**JENKINS**

Link: <https://www.youtube.com/watch?v=cOZcpHiOPc&list=PL87padGs1bp9X95IxGQtFW44qWzywhbHM>

**Jenkins:**

* It is an open source Continuous Integration server written in Java for orchestrating a chain of actions to achieve Continuous Integration in an automated fashion.
* It will support the complete Software development life cycle from building, testing and deployment. The manual work will be automated easily using Jenkins.
* A shell script with requirements will be written and run in build pipeline.
* All the devops operations are integrated with plug-ins.

**Continuous Integration:**

It is a development practice where developers integrate code into a shared repository frequently. Each integration can be verified by an automated build and automated tests.

We can write the script in GUI and Groovy script.

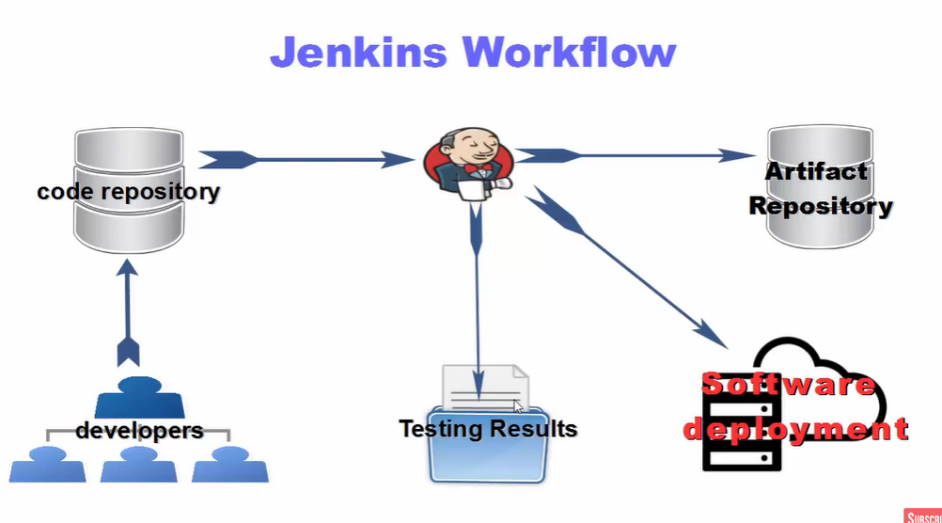
* **CI server continuously integrates newly checked in code into a build.**
* **Run unit test and rejects if they failed**
* **If passed then it Deploys into other servers**

These are the things that happen in CI it is also called build automation.

**Requirements to build a Pipeline:**

* Clone project
* Build
* Unit testing (test the code)
* Acceptance testing (tests for accurate results)
* Reporting (reports after the tests)
* Packaging (keeping all the codes and files in one place)
* Deployment (deploying into an tomcat server)
* Notifications (alerts after deployment whether failed or running)

**Jenkins detects the changes in subversion/git , performs tasks repeatedly. The tasks are build, test, deploy, package, integrate.**

**Jenkins Work Flow:**

Here in Jenkins The developers makes the commits in the GIT or subversions and after the commits the Jenkins will detect the changes and pulls the code and run unit tests and gives us the Test Reports. If the code is run successful then it will be sent to either a Artifact Repository or will be deployed into a server.

**Commands to install Jenkins,git,jdk and maven on linux**

Sudo yum -y install maven

Sudo yum -y install git

Sudo yum -y install Jenkins

sudo wget -O /etc/yum.repos.d/jenkins.repo <https://pkg.jenkins.io/redhat/jenkins.repo>

sudo rpm --import <https://pkg.jenkins.io/redhat/jenkins.io-2023.key>

These are the packages available to run the Jenkins.

**Jenkins login details:**

Username: admin

Password: Chsaikumar@720

**First job in Jenkins:**

To host a static website using Jenkins we need to have some permissions for Jenkins to use shell commands.

Git repo which contains code and install the httpd in the instance and start its services.

Clone the repo into the instance at a /var/www/html file

All these shell commands below should be as build steps in the Jenkins.

Auto trigger the job and execute every minute. By this any changes in the code gets executed.

JENKINS

VM

HTTPD

**Work flow of static website:**

Here the Jenkins install the HTTPD (apache tomcat server) in the virtual machine and started and by default a workspace is created with directory as /var/www/html.

Now the jenkins will clone the code from the git hub in to the /var/www/html and host the website in it.

The website can be access by the IP address of the Virtual Machine.

**Shell script: (this is in the execute shell)**

sudo yum -y install httpd

sudo systemctl start httpd

sudo rm -rf /var/www/html/\*

sudo rm -rf /var/www/html/.git

sudo git clone https://github.com/ch-saikumar-720/jenkins.git

This represents that the jenkins will install httpd to host the website and starts it. The html files along with .git will be removed and repository will be cloned. This process continuous every time we removed the files so as to avoid the errors.

**Steps to host website through Jenkins with GIT:**

* Should have a script in git either java or html & css
* Connect to the Ec2 instance through mobaxterm and install git, maven and Jenkins.

sudo systemctl start jenkins

sudo systemctl status jenkins

sudo systemctl restart Jenkins

sudo systemctl enable –now jenkins

* Use the commands to start the Jenkins
* sudo su – (use this command to move to root user)
* Enable Shell for jenkins User:

vi /etc/passwd (run this command in mobaxterm)

look for Jenkins and there the jenkins /bin/false is changed to /bin/bash. This allows the Jenkins to use the shell commands that are given to it.

* Change group permission for Jenkins like sudo user:

vi /etc/sudoers (run this command in mobaxterm)

﻿jenkins ALL= NOPASSWD: ALL (add this at the end)

Now restart the Jenkins with command sudo systemctl restart jenkins .

* Open Jenkins >> New item >> give name>> freestyle project >> add the repository clone link at the git repos.
* Give git username and password
* In Build, select execute from shell and give the shell script mentioned above in pink.
* Also add the build trigger periodically and give \* \* \* \* \* which means the build will invoke every minute.
* Save and start the build. If any doubts open the build item and check for console output.
* Copy the IP of the Jenkins and paste it in new tab. If not reachable then check the instance for inbound traffic where the port 8080 tcp and http 80 are allowed.
* Any changes made in the git repo will reflect in the website as it is automated to build it every one minute.

Important Points.

* If build agents not found then it will be because of the size of the instance. Minimum size should be 30GB so that agents can be there. Also in Configure change the settings for minimum disk space to 0GB.
* Jenkins URL : <http://IP> address of the instance:8080

**Security in Jenkins:**

Here similar to other services offered by the tools there is options for users and admin.

The admin creates users so that they have privileges to that particular jobs only. Only the admin will have access to all projects. Similar to IAM roles.

**Steps to do security configurations:**

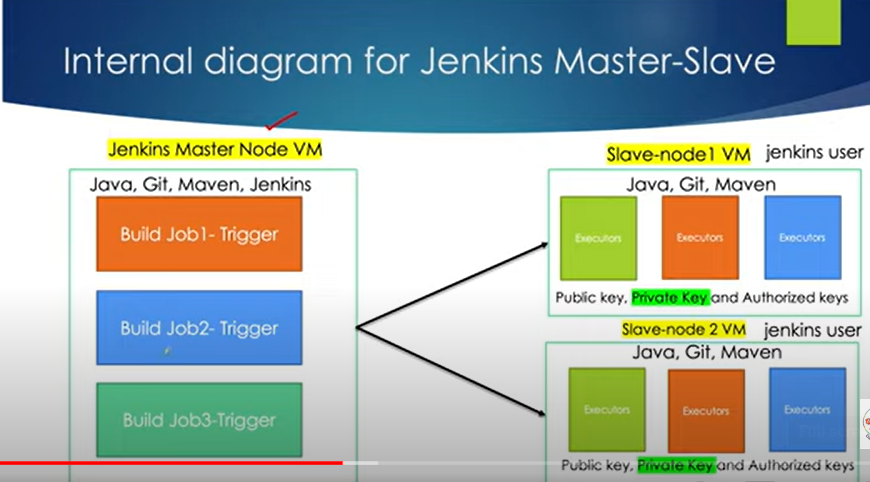
For security we create users and assign them roles, so that they can access what they have permission to.

* Go to Manage Jenkins >> system configuration>> plugins >> search for role-based category and install. >> save
* Security >> allow role based category >> save
* Security >> Users >> create some users with passwords.
* Manage and assign roles >> Manage role >> add a global role employee (read, create, write, delete, configure under View and under Overall read, create)

Here apart from admin the employees will come under employee role.

* Item roles >> add required roles and pattern (pattern means the job names start with some name will come under this role ex: dev.\* all jobs start with dev will come under this) and give all permission. >> Save
* Assign roles >> global role >> admin is admin , users comes under employee so add the users and mark at employee.
* Item role >> add all users >> mark developers under developers and testers under testers. >> Save

From now if you log into Jenkins with developers username and passwords then only jobs related to developers will appear.

**Setup Jenkins Prod Master and Slave in VM:**

Here If we setup all jobs and all projects in one single VM then there will be a chance of failure because all jobs require the usage of CPU from the VM. More jobs means more usage this causes the server down.

As a solution to this we setup the configuration related to the users and jobs in one virtual machine and we call it as Master Node and the execution of the jobs will be at the other virtual machines.

The Master node will trigger the jobs and the slave nodes will build the code and test and deploy.

As per the Master slave design we need multiple VMs for master and slave nodes.

First configure the Master Node in one Virtual Machine.

**Configure the Master Node:**

The Master node configuration is same as normal configuration as done previously.

Now create another two virtual machines in AWS and name them as Slave1 and Slave2.

Install Maven and Git in those two virtual machines using commands

sudo su – (to go to root user)

Sudo yum -y install maven

Sudo yum -y install git

So as maven is installed the java will also get installed. Maven is for build and testing the jobs.

**Configure Slave Node:**

Steps to install SlaveNode

* **Create new user**

sudo useradd Jenkins (a jenkins user is added)

id Jenkins (to check the Jenkins user)

sudo su jenkins (go to Jenkins user)

pwd (shows the ec2-user directory)

cd (do this twice)

cd

pwd (shows /home/Jenkins)

------------------------------------------------------

**Create a directory in jekins (continuous of above)**

mkdir ~/.ssh

ls -la (to check the permissions)

chmod 700 ~/.ssh

cd ~/.ssh (open the .ssh directory)

ssh-keygen -t rsa -C "The access key for Jenkins slaves" (generates a public and private keys in the .shh directory)

Type enter button for every question so that it will create default private key(id\_rsa) and public key(id\_rsa.pub)

cat id\_rsa.pub > ~/.ssh/authorized\_keys (the public key is copied into the authorized\_keys)

chmod 600 ~/.ssh/authorized\_keys (change its permission for access)

cat id\_rsa ( Copy private key and paste in jenkins node slave) (used in next step)

------------------------------------------

**Creating a Slave node in Jenkins Master:**

Open Jenkins >> Manage Jenkins >> Nodes >> New Node>>

Name : Slave2

Permanent agent >> ok

Description : Slave2

Remote root directory: /home/Jenkins

Labels: Slave2

Launch method: Launch agents via ssh

Host: IP of the VM of Slave2

Credentials : Add>> Jenkins >>

Kind : Username with Private key

ID & description: Slave2

Username: Jenkins

Private key : enter directly ( copy paste the private key by using command cat id\_rsa) >> ADD

Credentials : Jenkins Slave2

Host key verification : non verifying verification strategy

SAVE NODE.

The slave node will start to verify use command in CLI

pwd (in jenkins user directory)

ls -la (shows remoting files )

After this the Jenkins user in slave node should have the permissions to use shell commands so follow below

--------------------------------------------------------

**Give shell commands permissions for jenkins user in virtual machine in Slave Node?**

1) sudo vi /etc/sudoers

2) add below line in sudoers file

jenkins ALL=(ALL) NOPASSWD: ALL

**Create a Job in Master Node and run it in Slave Node:**

* Open Master node Jenkins
* New Item >> Name : Slaveproject >> free style >> can create a new project or copy from existing one by searching project name. >> Save
* Restrict where the project can be run

Label Expression: Type the Slave2

As the Slave Node is added in the Jenkins We can directly use it here. That slave node has details of Virtual Machine which in turn triggers this.

* SAVE
* Build now it will be successful
* To cross check it copy the IP of Slave2 VM and google it. You can view the website.

**Jenkins System Environment:(Configuration)**

* **Executors:** The executors can be increased or decreased. By default we have two which means we can run two jobs simultaneously. If you want to run 5 jobs at a time we can increase the executors.
* **Quite Period:** It states that if you use a git repo and did some commits then the build will start after the quite period. Ex: 5 secs, 10 secs or 15 secs.
* Along with this you can add your emails of admin and employees so as to receive the errors related the Jenkins.

We can edit the files using vi or gedit .

**Execute Shell Script using Jenkins:**

* Create a directory jenkins in the root
* Create a file with vi newjob.sh
* Enter the shell commands
* #! /bin/bash

echo “This the shell command” and save this

* Give the execute permissions chmod +x newjob.sh
* Run the script ./newjob.sh
* Open the Jenkins and create a new item and select freestyle project >> Ok
* Build >> execute shell commands >> paste the path of the file like /home/Jenkins/newjob.sh
* Save
* Build now

In this way we know that we can execute the shell commands directly or else create a file and write the shell script. We can run the shell script by pasting the path of the file.

**Automated Code Deployment Maven Project in Jenkins:**

* Maven is a build management tool.
* To install maven and git in Ubuntu we need commands

Sudo apt install maven

mvn --version

Sudo apt install git

git --version

* We can automate this

**Create simple Pipeline in Jenkins:**

1. Remove old workspace
2. Clone the project
3. Clean the existing build
4. Run test cases
5. Generate the jar file.

Below is the script to write a Jenkins in groovy language.

Here in this we follow the same.

>> Jenkins >> New item >> name >> Pipeline>> Save

Github project: URL of git account

Pipeline >> Pipeline script >> enter the below script

------------------------------------------------------------------------

pipeline {

agent any

stages {

stage('Install Apacheserver & start httpd') {

steps {

sh "sudo yum -y install httpd"

sh "sudo systemctl start httpd"

}

}

stage('Remove previous build') {

steps {

sh "sudo rm -rf /var/www/html/\*"

sh "sudo rm -rf /var/www/html/.git"

}

}

stage('clone repo') {

steps {

sh "sudo git clone https://github.com/ch-saikumar-720/jenkins.git"

}

}

}

}

-----------------------------------------------------------------------------------------------------------------------------

>>Save >> Build now

**JENKINS**

Link :

https://www.youtube.com/watch?v=Waet3sH4Qu4&list=PL0lvsZ5ieQicxwgSw4aLbuc9BMPGSjkYt&index=5

Jenkins is CI CD tool developed using JAVA. It has 1000+ plugins and used to create pipelines.

* After installing Jenkins in VM in Linux machine. It has a default path for jenkins.
* /var/lib/jenkins/ is the default path for jenkins in Linux.
* All the GUI information is available in the cli also using the above path.
* All the jobs or projects or items you created in the GUI will be available in inside the /jobs directory of the /var/lib/jenkins directory.

**Nodes:(these are VM)**

Master Node: The VM where the jenkins is installed and configured is called a master node.

Slave Node: The VM where the jenkins job will run without installing the jenkins and simply by adding the jenkins user.

**Plugins:** All the plugins like installed, available will be shown

**Secrets**: The passwords will be stored.

**Users:** Users of the jenkins will be available.

**Workspace:** After you build a job the configuration and output details will be stored here.

**Built Triggers:**

Many options are there to trigger the build

1. **Build after other projects are built** (the build of this project will start only after the build of the a certain project completes)
2. **Build Periodically** (the build will start every 1min or 5min or any time automatically)
3. **Poll SCM** ( the build will start only if there is any change in the source code like git repo then it will check every 1min or 5 min or set time. Then only it will build) ( if no changes then it won’t run)

**Deploying War file in tomcat using Jenikins:**