**JENKINS**

**INSTALLATION OF LATEST VERSION OF JAVA ON PUTTY**

**Step1:** we need a server. So, I am talking EC2 instance.

* Tag Name = JenkinsServer
* Allowing all traffic for now… (but recommended tight security)
* Launch

Step2: Login in to putty using public ID

* [Ec2-user@x.y.z.a](mailto:Ec2-user@x.y.z.a)
* Auth-key\_link
* Launch the Comand Line Interface

Step3: Check weather java in installed or not in that opened CLI by typing

* Sudo -i
* Java

**Note: Two ways in install JENKINS. 1. By Dockers and 2. WAR files. Here we are following WAR files**

**WAR FILE INSTALLATIONS**

Step 4: Copy link Address <http://mirrors.jenkins.io/war-stable/latest/jenkins.war>

* Download your warfile in terminal:

wget <http://mirrors.jenkins.io/war-stable/latest/jenkins.war>

* ls
* Extract that war file: java -jar Jenkins.war
* Check the version of java: java -version
* I am installing java 1.8
* Download java 1.8 [jdk-8u151-linux-x64.tar.gz](http://download.oracle.com/otn-pub/java/jdk/8u151-b12/e758a0de34e24606bca991d704f6dcbf/jdk-8u151-linux-x64.tar.gz)

Step5: Download WinSCP

* Open WinSCP (Secure CoPy)
* Hostname: IP
* Username: ec2-user
* Click on Advance -> ssh -> Authentication -> browse your .ppk key
* Once connected

Select the download JAVA file. Drag and drop in another window

Step7: Once the transfer in done. In your terminal

* Copy the jdk.gz file from ec2-user to root by **sudo cp /pwd/jdk.gz /root**
* You can see file is copied in root
* Go to root by **sudo -i**
* ls-> u can see your 1.8 version
* Extract .gz file by **tar -xvzf jdk-xxxxxxxx.tar.gz**
* Ls -> u can see .gz is been extracted

Step8: Once you extract, in the terminal

* **Which** **java**
* **cd /usr/bin**
* **ls -al | grep java**
* You can see java is in java -> /etc/alternatives/ java
* **cd /etc/alternatives**
* **ls -al | grep java**
* you can see java -> pointed to java 1.7 version, but we want to java to point java 1.8 version
* **cd /root/jdk1.8xxxxxxx/bin**
* ls : here we can see **java**
* So, Copy the path **cd /root/jdk1.8.1xxx/bin/java**
* Change the softlin.
* Go back to cd /usr/bin
* Ls -al | grep java
* Cd /etc/alternatives (or simple cd -)
* **ln -s /root/jdk1.8xxxx/bin/java java** (Here **/root/jdk1.8xxxx/bin/java** symbolic link)
* **ls -al | grep java**
* You can see now it is pointing to **java1.8.1**
* Check **java -version**

**Step9:** Change the java home directory Something like environmental variable

* **Echo $JAVA\_HOME** -> you get /user/lib/jvm/jre (old version java1.7.xxx path)
* **B**ut wee should get latest java1.8.xx
* **Cd /root/jdk1.8.x/**
* **Pwd ->** /root/jdk1.8.xx
* Copy the path**. That path should be our environmental path / Java home you can say**
* Export JAVA\_HOME=/root/jdk1.8.xx
* **Echo $JAVA\_HOME ->** we get latest path, but it works only once, its not permanent. We need to make global, for all terminals
* So copy the path and open file in etc/bashrc **vim /etc/bashrc**
* Add that path at the end of the file **JAVA\_HOME=/root/jdk1.8.xxxxx**
* :wq!
* If I don’t want to set for all user and I wany to make it only for me then add that path to basrc file which is present in-home directory
* Cd /root
* Ls -al -> I can see .bashrc
* Vim .bashrc
* Add end of the line

Similarly to ec2user

Cd /home/ec2-user/

Ls -al

Vim .bashrc file

* Run **source bashrc** to trigger the changes

**Step10: INSTALL JENKINS**

* Run **java -jar Jenkins.war**
* In new window -> copy and paste IPaddress:8080 -> enter
* You will see Jenkins screen asking for password
* Copy and paste password
* Install recommended plugins
* Create new user and password etc
* **Start using Jenkins**

**Step11:** logout and login to check weather working properly

* If you stop backside running terminal by ctrl+c automatically here Jenkins stops working
* Restart again by **Java -jar Jenkins.war &** -> enter -> its starts running properly
* Now you can close the terminal. & helps in running background

**NOTE: I have duplicated the session and closed the present running session or terminal**

**Logged in as a root and checked weather Jenkins is running or not by the command**

* **Ps -ef | grep Jenkins**
* **root 24322 1 20 10:08 ? 00:00:25 java -jar jenkins.war**

**root 24430 24412 0 10:10 pts/1 00:00:00 grep --color=auto Jenkins**

* **cd /root**
* **ls -al**
* **.jenkins**
* **Cd .jenkins**
* **Ls**

You will see **jobs, nodes, users,plugins,secrets,logs etc**

**Note:** We can see that Jenkins will store all of its files in **/root/.jenkins. If you use yum install Jenkins.war, then it will store in /var/lib/Jenkins**

**Manage Jenkins:**

Configure System:

* No of Executors : No of jobs running at a time. Remain build queue. No of concurrent build
* Usage = Use this node
* Retry count = 1

**Credentials**

* Jenkins
* Global Credentials
* Add credentials
* Username
* Password
* ID
* Description
* OK

**CREATE NEW CLONEJOB**

* Under git I have given github link
* Build script
* You can see material in your github will be cloned in your job workspace

**=============================================================================**

**HOW TO TAKE BACKUP OF JENKINS JOB**

**LOCALHOST:8080/job/CloneJob/config.xml**

**=============================================================================**

**ANT - BUILD.XML FILE**

**CREATE NEW BUILDJOB**

**When ever you have build.xml** file in github repo

* Create a new Job name BUILD\_JOB
* Clone the repo(https://github.com/AbhiReddyMiru/JavaApplication.git) into Jenkins first build job first. You will see build.xml in workspace but NOT build folder
* Then, go to **manage Jenkins**
* Global Tool Config
* Under ANT
  + Ant installations – name = Local\_Ant (Some\_Name)
  + Checkbox install automatically
  + Save

Goto Jobs

* In Your job
* Under build
* Invoke ant
  + Ant version – Local\_Ant (Which you have given during ANT INSTALLATION)
  + Targets – leave like that.
  + Save
* Build job now

You can see

Unpacking <https://archive.apache.org/dist/ant/binaries/apache-ant-1.10.1-bin.zip> to /root/.jenkins/tools/hudson.tasks.Ant\_AntInstallation/Local\_Ant on Jenkins

* AND ALSO U SEE

[jar] Building jar: /root/.jenkins/workspace/Build\_job/build/jar/Project.jar

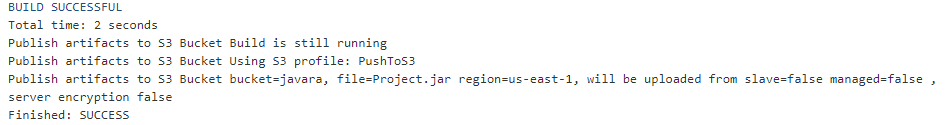
**In workspace you see BUILD.XML file has been unpacked and you can see some folder BUILD FOLDER Inside that you will project.jar file**

**OUR TARGET IS TO ACHIEVE CONTINIOUS INTEGRATION**

* In your build job go to configure
* In Build Trigger -> check the Github Hook Trigger for GITScm polling
* Save
* Go to GitHUB repo
* Under repo settings
* Integrations and service
* Add service
* Jenkins (github plugin)
* Under **Jenkins hook url -> Copy and pass localhost:8080/github-webhook/**
* Save
* Go to Jenkins website and check the build number so that if you make any changes automatically it should build new job So obviously build number increases

**NOW WE ARE STORING OUR ARTIFACTS ON S3 INSTEAD OF JFROG / SONATYPE NEXUS**

* Install new plugin in search type s3. Install s3
* So in BUILDJOB we have configure ANT (Convert .xml file to folder jar )
* So in POST BUILD ACTION, we can see publish artifacts to s3 buckets
* Create a bucket in S3(Not Public) and we need to push our artifacts to that s3 bucket
* Goto Jenkins ->Manage Jenkins -> Configuration System -> You can find AMAZON S3 PROFILES
* Profile name = pushtoS3
* Access key = aws access keys and secret key
* Go toBuildJOb -> Configure -> Add Post build actions -> select s3Profile
* Files to upload
  + Source = build/jar/\*.jar
  + Destination bucket = Bucket name
  + Bucket region = select region
  + Save
* Run the job

****

* It will save in s3 bucket

**DEPLOY THIS JAR FILE IN SERVERS**

* **W**e have done till continuous Integration. We are going to work on Continuous deploy
* We have jar file in S3. We need o download it automatically. Other cannot access our S3 bucket since it is private. So, we will create a ROLE and attach it to EC2 instances
* IAM ROLE
* Create Role in **IAM service**
* EC2 role
* S3fullAccess
* Role name – something like artifact download
* Create role
* Launch EC2 Instance using newly created role attach to it.
* Open Putty With that newly created IP address
* Google aws s3 Cli commands
* In putty
* Mkdir jars
* Cd jars
* pwd
* Aws s3 cp s3://javara(bucket\_name)/Project.jar(jar\_File\_name) Project.jar

If you run that command You can see .jar file in s3 bucket will be downloaded in your ec2

**DEPLOY JOB FOR AUTOMATIC DEPLOY ALL THE TIME**

* MAIN PURPOSE OF THIS JOB IS TO DOWNLOAD THE JAR FILE FROM THE ARTIFACTORY (S3 in our case) SERVER
* Create a new job DEPLOY\_TO\_QAJOB
* Manage plugin -> Publish over SSH -> install
* In Jenkins -> Configurations -> publish over ssh ->

Key -> copy and paste .pem file key

Add SSH server -> Name = QAServer

Hostname = publicIP of QAserver instance

Username = ec2-user

Test Configuraion = gives respose as success

* Open QAserver job
* In configure
* Under BUILD
* Select send files or execute commands over SSH
* Name = QAServer
* Exec Command for auto download -> aws s3 cp s3://javara/Project.jar Project.jar
* Save
* Build job
* You can see Project.jar file is downloaded in ec2 putty

**INTEGRATION AND DEPLOY SHOULD HAPPEN ONE AFTER THE OTHER**

DeployQAJob has to occur automatically once

* So go to QAServerJOB under Build Trigger
* Select Build After other projects are built option
* Projetcs to watch = BuildJob
* Select Trigger if build is stable
* Save

**CHECK THE WHOLE PROCESS CI-CD**

* Remove Project.jar file in ec2 instance of QASERVER
* Make changes in GitHUb
* Save changes
* Go to Jenkins website look at **BUILDQUEUE, You can see BuildJob is build first later followed by QAServer**
* Check in Ec2Instances Project.jar file is downloaded.

**MAVEN - POM.XML FILE**

Download a plugin

* Search maven unleash plugin

Create a new Job for MAVEN

* GlobalToolConfig – Maven
* Select Maven Job
* Configue – Github repo link
* Git
* Build POM.xml
* Clean package

