JENKINS

DevOps Tool:

JENKINS(CI/CD)

Jenkins is an open-source automation server that is widely used for continuous integration (CI) and continuous delivery (CD) in software development. It helps developers automate various aspects of the software development lifecycle, such as building, testing, and deploying code changes to production environments. Jenkins is written in Java and is highly extensible through plugins, making it flexible and customizable for various workflows.

Key features and concepts of Jenkins include

- **CI/CD Pipelines:** Jenkins enables the creation of CI/CD pipelines, which are sequences of automated steps that define how code changes progress from development to production.
- **Plugins:** Jenkins has a rich ecosystem of plugins that extend its functionality.

- **Freestyle Projects:** Jenkins allows you to create freestyle projects, where you manually configure build and deployment steps. It provides flexibility but might require more manual setup.
- **Declarative Pipelines:** Jenkins supports declarative pipelines, where you define the pipeline using a domain-specific language (DSL) in a Jenkinsfile. Declarative pipelines offer a more structured and concise way to define complex workflows.
- **Distributed Builds:** Jenkins supports distributed builds, allowing you to distribute build jobs across multiple agents (build nodes) to leverage resources efficiently and reduce build times.
- **Integration with Version Control:** Jenkins can integrate with various version control systems like Git, Subversion, and Mercurial. It can automatically trigger builds when code changes are committed.
- **Security:** Jenkins provides options for securing access to the server, projects, and plugins. You can set up user

authentication, role-based access control (RBAC), and manage permissions for different users and groups.

- **Monitoring and Logging:** Jenkins offers built-in monitoring and logging capabilities to help track build status, view build logs, and monitor the performance of Jenkins itself.

Installation

Jenkins is written in Java so java installation is mandatory to run Jenkins.

Windows

- Check which Java is running on system: Open command prompt and run : java -version
- If its not present then install Java : Download windows pakcage from https://adoptium.net/

- Best practice to install package as a administrator. Click n java msi and install it. provide appropriate installation path while installing.
- Follow official page for Jenkins installation: https://www.jenkins.io/doc/book/installing/windows/
- Download Jenkins .msi from above page and start installation. — Select appropriate path for jenkins Home and select Java Home/Bin path(we installed in above step) validate port(Jenkins default port 8080) we can change it here or after installation also.

Linux/Unix

- Pre-requisite: If you are using Linux/Unix OS on your personal Laptop then well and good, if not then best solution is to get one instance on AWS account.
- Follow steps to get new machine: https://linux.how2shout.com/how-to-create-a-ubuntu-linux-aws-ec2-instance-on-amazon-cloud/
- Check which Java is running on system: Open command prompt and run : java -version

```
sudo apt-get update
```

```
sudo apt install openjdk-11-jdk
```

```
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc
https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
```

```
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]
https://pkg.jenkins.io/debian-stable binary/ | sudo tee
/etc/apt/sources.list.d/jenkins.list > /dev/null
```

```
sudo apt-get update
```

```
sudo apt-get install jenkins
```

```
## for reference installation steps follow below
```

```
https://www.cherryservers.com/blog/how-to-install-jenkins-on-ubuntu-22-04
```

```
https://www.jenkins.io/doc/book/installing/linux/
```

```
## After installation open Jenkins UI - http://<VM_IP>:8080
```

```
## login with first time user is "admin" and password is available here: /var/lib/jenkins/secrets/initialAdminPassword
```

```
## Download all default plugins
```

GIT Installation:

Windows/Linux

- Check which Git is running on system: Open command prompt and run : git — version
- Installation steps: https://git-scm.com/download/win
- For Linux: https://www.digitalocean.com/community/tutorials/how-to-install-git-on-ubuntu-20-04

Maven : Build Tool

Installation *Windows& Linux*

- Windows:

  https://www.javatpoint.com/how-to-install-maven
- Linux:

  https://linuxhint.com/install_apache_maven_ubuntu/

Types of Jobs

Jenkins offers several types of items (projects) that you can create when setting up a new job. These job types allow you to implement various build and automation workflows based on your project's requirements.

- **Freestyle project:** This is the traditional type of job in Jenkins, where you can configure the build steps, triggers, and post-build actions using a graphical user interface. It is suitable for simple and straightforward build processes.

- **Pipeline:** Jenkins Pipeline allows you to define your build process as code using a domain-specific language (DSL) based on Groovy. With pipelines, you can model complex build, test, and deployment workflows, as well as integrate with version control systems.
- **Multi-branch pipeline:** This type of job is an extension of the Pipeline project, optimized for multi-branch development workflows. It automatically detects branches in your version control repository and creates individual pipelines for each branch, making it ideal for continuous integration with multiple branches.
- **Folder:** A folder is a way to organize jobs hierarchically within Jenkins. You can create folders to group related jobs together, providing better organization and management of projects.
- **Multi-configuration project:** A Multi-configuration project, also known as a Matrix project, is a type of item in Jenkins that allows you to perform a build or test in multiple configurations simultaneously. It is useful when

you need to test your application or code against different combinations of platforms, operating systems, environments, or parameters.

General In Jenkins, the "General" options refer to the basic settings and configurations that apply to a Jenkins job (item) regardless of its type.

- **Discard Old Builds:** This option allows you to control the number of builds to keep. You can specify the maximum number of builds to retain, the maximum number of days to keep builds, or both. Older builds beyond the specified limits will be automatically deleted to save disk space.
- **GitHub Project:** If your Jenkins job is associated with a GitHub repository, you can specify the GitHub project's URL in this option. It helps Jenkins display build status and links back to GitHub.

- **This project is parameterized:** When enabled, this option allows you to add parameters to the job, enabling you to customize the build based on user input or other dynamic values.
- Jenkins supports several parameter types, including:
- String Parameter: Allows you to define a simple text parameter.
- Choice Parameter: Presents a dropdown menu with predefined choices.
- Boolean Parameter: Represents a true/false or yes/no parameter.
- File Parameter: Enables users to upload a file when triggering the job.
- **Throttle builds** Throttle Builds is a Jenkins plugin that allows you to limit the number of concurrent builds for specific jobs or for jobs within certain categories. This plugin is particularly useful when you have limited resources, such as build agents or hardware capacity, and

you want to control how many builds of a particular job or group of jobs can run simultaneously.

- Concurrent Build: By default, Jenkins allows concurrent builds of the same job. You can set the maximum number of concurrent builds to control how many simultaneous builds are allowed.
- Quiet Period: The quiet period is the time delay between the moment Jenkins is triggered and when the build actually starts. It helps avoid excessive builds triggered by multiple changes in quick succession.

Sample Maven Project

- sample github repository : https://github.com/jenkins-docs/simple-java-maven-app.git
- This build required maven version 3.8.6 and above
- delete existing maven from system : sudo apt-get purge maven

```
wget <take latest build from official page>

tar -xvf apacheXXXX.tar.gz

sudo mv apacheXXXX /usr/share/maven

echo 'export PATH="$PATH:/usr/share/maven"' >> ~/.bashrc

echo 'export PATH="$PATH:/usr/share/maven/bin"' >> ~/.bashrc

source ~/.bashrc

On Jenkins - ManageJenkins - Tools - maven - install automatically - 3.9.3 vrsion

create new job - maven_project

      - Source code: git : <provide above sample repo>

      - Build steps : Shell : echo "Maven build"   and java -version

      - Add Build steps: Invoke top-level Maven targets - Goals - clean install
```

Demo: Parameterized Job

Create a freestyle job using build parameters - string,multi-line string, password, choice, credentials,Boolean, file parameters

This project is parameterized

string parameter : Project default value : Moodys

Choice parameter: Shift_days Mon, Tue, Wed

Multi-line String Parameter About

Boolean Parameter : status

File Parameter : path

Credentials Parameter : credentials

Password Parameter : password

Build step : Execute Windows batch command

```
echo %Project
echo %Browser%
echo %About%
```

```
echo %status%

echo %path%

echo %password%

echo %credentials%
```

Demo: Build Trigger

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

:https://www.devopsschool.com/blog/how-to-trigger-builds-remotely-in-jenkins/

```
 - Go to configuration of loggend in user- Ex.admin

     - Create new API token, save it on notepad

     - Go to job configuration - Build trigger- Trigger builds remotely (e.g., from scripts)

     - Authentication Token  : provide any random value and save job

     - Open Git Bash - and execute

     - curl -X POST http://admin:115c8edaf670ddb9d4e104c9e47c9765d5@localhost:8080/job/first/build?token=abc123
```

Build after other projects are built

- Check -Build after other projects are built

- Add job name

Build periodically

- */2 * * * *

WebHook (This will not work on jenkins on local machine, need to use jenkins on aws cloud instance)

- Go to your GitHub repository and click on 'Settings'.

    - Click on Webhooks and then click on 'Add webhook'

    - In the 'Payload URL' field, paste your Jenkins environment URL. At the end of this URL add /github-webhook/. In the 'Content type' select: 'application/json' and leave the 'Secret' field empty.

    - In the page 'Which events would you like to trigger this webhook?' choose 'Let me select individual events.' Then, check 'Pull Requests' and 'Pushes'. At the end of this option, make sure that the 'Active' option is checked and click on 'Add webhook'.

```
    - In Jenkins Jobs - Build trigger - GitHub hook trigger for GITScm polling
```

Jenkins: Master-Slave architecture

Jenkins: Master-Slave architecture

Jenkins has powerful feature of master slave architecture which enables distributed builds. This article we will learn…

medium.com

Different between Freestyle and Pipeline

Freestyle Projects

- Configuration is done through a user-friendly interface,
- Supports integration with a wide range of Jenkins plugins to extend functionality.
- Each build step, such as compilation, testing, and deployment, needs to be configured individually.

- Ideal for simple build and deployment processes that don't involve complex branching, conditional logic, or intricate workflows.

Pipeline :

- Pipeline jobs use code to define the build, test, and deployment pipeline. This code can be versioned and stored in source control, providing better traceability and change management.
- There are two syntax options for defining pipeline jobs: Scripted and Declarative. The Declarative syntax aims to provide a more structured and readable way to define pipelines, while Scripted syntax allows for more customization and flexibility.
- Pipeline jobs are designed to handle complex workflows, including conditional logic, parallel stages, dynamic branching, and more.

- The pipeline script is stored in source control, allowing for versioning, collaboration, and code reviews.
- Reusability: Pipeline scripts can be reused across different jobs

Syntax for Scripted Pipeline

```
node {

    // Define environment variables or tools here



    stage('Stage 1') {

        // Actions for Stage 1

    }



    stage('Stage 2') {
```

```
        // Actions for Stage 2

    }

    // More stages and actions as needed

}
```

Syntax for Declarative Pipeline

```
pipeline {

    agent any

    environment {

        // Define environment variables
```

```
    }


    stages {

        stage('Stage 1') {

            steps {

                // Actions for Stage 1

            }

        }


        stage('Stage 2') {

            steps {

                // Actions for Stage 2

            }
```

```
        }

        // More stages as needed
    }

    post {
        // Post-build actions
    }
}
```

pipeline

key aspects of a Jenkinsfile

- Stages: Jenkinsfiles are organized into stages, each representing a logical part of the build and deployment process. Stages define the sequence of actions, such as building, testing, deploying, etc., that your pipeline should perform.
- Steps: Each stage is composed of one or more steps. Steps are individual tasks or actions that contribute to the execution of a stage. For example, a step could involve checking out code, compiling, running tests, and deploying.
- The agent directive specifies where the pipeline will run, such as on a specific node or a container. It defines the environment in which the pipeline steps will be executed.

multibranch Pipeline

A multibranch pipeline is a feature in Jenkins that enables you to automatically create and manage pipelines for multiple branches in a version control repository, such as Git. It's particularly useful in projects where you have multiple feature branches

- New item — provide name — select multibranch pipeline
- Branch source : GIT — https://github.com/sagarkulkarni1989/sharedlibs.git
- Credentials — Its public repo so not required
- discover branches — if you want to use filter to get banch then use appropriate option. In this example you can ignore
- Build configuration : It get default Jenkinsfile . change the Jenkinsfile directory location as per your requirement