# DevOps UNIT-4

- ⇒ Build Systems
- ⇒ Jenkins Build Server
- ⇒ Managing Build Dependencies
- ⇒ Jenkins plugins
- ⇒ file System layout
- ⇒ The Host Server
- ⇒ Build Slaves
- ⇒ Software on the Host
- ⇒ Triggers
- ⇒ Job chaining and Build pipelines
- ⇒ Build Server & Infrastructure as Code
- ⇒ Building by Dependency Order
- ⇒ Build phases
- ⇒ Alternative Build Servers
- ⇒ Collating Quality measures

# Build Systems

⇒ Build System is the Software that Can Compile and test Code. to Verify Code is working Correctly or not. and then it will package code. i.e. it will Create an executable file for deployment.

⇒ Many different build Systems have been Created for Software development. Sometimes it may feel like there are more build Systems than programming languages. Here are few examples:-

Java :- Uses Maven, Gradle and Ant

C and C++ :- Use Make (Various Versions)

Clojure :- Uses Leiningen, Boot and Maven

Java Script :- Uses Grunt

Scala :- Uses Sbt

Ruby :- Uses Rake

Shell Scripts :- Various kinds for different needs.

Depending on your Company and the type of Software you are building you might use any of these tools. Some Companies even Create their own build tools.

⇒ Normally many Companies will use only one Standard tool. but it is not efficient. For example, building JavaScript Code is easier with Grunt than with maven or make. So we need to use different tools based on our needs.

⇒ In practice, it's good if a developer can checkout and build the code on their local machine, without problems.

⇒ If you have to support multiple build systems, you can combine them. this makes things simpler and allows different build systems to work together. For example, a Java Organization might always use the command 'mvn clean install' to build their projects.

↳ this command compiles java code, runs the tests, and packages the software and save it in your system.

## Jenkins Build Server

⇒ Build Server is a machine that runs the build process automatically. Build Server uses the build system (software) to compile code, run tests and create executable files. One popular build server is Jenkins, which is written in Java.

⇒ Jenkins began as a project named Hudson. When oracle took over Hudson, the original creator, Kohsuke Kawaguchi, kept working on jenkins as a different version. Today, Jenkins is more popular than Hudson.

⇒ Jenkins is famous for building Java programs. it can also build software made with other programming languages

⇒ Setting up Jenkins is not too difficult. For example, on ubantu Linux, you can install Jenkins using the command:

```
Sudo apt-get install jenkins
```

⇒ Start the jenkins service using command:-

```
Sudo Systemctl Start jenkins
```

⇒ Jenkins should now be running. You can access it through your web browser at url "http://localhost:8080"

⇒ to check status of Jenkins service, use command:

```
Sudo Systemctl Status jenkins
```

⇒ In jenkins, work is organized into "jobs". Jobs can perform various tasks like compiling code, running tests, or deploying applications. Jenkins helps automate these tasks, making it easier for developers to manage and troubleshoot their software projects.

⇒ Jenkins is especially useful for continuous integration and continuous delivery (CI/CD) pipelines, it ensures that software is quality and efficient.

# Managing Build Dependencies

⇒ Build Dependencies, in simple terms, refer to other Software Components or files that your application needs in order to successfully build and run your application.

⇒ Some build systems like maven for java and Grunt for JavaScript automatically create dependences (like pom.xml file) based on projects.

⇒ In C and C++ projects, tools like GNU Autotools and Auto Conf handle dependencies differently Compared to tools like maven. Instead of creating dependencies these tools will use dependencies that are already available.

⇒ In Software development, it is really important to know exactly what features your Software will have, if you dont know outcomes of your Software, it causes unexpected problems. For example you may miss features or functions when you deploy your Software to Servers that Customers use.

⇒ Systems like RPM (Red Hat Package manager) offer a Solution. when you upload your project into RPM, RPM will provide Specification file. this file will give you list of all the dependencies required for building and packing packaging your Software:

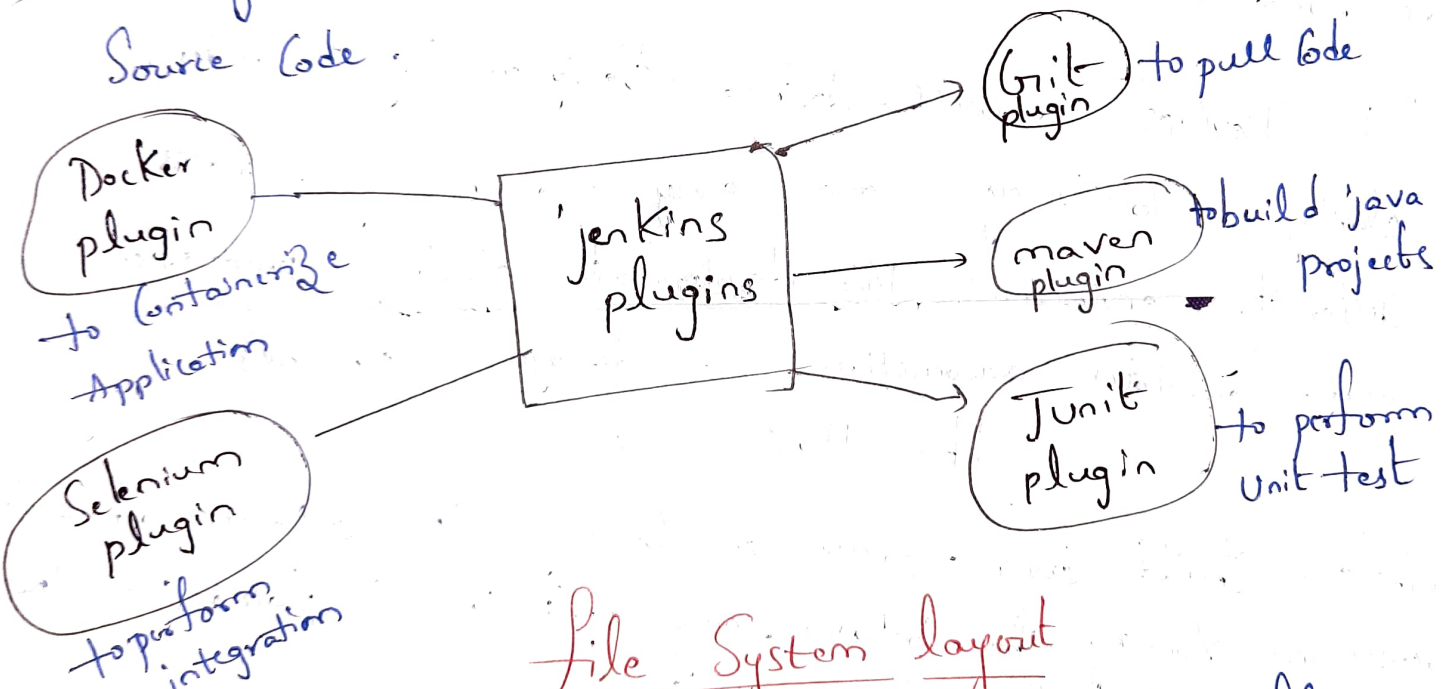
# Jenkins plugins

⇒ Jenkins uses plugins to add extra features to its build server.

⇒ there are many different plugins available and they can be installed directly from the jenkins web interface.

⇒ Some plugins can even be installed without restarting the jenkins.

⇒ Among other plugins, we need the git plugin to pull our source code.




# File System layout

⇒ In DevOps, the file system layout refers to how files and directories are organized on servers or systems and managed within a DevOps environment. This includes

① Application Deployment Structure :-

How applications and their dependencies are organized on servers or containers.

② Configuration management:

How Configuration files like YAML, JSON files etc are Structured and Stored.

③ Versim Control: (git)

How Code repositories are Organized,

④ Artifact Repositories:

How build artifacts are Stored in repositories.

⇒ In DevOps, a well-defined file system layout ensures clarity, consistency, and easy to manage across different environments like development, testing, production and facilitates automation and deployment processes

## The Host Server

⇒ while a build Server focuses on Creating the Software, a host Server focuses on running the Software. once application is placed in the host Server, end users can access it.

⇒ the build Server is a Very important Computer in an Organization. Building Software requires a lot of processing power, memory and diskspace. to ensure that builds are fast, the build Servers need to have good Specification with high disk Space and RAM.

⇒ Build Servers will Collect Code from different people. So build Servers must be fast. Since machines are cheaper than people, you Can invest your money on good machines.

# Build Slaves

⇒ To make build processes faster in Jenkins, you can add extra servers called build slaves. the main server (master) sends tasks to these slaves, either in a round-robin way or based on specific needs. Some tasks need particular operating systems, so having different build slaves helps. For example, you might have a Linux master server but need windows or Mac slaves for certain tasks.

⇒ Using build slaves lets you run multiple builds at once and build software on different systems. For instance, Apple's rules make it tricky to use virtual servers for building Mac software, so a mac build slave is helpful.

⇒ there are several ways to connect build slaves to the master. one common method is using SSH, and Jenkins has built-in support for it. Another method involves downloading a Java client (JNLP) from the master to slave, which is useful if the slave doesn't support SSH.

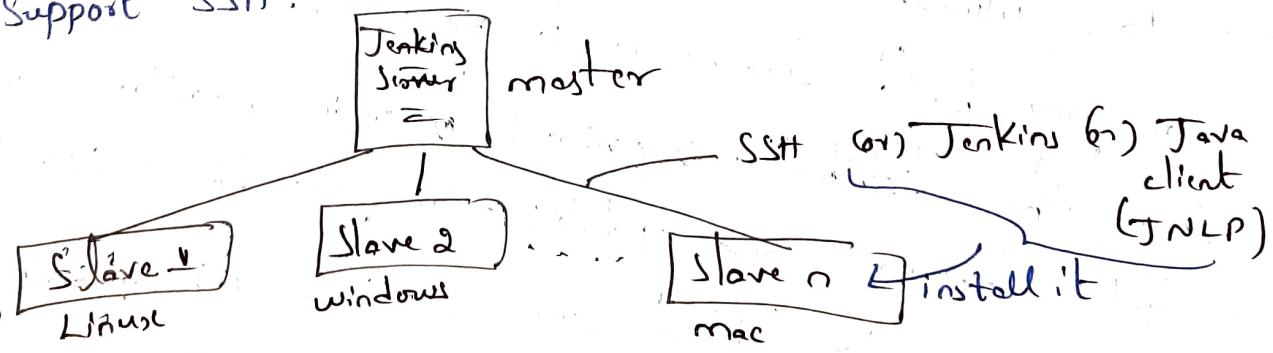



fig: master-slave Architecture

⇒ Sometimes, it's easier to use a Linux System to build windows Software Using Cross-Compilers like GCC with MinGW. this depends on the Specific Software we build.

⇒ Companies like teleComs and banks might use a lot of System-Specific Code for better performance. this allows you to build any type of application without depending on Separate machines.

⇒ Each Company has Specific needs, So you need to Set up your build environment to match those needs.

## Software on the Host

⇒ To build Software, you might need to install different types of build tools on your build Server. Jenkins is mostly used to Start builds, but not do the actual building. the actual build work is done by tools like maven or make.

⇒ It's usually best to use a linux based Operating System for your build Server. most build tools can be easily installed from the distribution repositories (an online repositories where you can install Software and updates for your Linux Operating System).

⇒ you can update build Server, Same like updating the application Servers.

# Triggers

=> A trigger is like a Start button that begins an action automatically based on certain conditions or events.

① **Scheduled Builds** :-
you can set a timer to run builds at specific times, like every hour or every day. this ensures builds happen regularly.

② **Polling for changes** :-
Jenkins can check your code repository like Git for updates. if changes are found, Jenkins starts a build automatically. this keeps your builds up-to-date with the latest code.

③ **Nightly Builds** :
these are scheduled builds that run at night, usually when no one is actively working. these night builds take long time, so employees will check it in next working day.

④ **upStream and DownStream Builds** :
when one job finishes successfully (upStream build), it can trigger another job (down stream build).

=> Using these triggers helps automate the build process, ensuring your software is always built and tested efficiently.

# Job changing and Build pipelines

In Jenkins, job chaining means Connecting jobs so that when one job finishes Successfully, it automatically Starts another job. This Creates a Sequence where each job depends on the previous one Completing without any issues. The first job in this Sequence is Called the upStream build, and the next one is the downStream build.

⇒ while basic job chaining is useful for many tasks, Sometimes you need more Complex workflows, Known as pipelines. Jenkins has plugins like the workflow plugin, which lets you Create advanced pipelines Using a Scripting language Called Groovy.

⇒ these pipelines are helpful because they allow you to See and Control each step of your build process more clearly. they ensure tasks happen in the Correct order and easy to manage and automate deployment tasks

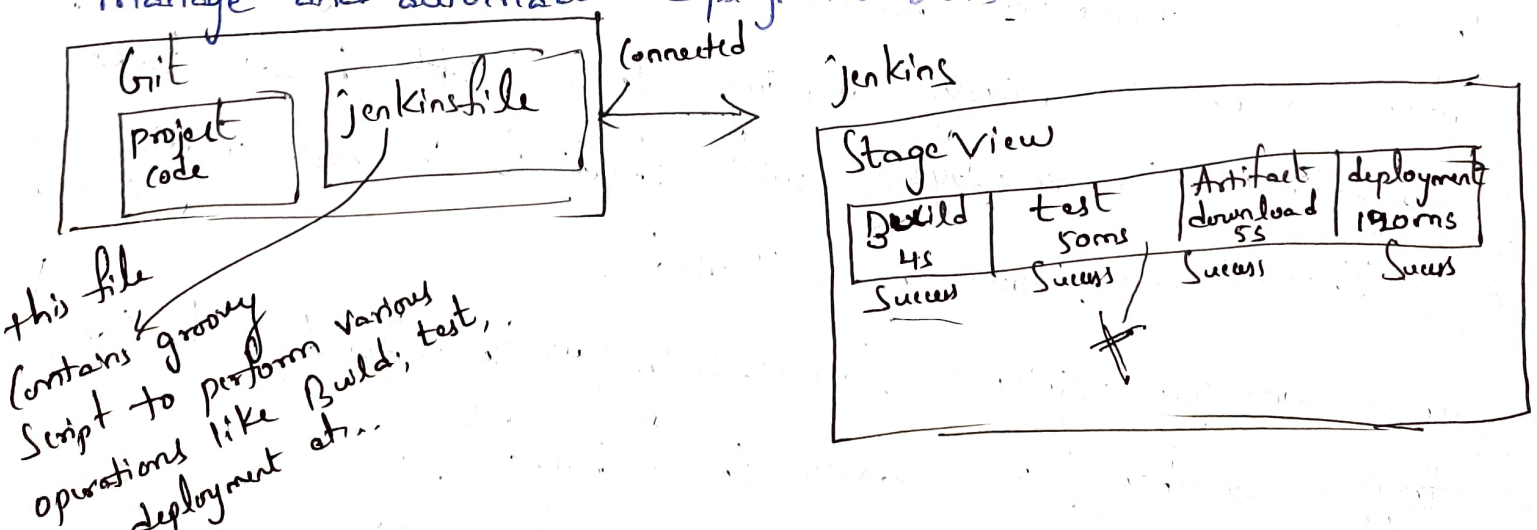



fig: Stage view in jenkins

# Build Servers and Infra Structure as Code

⇒ When we talk about DevOps, we can manage our servers and setups using code, same like writing code to create software.

## Problem with Jenkins :-

### (1) Jenkins interface :-

#### Text files Vs web interface :-

Jenkins uses text files to setup tasks, but most people use jenkins web interface to make changes.

Good Side :- Easy to use without much knowledge

Bad Side :- missing some features that programmers usually like, such as code reuseblity.

## Alternative - Gitlab :-

Gitlab build tasks using code for the beginners. So this is like programming and easier for developers to use. Using Gitlab you can handle build tasks easily. if you dont need all the advantages and features of jenkins. GitLab might be simpler and more friendly to use.

# Building by Dependency Order

⇒ when building Software, Some parts need to be built before others because they depends on each other. you Can do it by using Various build tools.

## make tool :-

⇒ in tools like make, you clearly State the order of building.

Example:

```
a.out : b.o c.o

b.o : b.c

c.o : c.c
```

this means 'a.out' depends on 'b.o' and 'c.o', So 'b.o' and 'c.o' must be build first.

## Maven and Gradle :-

Tools like maven and Gradle also follows build order based dependencies you define.

⇒ Maven uses a build graph Called the "reactor" to manage the order.

⇒ Gradle also Creates a build Graph before Starting the build.

## Jenkins :-

Jenkins Can show the build order (reactor) for maven projects in its website.

for example, In Software building, you need to follow order like Compile Code part A and part B, Combine part A and part B, Run testing, Generate Artifact.

# Build phases

⇒ Maven is a build tool that makes sure everyone follows the same steps when building software. this is helpful for big companies because they don't need to create their own rules for building projects.

Maven Vs other tools:

Maven very strictly follow order of steps, like building, testing and deployment. whereas other tool like Ant give more freedom but it can lead to disorganized build.

Build phases:

(1) Building :- Compiling the code.

(2) Testing :- Running tests to catch errors.

(3) Deploying :- moving the finished product to server.

maven will follow this strict order but other tools may skip some steps.

⇒ Testing is very important. A good Continuous Integration (CI) Server will catch errors by running automated tests.

⇒ maven strictly follows all these steps without skipping any steps like testing.

# Alternative Build Servers

Jenkins is a popular tool for building and testing software. but there are other options too.

Travis CI :- popular for open source projects.

Buildbot :- written in python

Go Server :- Developed by Thought works Company

Bamboo :- offered by Atlassian Company.

GitLab :- Also have build Server features.

⇒ There are many build Servers available, and its important to find which works best for your team.

## Collating Quality measures

↳ mean collecting

⇒ A build Server can help track and improve Software quality by providing testing results.

⇒ Jenkins can run Java Unit tests and show results directly on the job page. Sonar is a tool that provides detailed code quality reports. Sonar runs tests during the build process and Sends the results to a Sonar Server. Developers can see detailed visual reports of code quality and improvements.

⇒ Running Sonar tests can make builds slower. Run Sonartests during night to avoid slowing down regular build.

⇒ Using tools like Jenkins and Sonar can help monitor and improve code quality, but its best to run tests like Sonar at night to avoid slowing down regular builds.