenkins to support wide range of build tools, integration and deployment environment.

It is easy to setup and support scaling and creating complex project’s pipeline creation with groovy based declarative pipeline. These pipeline can be triggered based upon changes in SCM, periodically, by other dependent Jenkins Job and with web-hook from integrated applications like JIRA. It provides automated / single-click CI and CD till release without having much human effort and interaction.

Jenkins can be chosen for different kind of projects based upon the automation need.