**Jenkins Concepts notes:**

* Jenkins is an open **source Continuous Integration and continuous delivery** tool written in Java.
* It’s an automation server used to build and deliver software projects.
* It was forked from another project called Hudson, after a dispute with Oracle.
* A major benefit of Jenkins is that it has a lot of plugins available.

**Benefits:**

* Jenkins provides a feedback loop back to the developer to fix build errors.
  + Research has shown that it’s a lot quicker to have a developer fix the error immediately.
* Jenkins can publish every build of your software.
  + This build already has gone through automated testing.
  + When published and deployed to a dev/qa/staging server, you can advance the SDLC much quicker.

**Continuous Integration:**

* Developers commit code to a shared repository on a regular basis.
* Version control system is being monitored. When a commit is detected, a build will be triggered automatically.
* If the build is not green, developers will be notified immediately.
* To detect problems or bugs as early as possible.
* Continuous Integration is a practice of merging development work with the main branch constantly.
* Continuous Delivery is to deliver code to an environment once the code is ready to ship. This could be staging or QA.
* Continuous Deployment is the release of code to production as soon as it is ready.
* Hudson was started in 2004 and later renamed as Jenkins in 2011

**Architecture:**

* Master - Schedule build jobs, dispatch builds to slaves for actual job execution and monitor the slaves with record build results.
* Master can also execute build jobs directly.
* Slave execute build jobs dispatched by master.
* Salves are computers that are set up to build projects for a master.
* Jenkins runs a separate program called "slave agent" on slaves.
* When slaves are registered with master, a master starts distributing loads to slaves.
* Node is used to refer to all machines that are part of Jenkins grid, slaves and master.
* Executor is a separate stream of builds to be run on a node in parallel
* A node can have one or more executors.

**Installation:**

* Present version 2.60.1 installed in our project.
* Connect to the Linux instances in AWS console.
* Run “sudo yum update” to update all the packages.
* Check the java version “java -version”. Here we already installed 1.8.0
* If not, install java with “sudo yum install java-1.8.0”
* To check different versions of java installed, “sudo /usr/sbin/alternatives –config java
* Install the latest Jenkins code using
  + **sudo wget -O /etc/yum.repos.d/jenkins.repo** [**http://pkg.jenkins-ci.org/redhat/jenkins.repo**](http://pkg.jenkins-ci.org/redhat/jenkins.repo)
* Import a key file from Jenkins-CI to enable installation from the package.
  + **sudo rpm --import** [**http://pkg.jenkins-ci.org/redhat/jenkins-ci.org.key**](http://pkg.jenkins-ci.org/redhat/jenkins-ci.org.key)
* Now install the Jenkins
  + Sudo yum install Jenkins
* To start Jenkins,
  + Sudo service Jenkins start
* Access Jenkins server using public DNS of your ec2 on port 8080
  + http://{ec2-public-dns}:8080
  + Here you might have to allow port 8080 in your security group.
* To start the Jenkins on a different port
  + Update port number in /etc/sysconfig/Jenkins
  + We are using JENKINS\_HTTPS\_PORT as 8443
* To fetch admin password,
  + Cd /var/lib/Jenkins/secrets/
  + Cat initialAdminPassword
* To stop Jenkins
  + Sudo service Jenkins stop
* To uninstall Jenkins
  + Sudo service Jenkins stop
  + Sudo yum remomve Jenkins
  + Sudo rm -r /var/lib/Jenkins

**LDAP Integration:**

* To integrate the LDAP connection to the Jenkins, first set up the LDAP server and user details.
* Login as admin to the Jenkins.
* Manage Jenkins 🡪 Configure global security 🡪 select LDAP
* Fill the LDAP server details and if needed mention the root DN and manager DN details in advanced settings.
* Then try to test the connection with sample user and password.
* Once it’s ok, save the changes.

**Tasks**:

* Creating job, creating job from copying previous one
* Copy and rename jobs
* Update plugins
* Delete a job

Jenkins Job Example:

* In our project, we created Jenkins job to pull the code from gitlab.
  + Under SCM section, select GIT and paste the GIT URL and select brance as “release”
* For build triggers
  + Select build when a change is pushed to GitLab and provide URL.
  + Also select Poll SCM. \* \* \* \* \* to poll every minute
* For building, install a plugin “MSBuild”
* Select MSBuild as build and enter details
  + MSBuild version as “mono”
  + MSBuild Build file as “${WORKSPACE}\CMTool\ClientMigrationTool.sln
  + Command line arguments as “/p:Configuration=Release”
* <https://www.codeproject.com/Articles/878203/Integrate-Jenkins-with-MSBuild-and-NuGet>
* For few jobs, we have used Maven3 as build tool.
  + Maven version “Maven3”
  + Root POM “pom.xml”
  + Goals and options “clean compile package -DBUILD\_NUMBER=${BUILD\_NUMBER} -Dmaven.javadoc.skip=true”
* Select post buld action as “Deploy artifacts to artifactory”
  + Artifactory Server <https://artifactory.healthcareit.net/artifactory>
  + Target release repository “advisor-release”
  + Target snapshot repository “advisor-snapshot”
  + Override default credentials. Give service account username and password.

***Jenkins pipeline script examples:***

* <https://jenkins.io/doc/pipeline/examples/>
* <https://www.guru99.com/jenkins-pipeline-tutorial.html>

pipeline {

agent any

stages {

stage('one') {

steps {

parallel("first": {

echo "hello"

},

"second": {

echo "world"

}

)

}

}

stage('two') {

steps {

parallel("first": {

echo "hello"

},

"second": {

echo "world"

}

)

}

}

}

}

***Artifactory Maven Build:***

node {

// Get Artifactory server instance, defined in the Artifactory Plugin administration page.

def server = Artifactory.server "SERVER\_ID"

// Create an Artifactory Maven instance.

def rtMaven = Artifactory.newMavenBuild()

def buildInfo

stage('Clone sources') {

git url: 'https://github.com/jfrogdev/project-examples.git'

}

stage('Artifactory configuration') {

// Tool name from Jenkins configuration

rtMaven.tool = "Maven-3.3.9"

// Set Artifactory repositories for dependencies resolution and artifacts deployment.

rtMaven.deployer releaseRepo:'libs-release-local', snapshotRepo:'libs-snapshot-local', server: server

rtMaven.resolver releaseRepo:'libs-release', snapshotRepo:'libs-snapshot', server: server

}

stage('Maven build') {

buildInfo = rtMaven.run pom: 'maven-example/pom.xml', goals: 'clean install'

}

stage('Publish build info') {

server.publishBuildInfo buildInfo

}

}