Jenkins

Introduction

CI CD CD

Installation

a) Generic Way

b) Install Installable software for respective Operating System

c) Deploy war file into Tomcat server

Jenkins Directory structure

Create the ANT Project using FreeStyle Project

a) Integrate ANT software if not done.

b) Configure EMail Functionality

c) Integrate the SonarQube server and Enable the SonarQube for Jobs

d) Deploy the App into Tomcat

1) Through “Deploy to container “ plugin.

2) Through script

e) Poll SCM

f) Build Periodically

g) Web Hooks

h) Discard Old Build

i) Slack integration

j) JACOCO plugin

Deploy the App into JBoss/WildFly

a) Through script

b) Through “Deploy to container “ plugin.

Create the Maven Project using Freestyle

Integrate Maven software if not done. Integrate Nexus With Jenkins Integrate SonarQube with Jenkins Deploy the App into Tomcat

1) Through “Deploy to container “ plugin.

2) Through script

Plugin Management

a) Safe Restart Plugin : This plugin allows you to restart Jenkins safely.

Note: If you see below error while restarting

Jenkins cannot restart itself as currently configured.

Solution: Restart your command prompt in Administrator mode. So that you are having your all permission to run window as a service.

b) Next Build Number Plugin : Sets the build number Jenkins will use for a job's next build.

c) Email Extension Plugin :

d) SonarQube Scanner for Jenkins :

e) Schedule Build Plugin :

f) Artifactory Plugin : This plugin allows deploying maven artifacts and build info to Artifactory.

h) Cloud Foundry Plugin : Pushes a project to Cloud Foundry or a CF-based platform (e.g. IBM Cloud, Pivotal, Stackato) at the end of a build.

i) Blue Ocean :

k) Deploy to container : For deploying app into Tomcat server/Wildfly/Jboss.

l) Build Name Setter: This plug-in sets the display name of a build to something other than #1, #2, #3, ...

Port Number Change Build with parameters

External Plugins Installation

Urban Code Deploy

a) Create Users (Default Admin)

b) Provide the specific access Jenkins

c) Provide the access to specific access to specific projects

Jenkins CLI

Create the PipeLine Project Jobs

Create the Mulibranch Pipeline Project Jobs

http://localhost:8080/env-vars.html/

Create Master/Slave Backup

Introduction

Jenkins, is an open source Continuous Integration, cross-platform tool written in Java. Kohsuke Kawaguchi is Creator of the Jenkins CI server in 2004. Initially, it was called Hudson, but in 2011 it was renamed to Jenkins because of disputes with Oracle.

The tool simplifies the process of integration of changes in to the project and delivery of fresh builds to users.

Continuous Integration: Continuous Integration (CI) is the process of automating the build and testing of code every time a team member commits changes to version control.

(OR)

Continuous Integration is a development practice where developers integrate their code into a shared remote repository frequently, preferably several times a day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible.

CI – Benefits

- Immediate bug detection
- No integration step in the Software Development lifecycle
- A deployable system at any given point
- Record of evolution of the project

Continuous Delivery: Any and every successful build that has passed all the relevant automated tests and quality gates can potentially be deployed in to production via fully automated one click process.

Continuous Deployment: The practicing of automatically deploying every successful build directly into production without any manual steps knows as Continuous deployment.

(OR)

It is closely related to Continuous Integration and refers to keeping your application deployable at any point or even automatically releasing to a test or production environment if the latest version passes all automated tests.

What Jenkins can do?

- Integrate with many different Version Control Systems (GitHub, CVS, SVN, TFS …)
- Generate test reports (JUnit)
- Push the builds to various artifact repositories
- Deploys directly to production or test environments
- Notify stakeholders of build status (Through Email)

Benefits of Jenkins

- Its an open source tool with great community support.

- Easy to install and It has a simple configuration through a web-based GUI, which speeds up the Job

- It has around 1000+ plugins to ease your work. If a plugin does not exist, just code it up and share with the community (https://plugins.jenkins.io/).

- Its built with Java and hence, it is portable on all major platforms.

- Good documentation and enriched support articles/information available on internet which will help beginners to start easy.

- Specifically, for a test only project, it is used to schedule jobs for regression testing without manual intervention and hence monitor infrastructural and functional health of a application.

It can be used like a scheduler for integration testing and also can be used to validate new deployments/environments on a single click on a Build now button.

Jenkins Installation

- Jenkins is java based CI tool, so we need to install jdk/jre before installing.
- Pre-Requisite Software: Java (Check weather java is installed or not with java -version command)
- We can Install Jenkins in 3 ways.

Method 1:

You can simply deploy the Jenkins WAR in your application server - with Tomcat, for example, you can simply place the jenkins.war file in the webapps Tomcat directory.

Start the tomcat server as follows.

>cd TOMCAT_HOME/bin directory

Windows

>startup.bat (OR) > catalina.bat start

MAC/Linux:

#startup.sh (OR) > catalina.sh start

Tomcat will run 8080 port on by default.

Note: netstat -a will give the port numbers.

If you are running Jenkins on an application server, the URL to use to access Jenkins will be slightly different. For example, on a Tomcat installation by default, you can access Jenkins in your web browser at http://localhost:8080/jenkins .

Method 2 Linux:

yum -y install wget

wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key

yum install jenkins -y

Once the installation of Jenkins completed stop firewall or add the Jenkins port in Firewall in the following way.

firewall-cmd --zone=public --add-port=8080/tcp --permanent # firewall-cmd --zone=public --add-service=http --permanent # firewall-cmd --reload

firewall-cmd --list-all

Now we can start the Jenkins Service by using systemctl as shown below # systemctl start jenkins

systemctl status Jenkins

Now we can access the Jenkins webpage by using the Url: http://ipaddress of server:8080 or http://localhost:8080

Reference Url: https://wiki.jenkins.io/display/JENKINS/Installing+Jenkins+on+Red+Hat+distributions

---

.jenkins : This is the default Jenkins home directory (may be .hudson in older installations) and it will be placed in user's home directory (C:\Users\Teja_ADMIN\ ---> Windows &

/Users/pruthvi --> MAC).

Jenkins home directory contains the below sub directories and configuration files (.xml).

+- jobs

+- [JOBNAME] :Sub directory for each job

+- config.xml : Job configuration file

+- latest : Symbolic link to the last successful build)

+- builds

+- [BUILD_ID] : for each build one build id

+- build.xml : build result summary

+- log : log file

+- changelog.xml (change log)

+- logs ()

+- nodes ()

+- plugins : This directory contains any plugins that you have installed.

+- secrets ()

+- updates : This is an internal directory used by Jenkins to store information about available plugin updates.

+- userContent : You can use this directory to place your own custom content onto your Jenkins server. You can access files in this directory at

http://localhost/jenkins/userContent (if

you are running Jenkins on an application server) or http://localhost:8080/userContent (if you are running in stand-alone mode).

+- users: If you are using the native Jenkins user database, user accounts will be stored in this directory.

+- war : This directory contains the expanded web application. When you start Jenkins as a stand-alone application, it will extract the web application into this directory.

+- config.xml (jenkins root configuration)

+- *.xml(other site-wide configuration files)

+- fingerprints (stores fingerprint records)

---

http://localhost:8080/configure

Home directory: By default, Jenkins stores all of its data in this directory on the file system. Under the Advanced section, you can choose to store build workspaces and build records elsewhere.

There are a few ways to change the Jenkins home directory:

- Edit the JENKINS_HOME variable in your Jenkins configuration file (e.g. /etc/sysconfig/jenkins on Red Hat Linux).
- Use your web container's admin tool to set the JENKINS_HOME environment variable.
- Set the environment variable JENKINS_HOME before launching your web container, or before launching Jenkins directly from the WAR file.
- Set the JENKINS_HOME Java system property when launching your web container, or when launching Jenkins directly from the WAR file.
- Modify web.xml in jenkins.war (or its expanded image in your web container). This is not recommended.

This value cannot be changed while Jenkins is running.

It is shown here to help you ensure that your configuration is taking effect.

Ex: /Users/pruthvi/.jenkins is for my Jenkins which is installed in my local MAC.

Workspace Root Directory: Specifies where Jenkins will store workspaces for builds that are executed on the master.

Build Record Root Directory: Specifies where Jenkins will store build records on the file system. This includes the console output and other metadata generated by a build.

System Message: This message will be displayed at the top of the Jenkins main page.

# of executors: It shows the ow many builds run at a time. E.g.: If give 2, here two builds are running.

Labels:

Usage: Controls how Jenkins schedules builds on this node.

Quiet period:

SCM checkout retry count:

Restrict project naming: Naming Strategy

Strategy

Default ---> This is the default configuration and allows the user to choose any name they like.

Pattern ----> Define a pattern (regular expression) to check whether the job name is valid or not. Forcing the check on existing jobs, will allow you to enforce a naming convention on existing jobs - e.g. even if the user does not change the name, it will be validated with the given pattern at every submit and no updates can be made until the name confirms.

This option does not affect the execution of jobs with non- compliant names. It just controls the validation process when saving job configurations.

Global properties

Environment variables Tool Locations

SonarQube servers

etc….

Create the project/job in Jenkins

Step 1: Login into the Jenkins, go to the Jenkins dashboard left side top corner, click on New Item.

Step 2: Enter the project name in Enter an item name input box and select the Freestyle project

and click on OK Button.

Freestyle project: This is the central feature of Jenkins. Jenkins will build your project combining any SCM and any build system.

A Free-Style project is a project that can incorporate almost any type of build. The Free-Style project is the more "generic" form of a project. You can execute shell/dos scripts, invoke ant, and a lot more. Majority of the plugins are written to use the free-style project.

Maven project: A maven project is a project that will analyze the pom.xml file in greater detail and produce a project that's geared towards the targets that are invoked. The maven project is smart enough to incorporate build targets like the javadoc or test targets and automatically setup the reports for those targets.

Multi-configuration project: The “multiconfiguration project” (also referred to as a “matrix project”) lets you run the same build job in many different configurations. This powerful feature can be useful for testing an application in many different environments, with different databases, or even on different build machines. We will be looking at how to configure multiconfiguration build jobs later on in the book.

Monitor an external job: The “Monitor an external job” build job lets you keep an eye on non-interactive processes, such as cron jobs.

Specify when and how your build should be triggered. The following example polls the Git repository every 5 min. It triggers a build, if something has changed in the repo.

To Install any Jenkins Plugin, follow below steps

Manage Jenkins ---> Manage Plugins ---> Select the Plugin name (HTML Published plugin) ---> Install Without Restart

Package Names: SafeRestartPlugin: SonarQube Scanner:

JavaDoc Plugin jQuery Plugin

Next Build Number Plugin: In UI (Set Next Build Number) Email Extension Plugin: Is used to send the mail from Jenkins. Delivery Pipeline Plugin: To move war/ear/jar into UCD. Maven Integration plugin: For Maven project.

SSH Agent:

Pipeline Maven Integration Plugin: To see Maven Project option while creating job.

Run Condition Plugin Conditional BuildStep Parameterized Trigger plugin:

Cloud Foundry Plugin: To deploy app into Bluemix using cloud foundry.

Embeddable Build Status:

JobConfigHistory Plugin: This plugin saves a copy of the configuration file of a job (config.xml) for every change made and of the system configuration. You can also see what changes have been made by which user if you configured a security policy.

Repository browser:

Role-based Authorization Strategy: Slack Notification Plugin:

Cobertura Plugin: In UI we will see as Coverage Trend.

Artifactory Plugin: For JFrog …. Or any other repositories.

Deploy to container Plugin: For deploying war/ear into Application Server or Webserver.

Build History Metrics plugin: Hudson global-build-stats plugin: Delivery Pipeline View:

Enable project-based security Thin Backup:

Create the Maven project/job in Jenkins Method 1:

Install the Maven Integration Plugin and follow the below steps.

Create the Job using Freestyle project and in the Build section click on Add build step and select the Invoke Top level Maven targets.

Method 2:

Install the Maven Integration plugin and follow the below steps.

Create the New Item as follows.

Provide the item name and select the Maven project and click on OK.

Once you click on OK, you will come to jobs configuration page as follows.

Once you provide all the details click on Save.

Enable email notification

Step 1) Install Email Extension Plugin as follows.

Manage Jenkins ---> Manage Plugins ---> Install “Email Extension Plugin “

Step 2) Add the smtp server host as follows.

Click on Manage Jenkins ---> Configure System --->

Step 3: In Job configure Editable Email as follows.

Select any Job, which we need to configure Email notification ---> Click on Configure ---> Select the Post-build Actions section.

Click on Advanced Settings …

It will expand and will show more settings and click on Add Trigger and select the Always.

We can enable to attach the build logs while sending mail, as follows.

Output mail is like below.

We can enable to Compress and Attach Build Log to email as follows.

Output mail is like below.

Configure SonarQube with Jenkins

Step 1) Install "SonarQube Scanner for Jenkins" for Jenkins plugin. Step 2) Configure SonarQube as follows.

Manage Jenkins ---> Configure System ---> SonarQube servers

Generate the SonarQube server authentication token

Login into SonarQube with Admin user. http://localhost:9000/

User: admin Pwd: admin

Click on Administration tab.

Click on Security dropdown and select the Users

Click on Tokens

Configure SonarQube Scanner for Job/project

Possible Errors

If you see above error, please do the below steps.

Go to the Jenkins Dashboard ---> Click on Manage Jenkins ---> Global Tool Configuration ---> in SonarQube Scanner section - click on SonarQube Scanner installations… and give the below details.

To restart Jenkins manually, you can use either of the following URLs:

(jenkins_url)/safeRestart - Allows all running jobs to complete. New jobs will remain in the queue to run after the restart is complete.

Ex: http://localhost:8080/safeRestart

(jenkins_url)/restart - Forces a restart without waiting for builds to complete.

Ex: http://localhost:8080/restart

(OR)

You can install one plug called SafeRestartPlugin , once installed it will give one option Jenkins dashboard as follows.

Port Number Change

Open the /etc/default/jenkins file and change the below parameter with custom port number.
JENKINS_PORT

By default 8080 is the port, change from 8080 something like 8082 as follow.

#vi /etc/sysconfig/jenkins

JENKINS_PORT="8082"

Once you change the port restart the jenkins service by using below command.

#service jenkins restart

Jenkins - Security

How to create the users in Jenkins?

Click on Manage Jenkins ---> Manage Users ---> Create User ---> Provide the below details Username:

Password: Confirm password: Full name:

E-mail address: Click on Create User

How to see the list of Users in Jenkins?

Once you logged into Jenkins Dashboard

Go to Left Side Navigation Bar ---> Click on People You will see list of users available in Jenkins.

How to remove/delete the User in Jenkins?

Click on Manage Jenkins ---> Manage Users ---> click on below Gear icon one circle with cross symbol

It will ask Are you sure about deleting the user from Jenkins? confirmation message Click on ---> Yes

Now User is deleted successfully.

How to change the password for existing users?

Note: Need to document

Project-based Matrix Authorization Strategy is an authorization method using which we can define which user or group can do what actions on which job. This gives us a fine-grained control over user/group permissions per project.

To Enable the Project-based Matrix Authorization Strategy need to configure in Jenkins as follows.

Step 1: Click on Manage Jenkins and choose the ‘Configure Global Security’ option.

Step 2: Click on Enable Security option.

As an example, let’s assume that we want Jenkins to maintain it’s own database of users, so in the Security Realm, Select the radio button of ‘Jenkins’ own user database’.

Step 3: Under Authorization, select “Project-based Matrix Authorization Strategy” and add 2 or 3 users, one administrator (say devops) and a regular user (say user1 and user2).

All the checkboxes present besides users are for setting global permissions. Select all checkboxes against admin user to give admin full permissions.

For user1, we are selecting read permissions under jobs. With this, user1 would now have read permission to view all jobs which we would be creating later on.

We have to provide read permission under “Overall” category to any regular user otherwise the user won’t be able to see anything after login.

All the checkboxes present besides users are for setting global permissions. Select all checkboxes against admin user to give admin full permissions. For user1, we are selecting read permissions under jobs. With this, user1 would now have read permission to view all jobs which we would be creating later on. We have to provide read permission under “Overall” category to any regular user otherwise the user won’t be able to see anything after login.

Finally, you can click on Save button.

If you get this error, Pease follow below steps.

Solution:

Click on Manage Jenkins ---> Configure Global Security ---> User/group to add: Enter the user Name and click on Add button and --->

Enable the appropriate feature ---> Click on Save Button.

Jenkins Build Status Icon Colours

BLUE: Success

GREY: Disabled, Aborted and Pending YELLOW: Unstable

RED: Failed

Jenkins Job Weather Status Icon Colours

60-80 of recent builds failed.

No recent builds failed.

Note: Need to do more document on these icons.

How to enable the Poll SCM in Jenkins? Step 1: Install the “Git plugin” in Jenkins.

Step 2: Select the job which you need to enable hook and click on Configure ---> In Build Triggers

Section enable the Poll SCM

And provide the values as follows.

Deploy the application into Tomcat

Install the “Deploy to container” plugin.

Open the job which you want to configure deploy, and click on Configure and in Post-build actions

tab, click on ADD POST-BUILD ACTION and select the Deploy war/ear to container

<user username="admin" password="passw0rd" roles="admin-gui,manager-gui,manager-script"/> (OR)

Add the below script in Execute shell Linux/MAC for Tomcat

#!/bin/sh

echo "Starting to copy the build artifact"

cp $WORKSPACE/war/SampleAntProject.war

/Users/pruthvil/pruthviL/Softwares/Running/apache-tomcat-9.0.6/webapps/ echo "Deployed the build artifact into tomcat server successfully"

Windows

echo "Starting to copy the build"

copy %WORKSPACE%\war\SampleAntProject.war C:\\apache-tomcat-8.5.23\webapps\ echo "Copied the build to tomcat"

Linux/MAC for WIldFly

#Deploy in WildFly server

```
#!/bin/sh

echo "Starting to copy the build"

cp $WORKSPACE/war/SampleAntProject.war

/Users/pruthvil/pruthviL/Softwares/Running/wildfly11.0.0.Final/standalone/deployment s/

echo "Copied the build to WildFly successfully"
```

Note: If we want to deploy in Tomcat, which is installed in any remote machine, use below lines of code.

```
scp $WORKSPACE/war/SampleAntProject.war <<User Name>>@<<ServerIP>>:/opt/apache- tomcat-7.0.78/webapps
```

```
cp %JENKINS_HOME%\jobs\%JOB_NAME%\builds\%BUILD_NUMBER%\log C:\Users\windows7\Downloads\newfolder\
```

Integrate JFrog Artifactory with Jenkins

Install “Artifactory Plugin” plugin.

Got to the Manage Jenkins ---> Configure System --->

In the Artifactory section fill the below details and click on Save.

Note: Once you entered all the details click on TEST CONNECTION. IF connection is succeeded you will see the message like Found Artifactory <<Version>>.

Jenkins – Metrics and Trends

There are various plugins which are available in Jenkins to showcase metrics for builds which are carried out over a period of time. These metrics are useful to understand your builds and how frequently they fail/pass over time. As an example, let's look at the 'Build History Metrics plugin'. This plugin calculates the following metrics for all of the builds once installed

Mean Time To Failure (MTTF) Mean Time To Recovery (MTTR) Standard Deviation of Build Times

---

Enable LDAP security to Jenkins

Jenkins has a built-in command line interface (CLI) that allows users and administrators to access Jenkins from a script or shell environment. This can be convenient for scripting of routine tasks, bulk updates, troubleshooting, and more.

Advantages of Jenkins CLI:

- Easier
- Faster
- Memory management
- Automation tasks.

Pre-Requisites

a) Jenkins server should run.

b) Enable security as follows.

Go to Jenkins dashboard in Home page (e.g http://localhost:8080/) -> Manage Jenkins

-> Configure Global Security -> Click on “Enable security” checkbox

You can also configure “Access Control” and “Authorization” option in Global Security page.

Login Jenkins using username and Password

java -jar jenkins-cli.jar -s http://localhost:8080/ help --username devops --password passw0rd

Get the Version of Jenkins

#java -jar jenkins-cli.jar -s http://localhost:8080/ version --username devops --password passw0rd

Get all the jobs of Jenkins

#java -jar jenkins-cli.jar -s http://localhost:8080/ list-jobs --username devops --password passw0rd

Delete the Job

#java -jar jenkins-cli.jar -s http://localhost:8080/ delete-job ant-java-job-dev --username devops --password passw0rd

#java -jar jenkins-cli.jar -s http://localhost:8080/ disable-job ant-web-job-dev --username devops --password passw0rd

While executing above command if you see Enter passphrase, follow the below configuration. (Manage Jenkins ---> Configure Global Security ---> enable the Enable Security ---> Apply and Save.)

Manage Jenkins ---> Configure System --->SSH Public Keys (Enter here any value, same value u can use in CLI)

Jenkins Pipeline Project

Required Plugins

1) Pipeline Maven Integration Plugin
2) JUnit Attachments Plugin
3) Task Scanner Plugin

In Jenkins Pipeline project, we will use one file called Jenkinsfile, in this file we will write groovy code to build process.

We will write Jenkinsfile in 2 ways.

1) Declarative way
2) Scripted way.

1) Declarative Pipeline Syntax
2) Scripted Pipeline Syntax

Jenkins Multi Branch Pipeline Project

Required Plugins

1) Pipeline: Multibranch

Blue Ocean Plugin

Build Chain Executions

Create the 3 jobs like dev, stage and prod. Configure prod job with as follows.

Click on Save.

Jenkins Master-Slave setup

Manage Jenkins ---> Manage Nodes ---> New Node Provide the Node name and click on OK button.

Provide all the details as follows and click on Save button.

Note: Suppose if you don't see "Launch agent via Java Web Start" option, do the below configurations.

Manage Jenkins ---> Configure Global Security ---> enable the TCP port for JNLP agents

(by default, it is Disabled.)

Once you click on Save you will see the Nodes and Master detail, and select the Node which we have created and click on configure.

You will see below screen and click download the slave.jar file.

Copy slave.jar file into any directory (/Users/pruthvil/pruthviL/Softwares/Running/jenkins/node1)

Go to the path where slave.jar copied and run the below command.

java -jar agent.jar -jnlpUrl http://localhost:8080/computer/Node1-Teja-Technologies/slave- agent.jnlp -secret 8e6c24c3e977342073d2184d051b1fb87f30d57acd0c63ae0a913008e65ad86f - workDir "/Users/pruthvil/pruthviL/Softwares/Running/jenkins/node1/workdirectory"

Now slave become communicating to node and it is live. Now you can use this slave for job creation.

Create one Freestyle project/any kind of project and select the Restrict where this project can be run and select the Node which you have crated.

Provide the Git url and click on Save button.