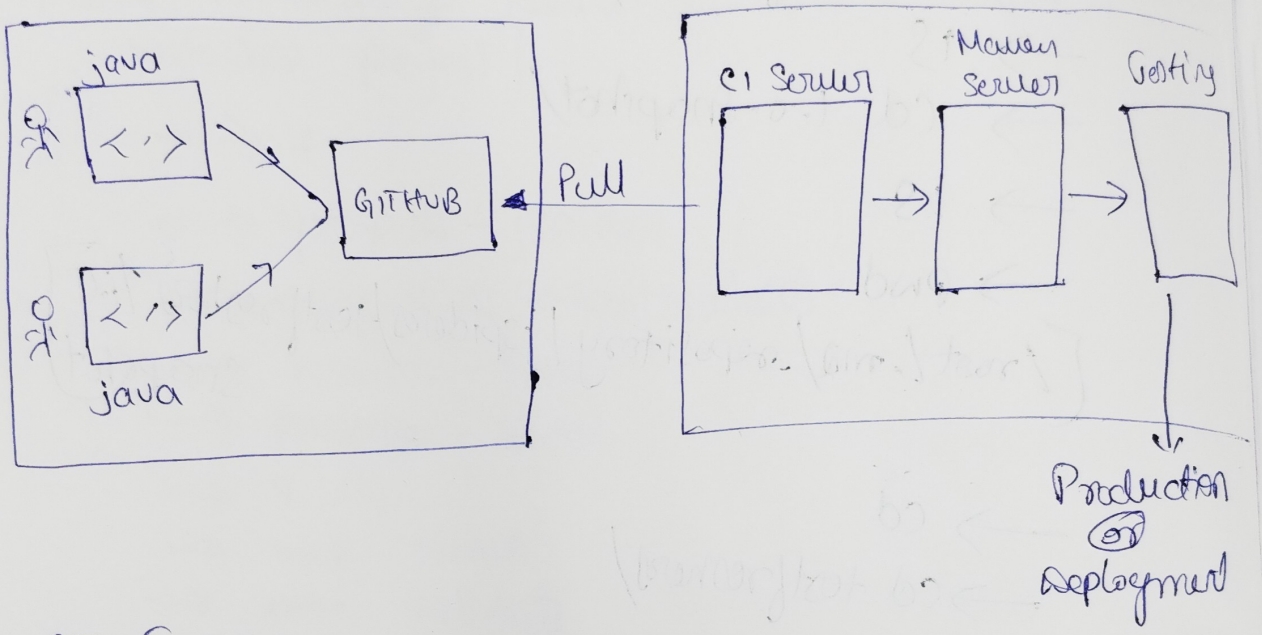
27/02

JENKINS

Jenkins is a open-source continuous integration tool which can be automated continuous integration process with various tools in the project.

* JENKINS ARCHITECTURE -



* CI TOOLS -

- Jenkins
- Github
- Bamboo
- Team City
- Circle CI

* ADVANTAGES -

* NEEDS OF JENKINS -

* INSTALLATION -

SYSTEM REQ:
- AMI amazon Linux 2
- t2. micro
- Allow all traffic
- Terminal - MobaXterm

→ sudo yum update -y

→ sudo su -

→ Google
1. download Jenkins ~~for amazon linux 2~~
   ~~First link~~ Jenkins
   download
   RedHAT
2. Jenkins - Linux
   long term support release
3. download Jenkins for amazon linux 2
   Scroll down for scripts

→ Sudo yum install java-11* -y

(or)

→ Sudo amazon-linux-extras install java-openjdk11 -y

→ java --version

→ Copy paste link [script]

→ y

→ y

→ jenkins --version
2.440.1

→ Public IP (instance) in new tab [Copy paste with portnumber :8080 [Jenkins default port num]
13:233:103:230:8080

→ Systemctl status Jenkins
disabled

→ Systemctl start Jenkins

→ systemctl status Jenkins

→ Refresh IP address
Copy link

→ Sudo cat paste link
[var/lib] Jenkins/secrets/initialAdminPassword

→ Copy password & paste in IP address page

→ Continue

→ Install Suggested plugins

→ Create Admin user - Admin

→ Save & install

→ Start using Jenkins

22/02

TASK 1 : Create a free-style job to execute Linux commands

Jenkins → Dashboard
→ New item
→ job name - Project 1
→ Select free style project
→ OK
→ Build steps ⇒ Execute shell
Pwd
ls
whoami
→ Apply
→ Save
→ Build now [Alt+B]

NOTE -] i] Default path of Jenkins server: /var/lib/Jenkins/workspace/project1

ii] Default user - Jenkins

~~Hobartcom~~

TASK 2 - Create a free-style project to perform GIT operations to prepare build.

→ Dashboard
→ New item
→ item name - Project 2
→ Select free style project

→ Ok

→ Source code Management

→ Select Git

→ Copy repo URL from Github

→ Paste repo URL

Mobax → Sudo gem install git -y (error)

→ git --version

Jenkins → Reload & paste repo URL

→ Branch specifier - */master

→ Apply & save

→ Build now [left]

cd /var/lib/jenkins/workspace/

ls

ls

cd project-2

→ workspace [check]

NOTE: INSTALL MAVEN IN JENKINS

→ Dashboard

→ Manage Jenkins

→ Tools

→ Add Maven

→ Provide name

→ Install.

TASK 3 - Maven Lifecycle

→ Project 2
→ Configure [left]
→ Build step [left]
Invoke top level Maven Targets
→ Select maven tool loaded. Maven
→ Goals - clean package
→ Apply & Save
→ workspace - target - .war file

```
cd /var/lib/Jenkins/workspace
ls
cd project 2
ls
pom.xml src target)
cd target
ls
.war
```

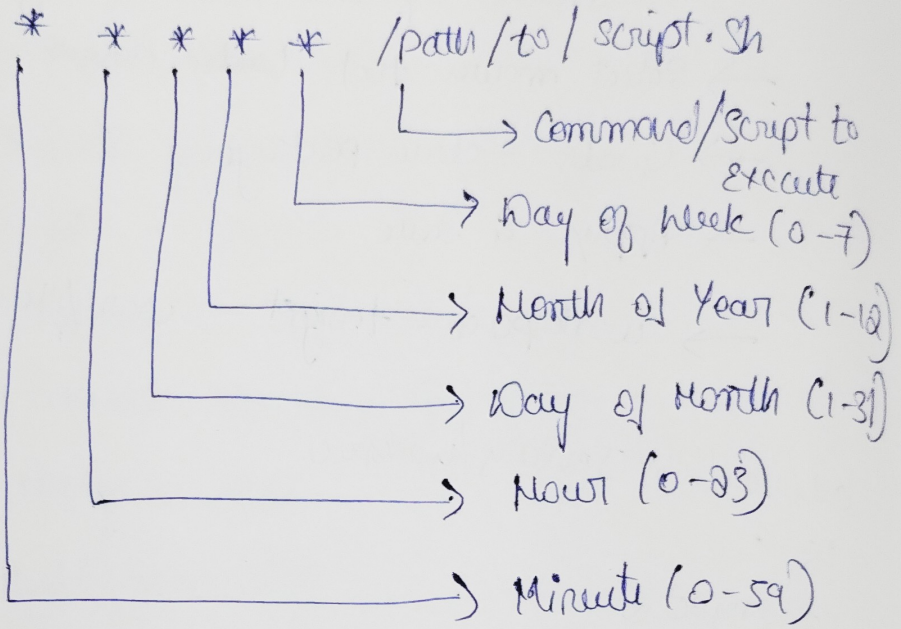
TASK 4: Create a free-style job & perform build lifecycle to prepare build on war file for every one minute

→ Dashboard
→ New item
→ item name - project 3
→ Select free style project
→ SCM
→ copy paste repo URL
→ Build steps
→ tool - Maven

→ Goals - clean package

1 → Build Triggers [yet]

NOTE - Crontab - Job Scheduler



→ Build periodically
(1 space)
* * * * *

→ Apply & Save

To stop → Configuration
→ uncheck build periodically
→ Apply & Save

3 → Build Trigger [yet]
→ Poll SCM [Execute only if there is changes]

Poll SCM is used to trigger build based on the schedule when changes are made in source code only it will trigger based on the schedule
→ * * * * *

→ Apply & Save

01/03

TASK 5 - Create free style project to perform build triggers

→ Dashboard
→ New item
→ name -
→ Select free-style project
→ OK
→ configuration
→ Select Git
→ Paste repo URL
→ Build step - invoke top level Maven
→ Manage Jenkins [left]
→ tools
→ Add Maven tools
→ Install automatically
→ Apply & Save
→ Configuration [left]
→ Goals - clean & package
→ Apply Save
→ Build now [left]
→ Configure → General
→ Build Triggers
Github Hook Trigger [when developer makes changes in repo, then build should be triggered without any schedule]

Github repo → Settings
→ Webhooks (left)
→ Add webhook
→ Payload URL - Jenkins URL/github-webhook/ :8080
http://65.2.177.162:8080/github-webhook/
→ content type - application/json
→ Add webhook.

Jenkins →
→ Apply & Save

TASK 6 - Upstream and downstream Project
→ Name
→ Freestyle Project
→ ok
→ Configuration
build steps
echo "Build succesfull"
→

→ Build after other projects are built
Trigger only job is stable

→ Apply & Save

→ New item
Task 5 - deployment

→ Free-style project

→ OK

→ Build step
Execute Shell
echo "deployement done"

→ Trigger build after other projects are built
Trigger only if build is stable

01/03

* JENKINS USER-MANAGEMENT -

→ Dashboard
→ Manage Jenkins
→ Users
→ Create user
→ Credentials (devops)
→ Create job [item name] sample
→ Build steps
Execute shell
pwd
→ Apply & save
→ Build now
→→ Signout [admin]
Signin [Devops]

→ open sample [job]
Sign as admin

→ Manage Jenkins [left]
→ Users → Jenkins own user's database
→ Devops

NOTE - Jenkins URL & public IP both should be same

* JENKINS USER PERMISSION -

→ Dashboard
→ Manage ~~Security~~ Jenkins
→ Security
→ Authorization
Matrix based security
check admin
→ Add user — Devops
→ Apply & Save.

Sigen as Devops
[Access denied]

Sigen as admin

To give permission for devops
→ Manage Jenkins
→ Security
→ Authorisation.
check devops
→ Apply & Save

PIPELINE

* CI [CONTINUOUS INTEGRATION] -

CI is a process of committing the code, preparing the build & then go for testing [development environment & testing environment] until the build get success with all the tests.

Commit → build → Test ~~→ release → deploy~~

* CD [CONTINUOUS DELIVERY]

CD is a process of releasing tested build from the development environment.

commit → build → test → release

* CONTINUOUS DEPLOYEMENT [CD] -

Continuous deployement is a process of continuous deployement of build in any environment [majorly in production environment / server]

commit → build → test → release → deploy

* JENKINS PIPELINE -

Jenkins Pipeline can be defined in two ways:
- From the script
- Script from SCM

TASK 1: create one pipeline project to print "Hello World"

→ Dashboard
→ New item
→ Pipeline -1
→ Select Pipeline
→ OK

Scroll

→ Pipeline → pipeline script

→ Script terminal [Template available]

```
pipeline {
    agent any
    stages {
        Stage ("Hello")
        {
          Steps{
              {
                echo 'Hello world'
              }
          }
        }
    }
```

→ Apply & Save

→ Build now

Script

```
Pipline {
  agent any
  stages {
    Stage ('compile')
    {
      Steps{
          echo 'compile'
          }
    }
  Stage ('build')
  {
    Steps {
        echo 'build'
        }
  }
```

```
Stage ('test')
{
  Steps{
      echo 'test'
      }
}
Stage ('deploy')
```

TASK 2: Create one pipeline job to execute maven lifecycle for the maven project.

→ Create job pipeline-2

Direct script → Pipline

→ Select pipeline script

Same

→ Manage Jenkins → Tools → Load Maven

→ Script

```
pipeline {
agent any
tools {
    maven 'maven'
}

stages {
  stage ('git clone')
  {
    steps {
      git 'github repo URL'
    }
  }

  stage ('compile')
  {
    steps {
      sh 'mvn compile'
  }  }
```

→ Apply & save

→ Build now

→ script (continued...)

```
stage ('test')
{
    steps {
        'mvn te'
            ↳ gives error pipeline broken

        'mvn test'
    }
}
```

[configure]

```
stage ('build')
{
    steps {
        sh 'mvn package'
    }
}
```

→ Apply & save

NOTE - If any stage is broken, the further stages are broken too.

SCM

→ Create job pipeline-3

→ Pipline → select script from SCM

SCM → GIT

URL → Copy paste repo URL

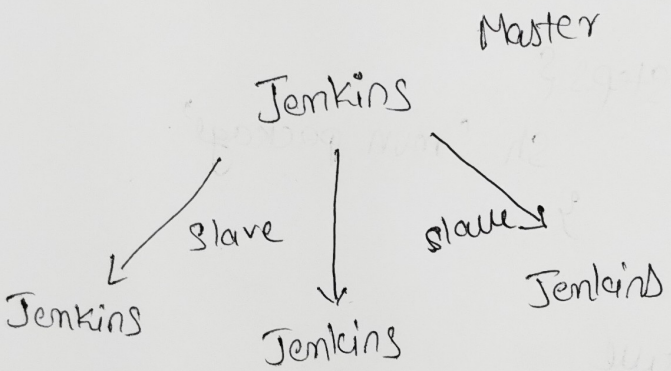
branch → master

→ Script path [copy name & go to repo]

spc → Add file → name Jenkinsfile
→ Paste Pipeline script (edit)
→ commit changes
Jenkins → Script path – Jenkinsfile
→ Apply & Save
→ Build now

02/03

* MASTER SLAVE ARCHITECTURE –



SYSTEM REQ FOR MASTER JENKINS –

- AMI amazon Linux 2
- t2. micro
- All traffic security group

NOTE – In the master slave architecture we will perform so many projects in the different environments like Java, Python, ruby etc.

From the Jenkins server / CI server, if we started with all the technology jobs / projects, the server might be corrupted bcz of more configuration

So that we will take slave nodes [server] to perform projects in the different environment.

slave node
→ Launch an instance = slave-java

Master node
→ sudo su -
→ sudo yum update -y
→ sudo yum install java-11* -y
[both in master & slave node]

NOTE: i) Install Jenkins only in the master node.
ii) Slave node doesn't have jenkins, don't install.

→ Create job - Jenkins 1 [Free-style]
→ Build Steps
Execute shell
uptime [gives server info]
pwd
echo $HOSTNAME
→ Apply & Save
→ Build now
→ Dashboard

TO CONNECT SLAVENODE WITH MASTERNODE -

→ Manage Jenkins
→ Nodes
→ New node
→ Node name - slave-java
→ Click box - Permanent agent [Jenkins will allocate agents]
→ Create
→ No. of executors - 2

→ Remote root directory - /home/root/Build/
(or)
✓ /home/ec2-user/Build

→ Labels = java-slave-job

→ Usage - use this node as much as possible

→ Launch method - launch agent by connecting it to controller (master)

→ Availability - Keep this agent online as much as possible

→ Node Prop - No change

→ Save

→ Click slave_java node

→ (Run from agent command line) Copy

→ slave-node - Paste.

→ Node - check if activated

→

→ Pwd

→ whoami [slave node activated]

→ configure

→ Restrict where this project can run
slave-java

→ Apply & save

→ Build now

slave → exit

→ ls

→ cd Build

→ ls

→ cd workspace

→ ls

→ cd project-1

→ .