**1. JENKINS**

Jenkins is an open source automation tool written in Java with plugins built for Continuous Integration purpose. Jenkins is used to build and test your software projects continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build. It also allows you to continuously deliver your software by integrating with a large number of testing and deployment technologies.

* At integration stage, build failures are cached.
* For each change in the source code an automatic build report notification is generated.
* To notify developers about build report success or failure, it is integrated with LDAP mail server.
* Achieves continuous integration agile development and test driven development.
* With simple steps, maven release project is automated.
* Easy tracking of bugs at early stage in development environment than production
* Maven 2 project
* Git
* Amazon EC2
* HTML publisher
* Copy artifact
* Join
* Green Balls

**2. CONTINUOUS INTEGRATION**

**Continuous integration** is a process in which all development work is integrated as early as possible. The resulting artifacts are automatically created and tested. This process allows to identify errors as early as possible. **Jenkins is a popular open source tool** to perform continuous integration and build automation.

**3. JENKINS PIPELINE**

The **Jenkins Pipeline plugin** is a game changer for Jenkins users. Based on a *Domain Specific Language (DSL)* in Groovy, the Pipeline plugin makes pipelines scriptable and it is an incredibly powerful way to develop complex, multi-step **DevOps pipelines**.

**4.I JENKINS PIPELINE EXECUTION TYPES**:

Jenkins Pipeline execution engine supports two DSL syntaxes: Scripted Pipeline and Declarative Pipeline. Scripted Pipeline allows users to code their Pipelines using a Groovy DSL. Declarative Pipeline replaces Groovy variable assignments, control structures, loops, and exception handling with a predefined structure and model to allow users of all experience levels to quickly create consistent, concise Pipelines without learning Groovy

**5. DSL JENKINS**

The Jenkins “Job DSL / Plugin” is made up of two parts: The Domain Specific Language (DSL) itself that allows users to describe jobs using a Groovy-based language, and a Jenkins plugin which manages the scripts and the updating of the Jenkins jobs which are created and maintained as a result

**6. JEKINSFILE**

A Jenkinsfile is a text file that contains the definition of a Jenkins Pipeline and is checked into source control.

Creating a Jenkins file, which is checked into source control, provides a number of immediate benefits

* Code review/iteration on the Pipeline
* Audit trail for the Pipeline
* Single source of truth for the Pipeline, which can be viewed and edited by multiple members of the project.

**7. BACKUP JENKINS SERVER**

To create a backup all you need to do is to periodically back up your JENKINS\_HOME directory. This contains all of your build jobs configurations, your slave node configurations, and your build history. To create a back-up of your Jenkins setup, just copy this directory. You can also copy a job directory to clone or replicate a job or rename the directory

**8. TRIGGERS OF JENKIS BUILDS**

Builds in Jenkins can be triggered periodically (on a schedule, specified in configuration), or when source changes in the project have been detected, or they can be automatically triggered by requesting the URL:

<http://YOURHOST/jenkins/job/PROJECTNAME/build>

**9. CREATE JOBS IN JENKINS**

Go to Jenkins top page, select “New Job”, then choose “Build a free-style software project”. This job type consists of the following elements:

optional SCM, such as CVS or Subversion where your source code resides.

optional triggers to control when Jenkins will perform builds.

some sort of build script that performs the build (ant, maven, shell script, batch file, etc.) where the real work happens optional steps to collect information out of the build, such as archiving the artifacts and/or recording javadoc and test results.

optional steps to notify other people/systems with the build result, such as sending e-mails, IMs, updating issue tracker,

**10.       How to integrate GITHUB web hook to configure job for commit**

Install github integration plugin   Configure GITHUB project URL link from browser  configure the Build Trigger “ GitHub Hook trigger for GITscm polling”

Confiuration to be done at GITHUB repository settings

<https://jenkinsurl/github-webhook/>  This line instructs the Jenkins to redirect request to run the Jenkins job.

**11.       How to integrate BITBUCKET  web hook to configure job for commit**

Install Bitbucket integration plugin   Configure GITHUB project URL link from browser  configure the Build Trigger “Configure when change is pushed to Bitbucket”

Confiuration to be done at GITHUB repository settings

<https://jenkinsurl/bitbucket-hook/>  This line instructs the Jenkins to redirect request to run the Jenkins job.

**12.       How to integrate Sonar Qube  integration with Jenkins**

**Sonarqube server** – Running on 9000 Port

**Sonar Qube scanner** – it is client component need to be installed on Jenkins client machine

vim /opt/sonar\_scanner/conf/sonar-scanner.properties

    sonar.host.url=https://sonarqubeurl:9000

**Install sonar scanner plugin**

Manage Jenkins   manage plugin  sonar scanner plugin

Configure sonar server details in Jenkins **configure system**

Configure sonar runner tool configuration in **global tool configuration** tab

Finally configure the job with sonar properties

     Sonar.projectKey=valaxy

     Sonar.projectName =valaxyname

     Sonar.projectVersion=1.0

Sonar.sources=/var/lib/Jenkins/workspace/$JOB\_NAME/webapp/src

**13.       EMAIL configuration:**

    Manage Jenkins  configure system   E-mail notification session ( provide details of SMTP server and port details  Test  mail functionalities using recipient address

Then Create New Job  Go to post build action  select E-mail notification  Add mail recipients

5.       Docker build and publish plugin to build and upload docker images to Docker HUB

**14. How will you secure Jenkins?**

The way I secure Jenkins is mentioned below, if you have any other way to do it than mention that:

* Ensure global security is on.
* Ensure that Jenkins is integrated with my company’s user directory with appropriate plugin.
* Ensure that matrix/Project matrix is enabled to fine tune access.
* Automate the process of setting rights/privileges in Jenkins with custom version controlled script.
* Limit physical access to Jenkins data/folders.
* Periodically run security audits on same.

**15.  Explain how you can deploy a custom build of a core plugin?**

Below are the steps to deploy a custom build of a core plugin:

* Stop Jenkins.
* Copy the custom HPI to $Jenkins\_Home/plugins.
* Delete the previously expanded plugin directory.
* Make an empty file called <plugin>.hpi.pinned.
* Start Jenkins

**16. How do you create Multibranch Pipeline in Jenkins?**

Answer # The Multibranch Pipeline project type enables you to implement different Jenkinsfiles for different branches of the same project. In a Multibranch Pipeline project, Jenkins automatically discovers, manages and executes Pipelines for branches which contain a Jenkinsfile in source control.

XML Job to Job DSL - convert XMl jobs into DSL based groovy scripts

ansi color

java -jar jenkins-cli.jar -s <http://localhost:8080/> help

println("welcome")

8 + 5

import dateutil.\*

def today = new Date()

println(today)

Plugins Used in Jenkins:

XML Job to Job DSL - convert XMl jobs into DSL based groovy scripts

ansi color

deploy to tomat container

Post Build Steps:

Publish Junit test results, archive the artifacts, Delete the workspace, email notification , trigger down stream jobs [Authorize Project](https://plugins.jenkins.io/authorize-project)