## CONTINUOUS INTEGRATION:

It is the combination of continuous build and continuous test

**CONTINUOUS INTEGRATION = CONTINUOUS BUILD + CONTINUOUS TEST**

## CI SERVER:

Here Build, Test & Deploy all these activities are performed in a single CI server

## CONTINUOUS INTEGRATION:

Whenever a developer commits the code using source code management like GIT, then the CI pipeline gets the changes of the code runs automatically build and unit test.

- Due to integrating the new code with old code, we can easily get to know the code is a success or failure.
- It finds the errors more quickly
- Delivery the products to client more frequently
- Developers don't need to manual tasks
- Reduces the developers time 20% to 30%

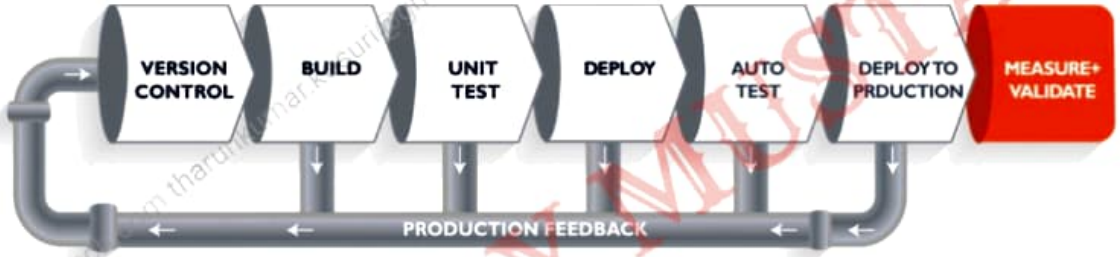
## CD: CONTINUOUS DELIVERY/DEPLOYMENT

**Continuous Delivery** is making it available for deployment. Anytime a new build artifact is available, the artifact is automatically placed in the desired environment and deployed.

**Continuous Deployment** is when you commit your code then it gets automatically tested, build and deploy on the production server.

EX:

## CI/CD PIPELINE:




It looks like Software Development Life Cycle, but let's see how it works.
Lets consider an example, if you are developing a web application

**Version Control:** Here developers need to write code for web applications. So it needs to be committed using version control system like GIT or SVN.

**Build:** Lets consider your code is written in java, it needs to be compiled before execution. In this build step code gets compiled.

**Unit Test:** If the build step is completed, then move to testing phase in this step unit step will be done.

**Deploy:** If the test step is completed, then move to Deploy phase in this step you can deploy your code in testing environment. Here you can see your application output.

**Auto Test:** Once our code is working fine in testing servers, then we need to do Automation testing using Selenium or Junit.

**Deploy to Production:** If everything is fine then you can directly deploy your code in production server.

**NOTE**: If we have error in Code then it will give feedback and it will be corrected, if we have error in Build then it will give feedback and it will be corrected, Pipeline will work like this until it reaches Deploy.

Because of this pipeline, Bugs will be reported fast and get rectified so entire development is fast.

## JENKINS:

Jenkins is an **open source** project written in **java** that runs on the **Window, Linux and Mac OS**


It is community-supported, Free to use, and the First choice for Continuous Integration.

- Consist of Plugins
- Automates the Entire Software Development Life Cycle (SDLC).

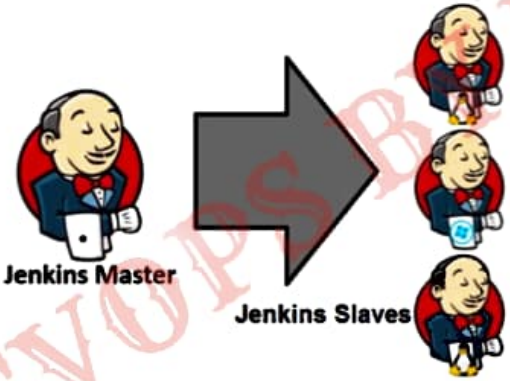
- It was originally developed by Sun Microsystem in 2004 as HUDSON.
- Hudson was an enterprise Edition we need to pay for it.
- The project was renamed Jenkins when Oracle brought the Microsystems.

- It can run on any major platform without Compatibility issues.


- Whenever developers write code, we integrate all the code of all developers at any point in time and we build, test, and deliver/deploy it to the client. This is called CI/CD.

## ADVANTAGES:

- Jenkins follows Master-Slave Architecture.
- You can write your own plugin, can use the community plugin also.
- Can understand the process of what is going on.



- Jenkins master is going to assign a job to the slave..
- If Slaves are not available Jenkins itself does the job.
- By using the labels we can specify the jobs to the nodes.

JENKINS ALTERNATIVES:



GitLab


## JENKINS SETUP:

- Go to **jenkins.io** and copy the links

1. sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
2. sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key

- **Install EPEL** (Extra Package for Enterprise Linux)

1. sudo amazon-linux-extras install epel -y

- **Install java**

1. yum install java-1.8.0-openjdk -y

- **Install git, maven & Jenkins**

1. yum install git jenkins maven -y

- **Restart Jenkins**

1. systemctl restart/start jenkins

- Check Jenkins server is running or not:

1. systemctl status jenkins

- **Connect to dashboard:** copy the public IP address of the server and make a paste on the new tab with Jenkins port number (8080)

1. public_ip:8080

- **To unlock the jenkins:**

1. cat /var/lib/jenkins/secrets/initialAdminPassword
2. Install suggested plugins
3. Add user name, Password and mail id
4. connect to dashboard

## JOB: To perform some set of tasks we use a job in jenkins.

**Freestyle project**

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**Maven project**

Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

**Pipeline**

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**Multi-configuration project**

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

**Folder**

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

**GitHub Organization**

Scans a GitHub organization (or user account) for all repositories matching some defined markers.

**Multibranch Pipeline**

Creates a set of Pipeline projects according to detected branches in one SCM repository.

## PARAMETER TYPES:

- String: any combination of characters and numbers
- Choice: a pre-defined set of strings from which a user can pick a value
- Credentials: a pre-defined Jenkins credential
- File: the full path to a file on the filesystem
- Multi-line String: same as String, but allows newline characters
- Password: similar to the Credentials type, but allows us to pass a plain text param
- eter specific to the job or pipeline
- Run: an absolute URL to a single run of another job

### FILE PARAMETER:

This is used when we want to build our local files.
General — > This Project is Parameterized — > Fille Parameter

### CHOICE PARAMETER:

This parameter is used when we have multiple options to generate a build but need to use only on specific one.
General — > This Project is Parameterized — > Choice Parameter

### STRING PARAMETER:

This parameter is used when we need to pass an parameter as input by default.
It can be any combination of characters and numbers.
General — > This Project is Parameterized — > String Parameter

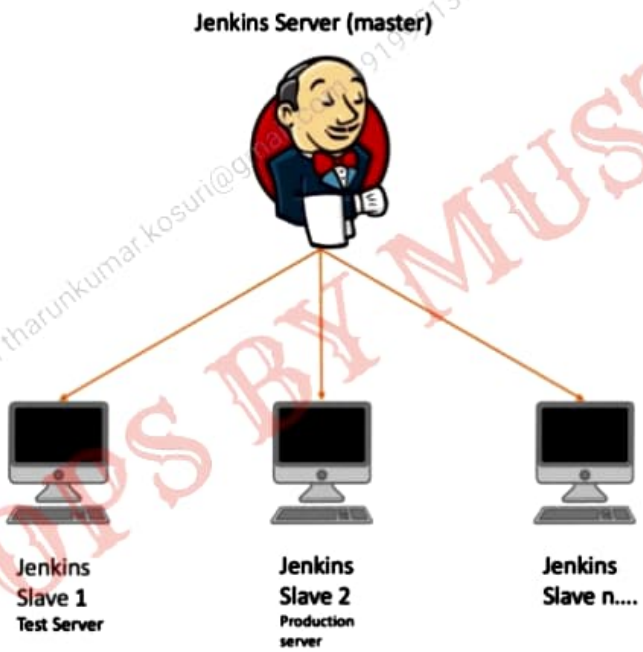
### MULTI STRING PARAMETER:

This will work as same as String Parameter but the difference is instead of one single line string we can use multiple strings at a time as a Parametres.
General — > This Project is Parameterized — > Multi-String Parameter

**LINKED JOBS:** This is used when a job is linked with another job

**TYPES:** Up stream and Down stream
Here for job-1 both job-2 & job-3 are Downstream
For job-2 upstream is job-1 and Downstream is job-3
for Job-3 Both Job-2 & job-1 are Upstream

## MASTER & SLAVE:



**SETUP:**

```
sudo yum update -y
sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key

sudo yum install java-1.8.0-openjdk git maven jenkins -y
sudo systemctl restart jenkins
sudo systemctl status jenkins
copy the IPV4 and paste it on browser like {ipv4:8080}
sudo cat /var/lib/jenkins/secrets/initialAdminPassword

sudo vim /etc/passwd
sudo passwd jenkins
sudo visudo
sudo vim /etc/ssh/sshd_config
sudo systemctl restart sshd
sudo systemctl status sshd
```

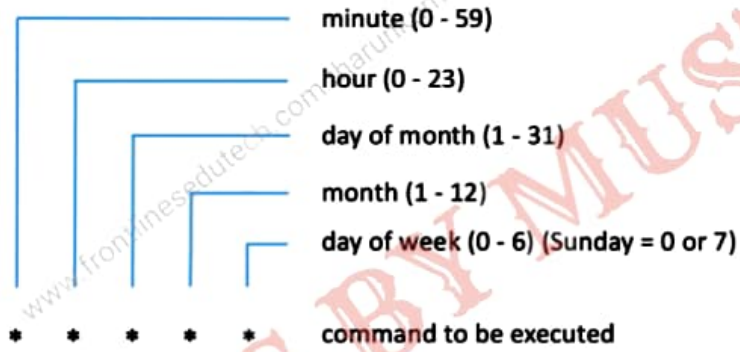
## LOGIN TO SLAVE

```
sudo useradd jenkins
sudo passwd jenkins
sudo visudo
sudo vim /etc/ssh/sshd_config
sudo systemctl restart sshd
sudo systemctl status sshd
```

## LOGIN TO MASTER

```
sudo su jenkins
ssh-keygen
ssh-copy-id jenkins@localhost
yes
exit
ssh-copy-id jenkins@public IPV4 of slave
ssh jenkins@public IPV4 of Slave
```

**CRON JOB:** We can schedule the jobs that need to be run at a particular intervals.




Build Triggers -- > Build periodically -- > * * * * * -- > save

If you want to make it customize then we can use cron syntax generator.