**Jenkins**

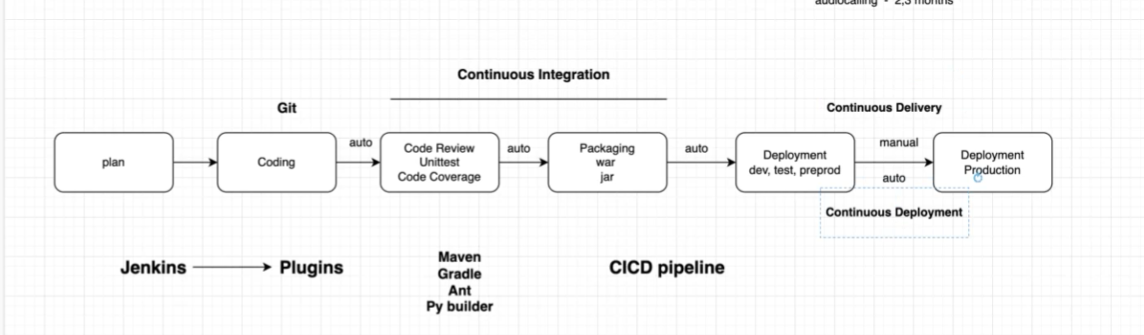
https://www.microsoft.com/en-us/learning/dashboard.aspx

Microsoft exam:--

https://esi.microsoft.com/getcertification

https://www.microsoft.com/en-us/learning/dashboard.aspx - dashboard

https://wsr.pearsonvue.com/testtaker/registration/Dashboard/MICROSOFT



Integration(building your application): : Maven , Gradle, python builder ,Ant

continuous (automation) : :Jenkins

{ Plan --> coding(git) -->continuous Integration[code review,compiler ,unittest,code coverage --> packaging war/jar] --> continuous delivery[Deployement dev ,test,preprod,prod] } -->CICD Pipeline ,if pipeline is block from any part it will not proceed to further steps.each and every things need to done in steps correctly.

earlier process--> we used to work in single integration in multiple months. Now we are having multiple integration in single months. Every single feature is going to be

complied ,review ,Tested ,covered and packaged. Every single minor is going to compiled and reviewed.

>jenkins is easy to install

>Jenkins has good operating support (ubuntu/linux/windows)

>Multiple tool integrated in jenkins as a plugin.

**Cd /var/lib/Jenkins** 🡪 you will get all the details about Jenkins under this path .ex – usercontain,secret,nodes, etc.

>If you want to build your project with maven tool .maven will ask for POM.xml (project object model) file. This is the file which contain all your instruction.

Jenkins port : 8080

Public IP:8080

While doing the set we replace the public Ip with private because when ever we reboot or poweroff the server the public IP will change.

Plugins –

1.Compile -> HLL to LLL -> It is the stage where we convert high level language (JAVA,Python etc) to Low level language (Binary).so computer can understand the code language.

2.Code Review -> suppose we have written a code in JAVA language and there can be some syntaxing issue like " ,' ,[ , etc.. .we do the checks related to syntax check in this stage.

3.Unit Test -> we have application code written and want to

Code testing. need to check the logical test menas the code the written logically test. we called it as unit test because the code testing done piece by piece.

4.code coverage -> how much actually tested during unit test.

building of application/artifact creation /packaging means - extention provided for your application

.dmg- mac OS , .exe -window ,.war -need tomcat software to be install ,.ppk -mobile application

Build : In Goals: to do code review

pmd (program mistake ditector) - This is just a syntaxing need to follow it. (-P metrics pmd:pmd )

**We need to install the plugin to get the output in GUI . So select the plugin in "post build action" to see better view.**

>warnings next generation

>cobertura

>build pipeline

**General Job vs Scripted Pipeline Job**

Main difference

The main difference between any job and a Pipeline Job is that the Pipeline Scripted job runs on the Jenkins master, using a lightweight executor expected to use very few resources in order to translate the pipeline to atomic commands that execute or send to the agents.

In our example, in the Freestyle job everything is executed in the agent, but for the Scripted Pipeline Job, the pipeline code is translated in the master to atomic commands that are sent to the agents.

Scripted Pipeline Job vs Declarative Pipeline

**Main difference:** There are several differences:

You need to use a Jenkins file to define it that have to be located in an SCM.

By default, each step is going to do a “checkout SCM” on each “node”.

In our example, if the Master doesn’t have access to the SCM it couldn’t work because the first step is that the Master get the Jenkins file from the repository and translate the steps in atomic commands.

One of our nodes doesn’t have access to the SCM so by default it’ll fail because each “node” do a “checkout scm” perhaps it doesn’t need anything from there.

To avoid that you need to add this piece of code:

options {

skipDefaultCheckout()

}

* free style ->The purpose of the freestyle project is to implement, develop, or run simple jobs like allowing you to specify the version control system from which you required to extract code and build it and call tests if available. Freestyle projects are meant to orchestrate simple jobs for a project.
* maven ->Maven build for java coding
* pipeline ->Pipeline Project is a new type of Jenkins project that is suitable either when you have to set up a continuous delivery pipeline or to define the deployment pipeline as code. The pipeline project is applicable to build pipelines for complex activities that are not suitable for freestyle project.
* Build Queu and Build executer status -->when we started more then 2 or 3 job at a time. the 3rd job will got build queue.

so for managing the no of executer we have to change the setting from manage jenkins ->manage nodes and clouds -> build in node (i.e master node)-> we can change the no of executer from here.

To add a new Node ,we need to make a changes in setting--> go to manage Jenkins -> configure global security --> Agent --> make it "Random”. If it is disable we can not add any worker node.

> Now create a new instance for worker node.

>Now connect to your worker node

>Add a new node --> give the name for worker node --> no of executer

>remote root executer -->here we have tell the directory where the work space will create.

>create a directory in worker node first : mkdir project

cd project/

copy the path and keep in the remote root executer.

>labels - Node 1

>launch --> launch agent by connecting it to the controller(master).

<https://github.com/lerndevops/labs> - ansible

**Jenkins security : ( configure global security)**

Authenticationwho are you? means what is the user name , id ,password ,certificates ,token.

Autherisation:what can we do? means what all the access you have to perform any task.

If we forgot the jenkins dashboard password.

for that we have to go to jenkins ec2 instance server --> got to cd /var/lib/jenkins --> ls -ltr --> config.xml

--> vi config.xml

--> modify the line " usesecurity to false "-->:wq!

restart the service

systemctl status jenkins

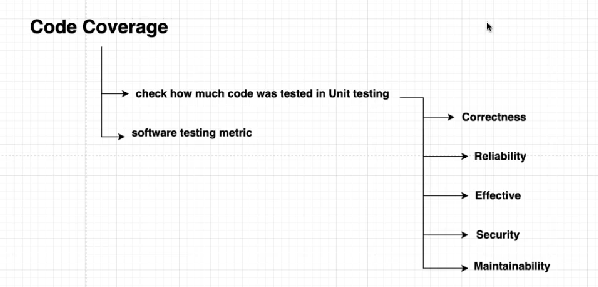
systemctl restart jenkins

and just dashboard now.

go to manage jenkins - configure global security -security realm -> select jenkins own user database - save

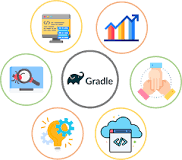
manage user -> create a admin user with all privilege.

And then go to Jenkins server and modify the line " use security to true "-->:wq!



Cobertura , jocoba coverage report – tool

What is Gradle tool?

[[](https://www.google.com/search?rlz=1C1GCEU_enIN987IN988&q=What+is+Gradle+tool?&tbm=isch&source=iu&ictx=1&vet=1&fir=CpCTLvf8ahcFVM%2CZokHHRfiYKkX3M%2C_&usg=AI4_-kRsL35I4Z1vZWTHs56czNF8LO28ng&sa=X&ved=2ahUKEwir46mSku72AhX0wjgGHQ92B8oQ9QF6BAgUEAE#imgrc=CpCTLvf8ahcFVM)](https://www.google.com/search?rlz=1C1GCEU_enIN987IN988&q=What+is+Gradle+tool?&tbm=isch&source=iu&ictx=1&vet=1&fir=CpCTLvf8ahcFVM%252CZokHHRfiYKkX3M%252C_&usg=AI4_-kRsL35I4Z1vZWTHs56czNF8LO28ng&sa=X&ved=2ahUKEwir46mSku72AhX0wjgGHQ92B8oQ9QF6BAgUEAE" \l "imgrc=CpCTLvf8ahcFVM)

[4 days ago](https://www.google.com/search?rlz=1C1GCEU_enIN987IN988&q=What+is+Gradle+tool?&tbm=isch&source=iu&ictx=1&vet=1&fir=CpCTLvf8ahcFVM%252CZokHHRfiYKkX3M%252C_&usg=AI4_-kRsL35I4Z1vZWTHs56czNF8LO28ng&sa=X&ved=2ahUKEwir46mSku72AhX0wjgGHQ92B8oQ9QF6BAgUEAE" \l "imgrc=CpCTLvf8ahcFVM)

Gradle is **a build automation tool known for its flexibility to build software**. A build automation tool is used to automate the creation of applications. The building process includes compiling, linking, and packaging the code.

**Question: What is Gradle?**

Gradle is an open-source build automation system that builds upon the concepts of Apache Ant and Apache Maven. Gradle has a proper programming language instead of XML configuration file and the language is called ‘Groovy’.

Gradle uses a directed acyclic graph (“DAG”) to determine the order in which tasks can be run.

Gradle was designed for multi-project builds, which can grow to be quite large. It supports incremental builds by intelligently determining which parts of the build tree are up to date, any task dependent only on those parts does not need to be re-executed.

To get in-depth knowledge on DevOps, you can enroll for live [DevOps online training](https://onlineitguru.com/devops-online-training-placement.html) by OnlineITGuru with 24/7 support and lifetime access

**What Are the Advantages of Gradle?**

Gradle provides many advantages and here is a list

Declarative Builds: Probably one of the biggest advantages of Gradle is the Groovy language. Gradle provides declarative language elements. Which provides build-by-convention support for Java, Groovy, Web, and Scala.

Structured Build: Gradle allows developers to apply common design principles to their build. It provides a perfect structure for a build so that well-structured and easily maintained, comprehensible build structures can be built.

Deep API: Using this API, developers can monitor and customize their configuration and execution behaviors.

Scalability: Gradle can easily increase productivity, from a simple and single project, builds to huge enterprise multi-project builds.

Multi-project builds: Gradle supports multi-project builds and also partial builds. Build management: Gradle supports different strategies to manage project dependencies.

First build integration tool − Gradle completely supports ANT tasks, Maven and levy repository infrastructure for publishing and retrieving dependencies. It also provides a converter for turning a Maven pom.xml to Gradle script.

Ease of migration: Gradle can easily adapt to any project structure.

Gradle Wrapper: Gradle Wrapper allows developers to execute Gradle builds on machines where Gradle is not installed. This is useful for continuous integration of servers.

Free open source − Gradle is an open source project and licensed under the Apache Software License (ASL).

Groovy: Gradle’s build scripts are written in Groovy, not XML. But unlike other approaches, this is not for simply exposing the raw scripting power of a dynamic language. The whole design of Gradle is oriented towards being used as a language, not as a rigid framework.

**Why Gradle Is Preferred Over Maven or Ant?**

There isn’t great support for multi-project builds in Ant and Maven. Developers end up doing a lot of coding to support multi-project builds.

Also having some build-by-convention is nice and makes build scripts more concise. With Maven, it takes to build by convention too far, and customizing your build process becomes a hack.

Maven also promotes every project publishing an artifact. Maven does not support subprojects to be built and versioned together.

But with Gradle developers can have the flexibility of Ant and build by a convention of Maven.

Groovy is easier and cleaner to code than XML. In Gradle, developers can define dependencies between projects on the local file system without the need to publish artifacts to the repository.

**Gradle Vs Maven?**

The following is a summary of the major differences between Gradle and Apache Maven:

Flexibility: Google chose Gradle as the official build tool for Android; not because build scripts are code, but because Gradle is modeled in a way that is extensible in the most fundamental ways.

Both Gradle and Maven provide convention over configuration. However, Maven provides a very rigid model that makes customization tedious and sometimes impossible.

While this can make it easier to understand any given Maven build, it also makes it unsuitable for many automation problems. Gradle, on the other hand, is built with an empowered and responsible user in mind.

Performance:

Both Gradle and Maven employ some form of parallel project building and parallel dependency resolution. The biggest differences are Gradle’s mechanisms for work avoidance and incrementally. Following features make Gradle much faster than Maven:

Incrementally: Gradle avoids work by tracking the input and output of tasks and only running what is necessary.

Build Cache: Reuses the build outputs of any other Gradle build with the same inputs.

Gradle Daemon: A long-lived process that keeps building information “hot” in memory.

User Experience:

Maven has very good support for various IDE’s. Gradle’s IDE support continues to improve quickly but is not great as of Maven.

Although IDEs are important, a large number of users prefer to execute build operations through a command-line interface. Gradle provides a modern CLI that has discoverability features like `Gradle tasks`, as well as improved logging and command-line completion.

Dependency Management:

Both build systems provide the built-in capability to resolve dependencies from configurable repositories. Both are able to cache dependencies locally and download them in parallel.

As a library consumer, Maven allows one to override a dependency, but only by version. Gradle provides customizable dependency selection and substitution rules that can be declared once and handle unwanted dependencies project-wide. This substitution mechanism enables Gradle to build multiple source projects together to create composite builds.

Maven has few, built-in dependency scopes, which forces awkward module architectures in common scenarios like using test fixtures or code generation. There is no separation between unit and integration tests, for example. Gradle allows custom dependency scopes, which provides better-modeled and faster builds.

**What are Gradle Build Scripts?**

Gradle builds a script file for handling projects and tasks. Every Gradle build represents one or more projects.

A project represents a library JAR or a web application.

**What is Gradle Wrapper?**

The wrapper is a batch script on Windows and a shell script for other operating systems. Gradle Wrapper is the preferred way of starting a Gradle build.

When a Gradle build is started via the wrapper, Gradle will automatically download and run the build.

**Question: What is Gradle Build Script File Name?**

This type of name is written in the format that is build.gradle. It generally configures the Gradle scripting language.

**How To Add Dependencies In Gradle?**

In order to make sure that dependency for your project is added, you need to mention the

configuration dependency like compiling the block dependencies of the build. gradle file.

**What Is Dependency Configuration?**

Dependency configuration comprises of the external dependency, which you need to install well and make sure the downloading is done from the web. There are some key features of this configuration which are:

1. Compilation: The project which you would be starting and working on the first needs to be well compiled and ensure that it is maintained in the good condition.

2. Runtime: It is the desired time which is required to get the work dependency in the form of collection.

3. Test Compile: The dependencies check source requires the collection to be made for running the project.

4. Test runtime: This is the final process which needs the checking to be done for running the test that is in a default manner considered to be the mode of runtime

**What Is Gradle Daemon?**

A daemon is a computer program that runs as a background process, rather than being under the direct control of an interactive user.

Gradle runs on the Java Virtual Machine (JVM) and uses several supporting libraries that require a non-trivial initialization time.

As a result, it can sometimes seem a little slow to start. The solution to this problem is the Gradle Daemon: a long-lived background process that executes your builds much more quickly than would otherwise be the case.

We accomplish this by avoiding the expensive bootstrapping process as well as leveraging caching, by keeping data about your project in memory. Running Gradle builds with the Daemon is no different than without

**What Is Dependency Management in Gradle?**

Software projects rarely work in isolation. In most cases, a project relies on reusable functionality in the form of libraries or is broken up into individual components to compose a modularized system.

Dependency management is a technique for declaring, resolving and using dependencies required by the project in an automated fashion.

Gradle has built-in support for dependency management and lives up the task of fulfilling typical scenarios encountered in modern software projects.

**What Are the Benefits Of Daemon in Gradle 3.0?**

Here are some of the benefits of Gradle daemon

1. It has good UX

2. It is very powerful

3. It is aware of the resource

4. It is well integrated with the Gradle Build scans

5. It has been defaulting enabled

**What Is Gradle Multi-Project Build?**

Multi-project builds helps with modularization. It allows a person to concentrate on one area of work in a larger project, while Gradle takes care of dependencies from other parts of the project

A multi-project build in Gradle consists of one root project and one or more subprojects that may also have subprojects.

While each subproject could configure itself in complete isolation of the other subprojects, it is common that subprojects share common traits.

It is then usually preferable to share configurations among projects, so the same configuration affects several subprojects.

**What Is Gradle Build Task?**

Gradle Build Tasks is made up of one or more projects and a project represents what is been done with Gradle.

Some key of features of Gradle Build Tasks are:

1. Task has life cycled methods [do first, do last]

2. Build Scripts are code

3. Default tasks like run, clean, etc

4. Task dependencies can be defined using properties like depends on

**What is Gradle Build Life Cycle?**

The Gradle Build life cycle consists of the following three steps

-Initialization phase: In this phase, the project layer or objects are organized

-Configuration phase: In this phase all the tasks are available for the current build and a dependency graph is created

-Execution phase: In this phase tasks are executed.

**What is Gradle Java Plugin?**

The Java plugin adds Java compilation along with testing and bundling capabilities to the project. It is introduced in the way of a SourceSet which act as a group of source files compiled and executed together.

**What is Dependency Configuration?**

A set of dependencies is termed as dependency configuration, which contains some external dependencies for download and installation.

Here are some key features of dependency configuration are:

Compile:

The project must be able to compile together

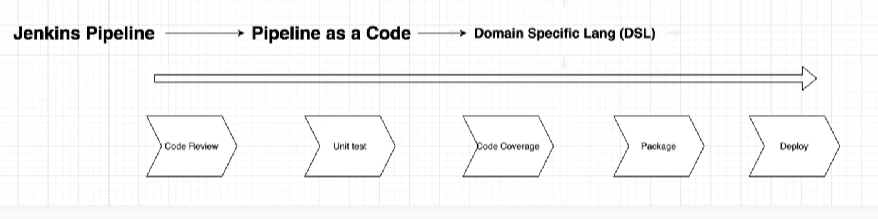
Runtime:

It is the required time needed to get the dependency work in the collection.

Test Compile:

The check source of the dependencies is to be collected in order to run the project.

Test Runtime: The final procedure is to check and run the test which is by default act as a runtime mode.



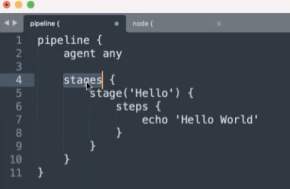
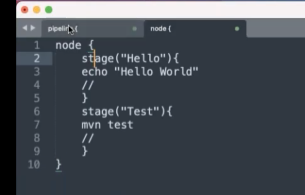
**Jenkins Pipeline is the series of action that will executed in a particular order.**

Pipeline as a code we can write in DSL language. Like YAML can be use as ansible , Docker, qubernetive, gitlab etc. But Jenkins can written in groovy language which can be written only in Jenkins only .that is why called it as DSL.

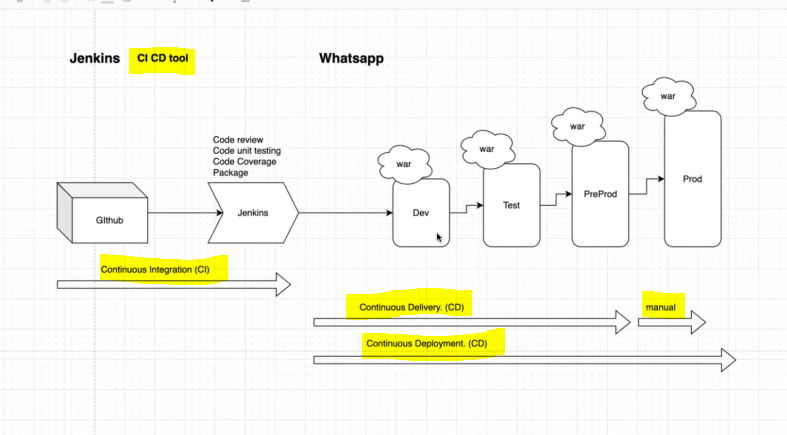
Jenkins pipeline definition stored in a text file. that text file called as Jenkins File. we can write a plain readable text file. which can be then committed to the project Source code repository as well.

either we can write the code in your pipeline syntax also. or write in the Jenkins file and place that file in Source code as the SRC will be reside in the GitHub. so that we can have the everything in a single place. we can have our code and Jenkins files in the same place.

Declarative pipeline Scripted pipeline

**Automated and continuous Deployement:**



Tomcat enables you to access your application on webpage.

* vertical scaling 🡪 Scaling up the resources of the same machine. Its require reboot after scaling up the CPU /capacity.
* Horizontal scaling 🡪 create a new machine with same kind of resources.