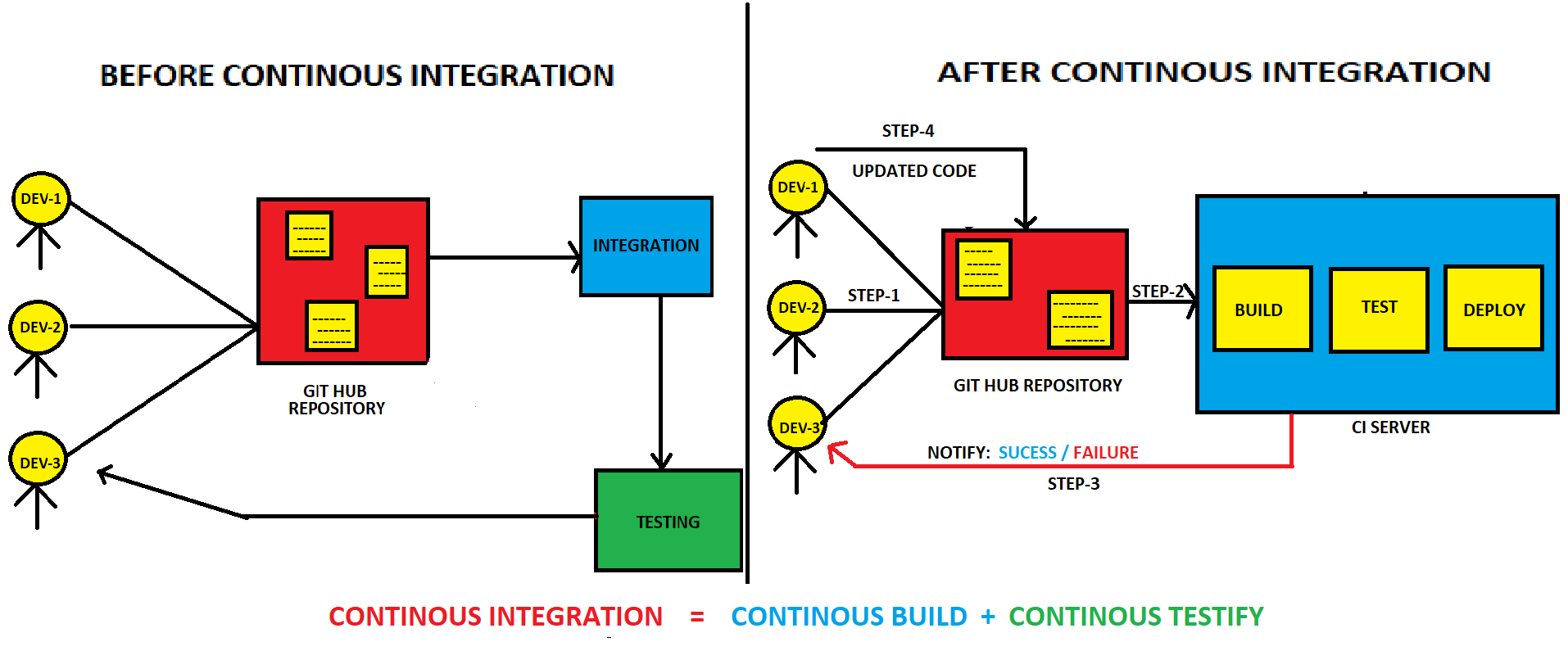
**JENKINS**

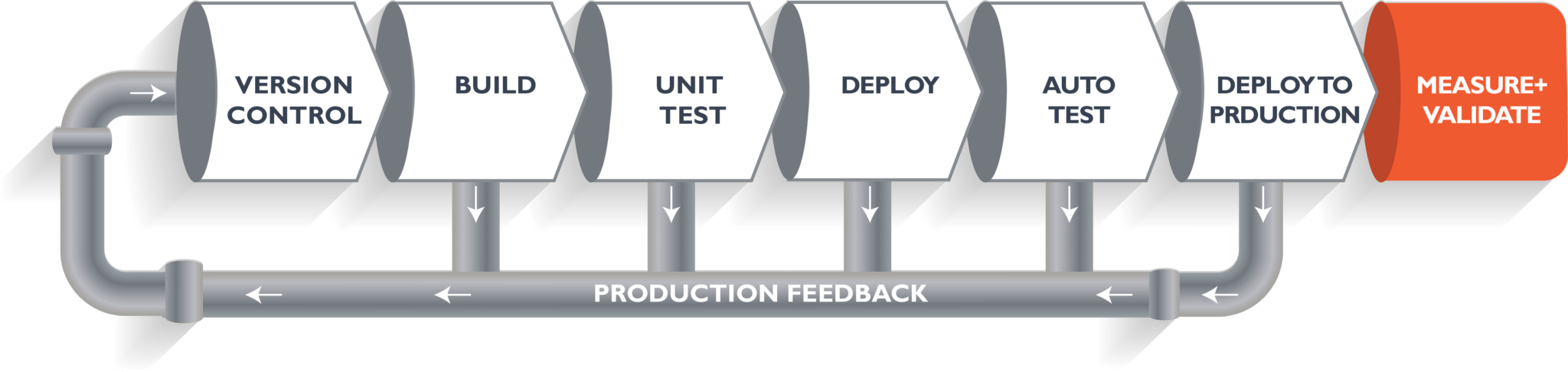
CI = CONTINOUS INTEGRATION CD = CONTINOUS DELIVERY/DEPLOYMENT

CI/CD means it is not a tool a it is a methodology/Framework which used to develop SDLC.

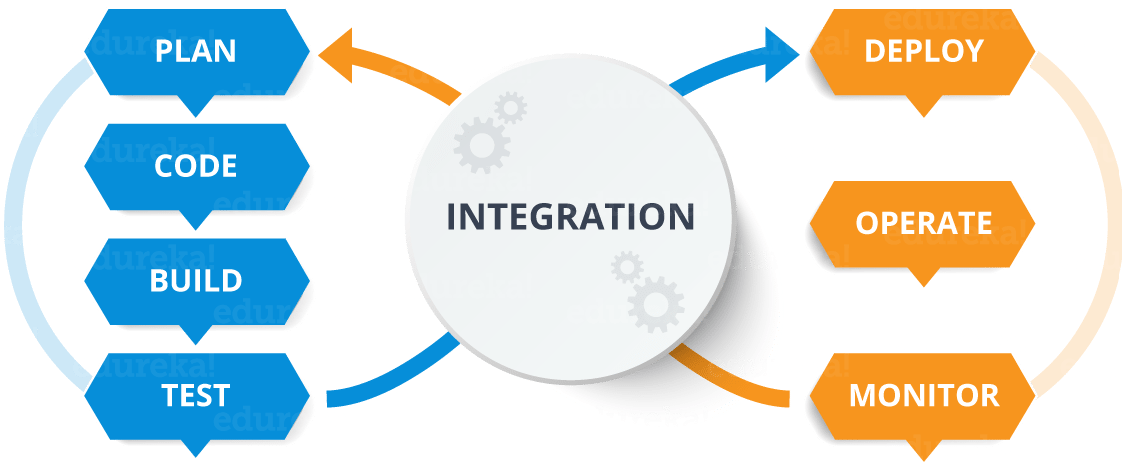
Pipeline follows First come first serve

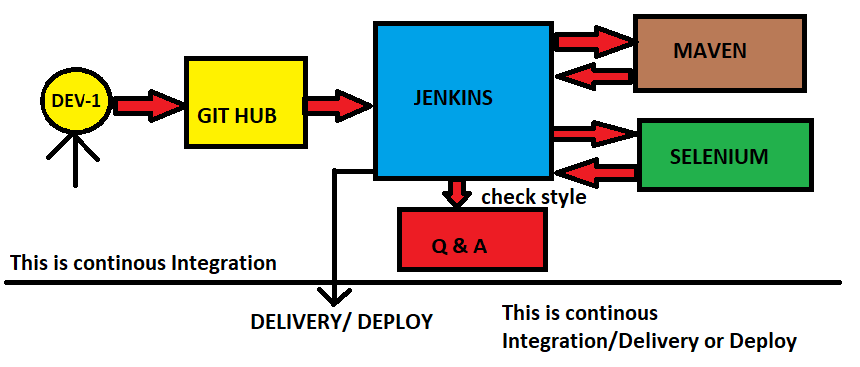
**WHY WE USE IT**





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**.



* Jenkins is an Opensource project written in java that runs on Windows, Linux, Mac-OS.
* It is community-supported and Free to use and 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 as Jenkins when Oracle brought the Microsystems.
* It can run on any major platform without Compatibility issue.
* Whenever developers write code, we integrate all the code of all developers at any point of time and we build, test and deliver/deploy to client. This is called as CI/CD.
* Because of this CI, Bugs will be reported fast and get rectified so entire development is fast.

**WORKFLOW**

* We can attach Git, Maven, Selenium and Artifactory plugins to the Jenkins.
* Artifactory consists of final code which is ready to use.
* Once Developer put code in GitHub Jenkins pull that code and send to Maven for Build.
* Once Build is done, Jenkins pull that code and send to Selenium for Testing.
* Once Testing is done, Jenkins pull that code and send to Artifactory as per requirement.
* We can also Deploy with Jenkins.

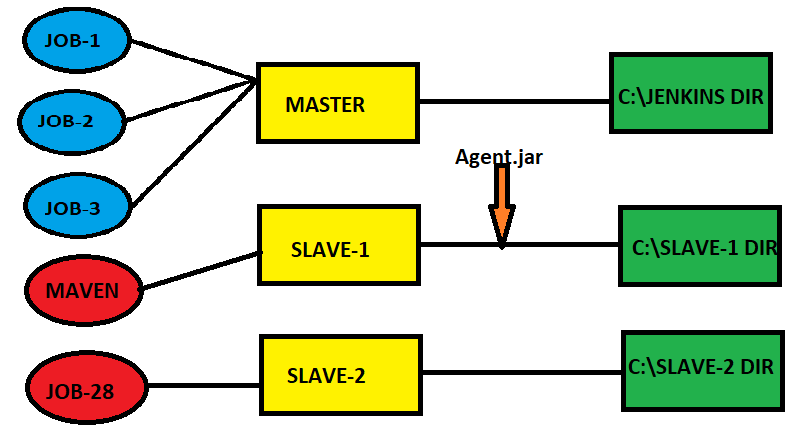
**ADVANTAGES**

* It has lot of plugins and You can write your own plugin, can use community plugin also.
* It is not a tool it is a framework. i.e., you can do whatever you want all you need is plug-ins.
* Jenkins follows Master-Slave Architecture.
* We can attach slaves (Nodes) to Jenkins’s master. It instructs other (Slaves) to do the Job.
* If Slaves are not available Jenkins itself do the job.
* Jenkins also behave as Server Replacement. i.e.., it can do schedule job.
* It can create labels. i.e..., means who will do that task and assigns the tasks.

**JENKINS ALTERNATIVES**

* Bamboo, Travis CI, Circle CI, Team city, AWS code pipeline, Semaphore, Buddy, Build master.

**MASTER-SLAVE CONCEPT**



Manage Jenkins -- > new node -- > name -- > ok -- > Remote root dir: c:\slave1dir -- > Launch method

Select 2nd method -- > click on question mark -- > download Agent.jar and paste it on c: drive -- > java -jar c:\agent.jar give that on launch command -- > Save

Now you build jobs then either master or slave do that builds.

If some specific job (MAVEN) wants to done by specific node (SLAVE1) then

Slave1 -- > config -- > labels: Maven -- > save

Maven job -- > configure -- > Restrict where this project can be done -- > MAVEN -- > Save.

Now build maven job then it will be done by Slave1.

**JENKINS 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 amazon-linux-extras install epel -y : Extra package for enterprise linux

sudo yum install java-1.8.0-openjdk -y

sudo yum install git -y

sudo yum install maven -y

sudo yum install 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 SERVER

sudo useradd jenkins

sudo passwd jenkins

sudo visudo

sudo vim /etc/ssh/sshd\_config

sudo systemctl restart sshd

sudo systemctl status sshd

===========================================================================

GO BACK TO MASTER

sudo su jenkins

ssh-keygen

ssh-copy-id jenkins@localhost

yes

exit

ssh

ssh-copy-id jenkins@public IPV4 of slave

ssg jenkins@public IPV4 of Slave

===========================================================================

GO BACK TO SLAVE

sudo su jenkins

ssh jenkins@public IPV4 of Master

yes

password

logout and restart jenkins

===========================================================================

JAVA INSTALLATION

* Search jdk download -- > install all default values -- > c:users:programfiles:jdk copy path

Environment variables -- > Name: JAVA\_HOME Path: copy and paste path for both user and system variables.

* Open bin and copy that path again and paste it on system variable -- > path -- > paste -- >ok.
* Cmd prompt : java -version and echo %JAVA\_HOME%

**MAVEN INSTALLATION**

* Maven.apache.org -- > Binary zip archive -- > extract on C:\Dev tools -- > Open the location

And copy path -- > Environment variables -- > System variables -- > name:M2-HOME path: paste the location -- > ok. And open bin folder and copy path and paste it on system variable -- > path -- > paste -- >ok now check version: mvn -version

**JOBS IN JENKINS**

* Name -- > freestyle -- > ok -- > Build -- > Add build step -- > execute command -- >Echo “hello” -- > save -- > Dashboard -- > Select it and build.
* If build is green, it is success and red it will be failed. Check details in console output.
* If you want to copy the job new item -- > name: copy project -- > copy from: select job want to copy -- > Ok -- > and you will see the same details of your original job.

**MAVEN JOB, TASK**

* Manage jenkins -- > Manage plugins -- > Download plugins you need
* Restart jenkins once the plug in installation is done.
* Manage jenkins -- > Global tool config -- > config java and maven by giving url.
* Search github.com/RAHAMUSER1007/time-tracker -- > fork -- > it will come to your GitHub.
* In EC2/cmd prompt give git clone [https://github.com/RAHAMUSER1007/time-tracker.git](https://github.com/RAHAMSHAIK007/time-tracker.git)
* cd time-tracker -- > mvn clean package

**BY USING JENKINS NOW**

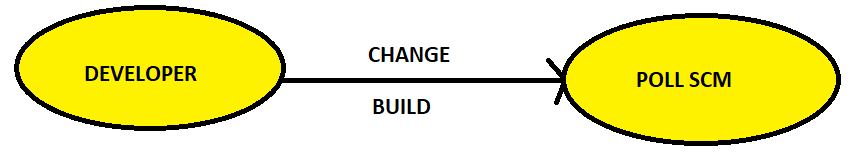
Name -- > maven project -- > git -- > url -- > goals and options: clean package -- > save -- > build now

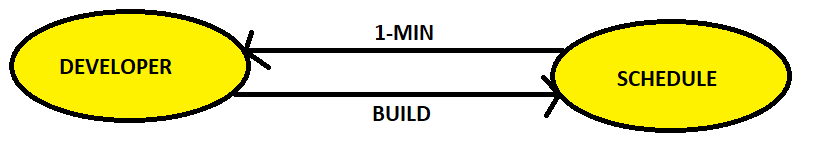
**SCHEDULE PROJECT**

* click on any project -- > configure -- > build -- > triggers -- > build periodically -- > \*\*\*\*\* -- >

(1\*: minutes, 2\*: hours, 3\*: days, 4\*: month, 5\*: week) -- > save.

* Can see automatic build after every minute.
* You can manually trigger build as well.
* For schedule jobs it will be built for every one minute continuously.
* For poll SCM it will het build when there will be only change in the file.





**LINKED PROJECTS**

* Used to do the job one by one. If job1 is done then it will tell to job2 I’m done you go ahead.

**UP STREAM**

* Create a job B and copy from job A you previously build and save.
* Select on job A and click configure -- > build other project -- > select the job B gave -- > ok.
* Build job B and the job A will be waiting to get build.

**DOWN STREAM**

* Create a job B and copy from job A you previously build and save.
* Configure -- > Build after other projects are built -- > select the job you gave -- > save
* Now build the job and the other job will be in waiting process.

**USER MANAGEMENT**

* Jenkins Homepage -- > Manage Jenkins -- >Manage users by default you see a user.
* Create 2 users -- > User1 and User2 & login as Raham (you have all permissions by default).
* Login as raham again -- > manage Jenkins -- > manage plugins -- > Select Role based Authorization strategy and Authorize project -- > install without restart.
* GO to Jenkins home -- > manage Jenkins -- > Config global security -- > select Jenkins’ own user database tick -- > role-based strategy -- > Save and Login as Raham -- > Access denied.
* Now, attach permissions got to Jenkins -- > manage and assign role -- > manage roles -- > role to add -- > Employee.
* Go to item project -- > add developer and tester -- > pattern (dev\* and test\*) -- > Assign roles user/groups to add User1 and User2 (**User1: developer and User2: tester**).
* Config global security -- > Project-based Matrix Authorization Strategy -- > add the user1 and user2 there and save. Now logout and login as user1 once
* Now you will see the user1 dashboard.