Jenkins

1. Introduction to jenkins
2. Installation
3. Archetecture
4. Master and slave conf
5. wt is job
6. creating jobs
7. Sheduling jobs
8. labels
9. continuous Integration
10. Maven project
11. dependencys
12. Continuous Deployment
13. Email Notifications
14. creating users
15. creating roles
16. Blue ocean
17. Catlight Notifier

* Without CI : usually when all the developers commits the code to rep, at the end of the QA collects.
→ Integrate all the code and start testing if any bugs found they've to wait up to next day morning to report to developer.

* whenever dev commits, instantly it integrate the changes and test (unit) it. 15 minites to report to dev.

* Benefits of CI :-
@ immediate bug deduction
@ minimal workflow.
@ we can deploy at any point of time
@ record the build history for tracking

*. why jenkins :
→ jenkins has thousands of plugins which is used to connect to other tools also
→ jenkins is a framework
→ jenkins Acts as cron server replacement.

* Archetecture :
→ 10 projects, 2 hrs each to build, only we hv 7 hrs to build that time we use node.

*. Configurations :-
1. Global Tool configuration
2. Job configuration
3. Node configuration

→ # jenkins is client and server architecture, but no need to install jenkins in both sides, only one i.e jenkins / mast server.

* Labels: if jobs are running in one server, all of sudden it went down then the jobs in that server will not run. So, to overcome this, in jenkins by grouping the server and label them with a name and assign jobs.

*. continuos integration

→. devops guy is responsible to perform a CI in a pro

1. Poll Scm
2. integrating the changes
3. build

→ we need to integrate github with jenkins, to maintain the repo. wt ever changes done by the developer we need to integrate with repo by using poll SCM.

→ Go to jen 1 → configure → SCM

⊙ Git

→ Go to github ~~take~~ one repo, we have to clone that repo URL, and give it to the source code management

→ Git → Repository URL : $\boxed{\text{http://github.com/uma.git}}$

→ add credentials github username / pwd

→ Go to Build trigger, click on poll SCM.

☑ poll SCM

Shedule $\boxed{* \; * \; * \; * \; *}$

→ wt is poll SCM:

It will run as sheduled and go to github repo and check whetter the new changes happen or not in github.

→ build periodically: It will run as we sheduled.

→ Go to Build Environment

☑ add time stamp to the console output.

$\boxed{\text{apply}}$ $\boxed{\text{Save}}$

*. Scheduling :

1. on demand (Build Now)
2. time based (Build periodically)
3. poll based (poll SCM)

*. Dependences:

→ (post build)

→ upstream : before it proceeds to execute a job, it will check for dependences job and execute.

→ downstream: Once we complete the job at the end it calls another job.

*. delivery pipeline:

→ Create some jobs
→ and link one job to other
→ install delivery plugin
→ start plugin

*. Jenkins Views:

1. list view
2. nested view

*. Maven project:

→ Go to global tool config
→ JDK
  Name : [java]
  Java home : [C: program]
→ Maven [maven]
  Maven home [A:\apachemaven3] → [apply]
→ Create a new job called maven and select maven project
→ Go to general Description [develo]
→ SCM
  ⊙ Git
  Repo URL [github URL]
→ Build
  • Root pom [pom.xml]
  • goals & options [package] [apply] [save]

*. Email Notifications:-
If any job fail it will send Notification to the developer

*. Deploy: install deploy plugin
→ create Deploy job
→ post-build actions → deploy war/ear container
→ run empty job [save] [apply]
→ Build now
→ we hv to configure apache tomcat with jobs.
→ Apache tomcat will provide a one basic web browser that is sample.war → click on sample.war in different web browse
→ download sample.war file paste into deploy folder.
→ Go to jenkins → deploy → configure → post-build action
war/ear files [**/*.war]
context path [sample.war]
containes [add ▾]

credentials $\boxed{\text{admin /xx}}$

Tomcat URL $\boxed{\text{http/localhost :80}}$

$\boxed{\text{apply}}$ $\boxed{\text{save}}$

→ Build now → click on console o/p it will show it copied source code from sample.file to apache tomcat.

→ Go to localhost :80 / sample.war then it will come sample "Hello world" application.

- To a JSP page
- To a servelet.

* Users :

→ create new users
→ configure users
→ create and manage user roles.
→ Roles strategy plugin - download -restart jenkins

* cat Light :

→ Status notifier for developers
→ catlight will notify your when builds, bugs and tasks need your attention.
→ It is very handy and useful when you hv to manage multiple job.

1. choose. things to track
2. see status in tray
3. Get notified about the changes

* Blue ocean : Look and feel of jenkins pipelines, jobs, nodes...etc.

* Jenkins Backup home :-

→ jenkins home directory : wr we store all the info about jobs, builds, nodes, logs, plugins, config...etc.
→ we hv a plugin called " back up " plugin.

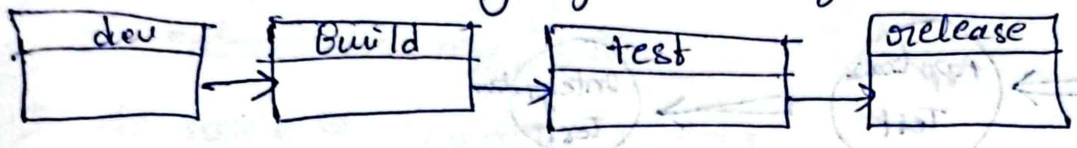
* pipeline :

→ pipeline is workflow with group of events / jobs that are chained and integrated with each other in sequence.

1. delivery pipeline.
2. Build pipeline.

→ install delivery pipeline plugin.

→ create delivery job and give a initial job $\boxed{\text{dev}}$



---

* <u>How to Install jenkins on the Apache Tomcat Server</u> ?

1. Go to the official tomcat website

   https://tomcat.apache.org

2. In downloads select a tomcat version
3. choose windows 64 bit
4. copy the downloaded zip file to the your local fol
   and unzip it.
5. go to the subfolder bin and click on startup
6. Browse to the https://localhost:8080 ~~to~~ launch tomcat s
7. Installing jenkins
8. download jenkins.war file. ~~to the~~ and copy to the subfolder webapps in the tomcat installation
9. Go ~~to~~ the url https://localhost:8080/jenkins to launch the jenkins page