Jenkins 4

Plugin Management

Plugins extends the functionality of Jenkins, maven etc.

How to install Jenkins Plugins

- Click on "**Manage Jenkins**" in the left-hand sidebar.
- In the "Manage Jenkins" page, click on "**Manage Plugins**."
- Go to the "**Available**" tab to see the list of plugins that are available for installation.
- Use the **search box** to find specific plugins by name.
- Select the **checkbox** next to the plugins you want to install.
- Click "**Install without restart**" to install the selected plugins immediately, or "**Download now and install after restart"** to install them during the next Jenkins restart.

Some Important and commonly used Jenkins plugins

Plugin	Usage
JACOCO	Code coverage
SSH	Helps Jenkins to run commands remotely in other Linux servers
Publish Over SSH	Transfer files from Jenkins to other Linux servers Transfer artefacts from Jenkins Helps Jenkins to run commands remotely in other Linux servers
SSH Agent	Helps Jenkins master to connect to slaves/agents

Deploy to Container	It helps in deploying applications to Tomcat/GlassFish/JBoss servers.
Deploy WebLogic	Deploys applications to WebLogic servers
Maven Integration	Ease the configuration of applications that uses Maven
Safe Restart	Prevents restarting of Jenkins server abruptly by giving warning if some jobs are still running.
Next Build Number	
Build Name Setter	Customizes build names in Jenkins based on criteria like branch names, aiding in identifying builds within multi-branch pipelines
Email Extension	Extends Jenkins' email notifications, allowing customization of content and recipients based on build status.
Audit Trail	Records system-wide activities in Jenkins, providing an audit trail for tracking changes, user actions
SonarQube Scanner	Integrates Jenkins with SonarQube for automated code quality and security analysis during builds, ensuring code quality standards are met.
Slack Notification	Sends build notifications and updates from Jenkins to Slack channels, enabling real-time communication and collaboration within teams
Schedule Build	Automates the execution of repetitive tasks by scheduling builds at specified times or intervals.
Blue Ocean	Offers a modern and intuitive user interface for Jenkins Pipelines, simplifying pipeline creation, visualization, and management.

Job Import	This facilitates the migration or replication of job configurations across Jenkins environments.

Jenkins Master/Agent/Slave Architecture

The Jenkins master/agent/slave architecture, also known as the Jenkins master/agent architecture, allows distributing the workload of building and testing across multiple machines, known as agents or slaves.

How this Master/agent architecture works.

- Jenkins uses Executors to run tasks on the agents
- SSH Agent plugin is used to securely connect to the agents.
- Only Java is installed on the agents i.e. No need for Jenkins software.

Example: **Creating a Jenkins/Agents (Agent1, Agent2)**

- The two servers are created and named respectively in AWS as EC2 instances.
- Their different SSH connection requirements are noted.
- In Jenkins master server:
  - Navigate to "Manage Jenkins" > "Manage Nodes and Clouds" > "New Node" to add each agent.
  - **Name**: Agent1
  - **Mode**: Permanent Agent
  - Click on "Create Node"
    - **Number of executors:** 2, since it is the second node after the master
    - **Remote root directory:** /home/username e.g /home/ec2-user
    - **Usage:** Use this node as much as possible
    - **Launch method:** Launch agents via SSH
      - **Host:** Agent1 IP address
      - **Credentials:** Add
        - **Kind:** SSH username with private key
        - **ID**: agentCredentials
        - **Description:** agentCredentials
        - Username: ec2-user

- **Private Key:** Enter key directly (The private key can be copied and pasted)
- **Host key verification strategy**: Manually trusted key verification strategy

- **Availability**: Keep this agent online as much as possible
- Click "Save"
- Repeat the above steps to create "Agent2"

Running a Job in a chosen agent e.g. Agent1

Check "Restrict where this project can be run" found in a select project configuration

- **Label Expression:** Agent1
- Click **"Save"** and build the project

NB: The project will be built in the Agent1 even though Maven isn't installed the Agent1.

This is possible because Jenkins master uses executor to run the project in the Agents.

Pipeline Projects in Jenkins

- A pipeline project in Jenkins is a type of job configuration that allows you to define your build process as a script or a series of scripted steps.
- Jenkins Pipeline provides a powerful suite of tools for modelling complex build workflows as code,
- It enables Continuous Integration (CI) and Continuous Delivery (CD) pipelines to be defined and managed alongside the source code they build.
- The scripts are written in groovy language.
- The scripts can be scripted or declarative.
- The created file is known as jenkinsfile

1. **Scripted Jenkinsfile (For a CI/CD pipeline configuration)**
    - Create a new job and select "Pipeline" option
    - Navigate down to "script" textarea and start writing the jenkinsfile.
    - Click **"Save"**

- Build the project. It will then use the written script.
- Builds take place at the nodes

```
node {
    def mavenHome = tool name: 'maven3.8.6'

    stage('1 Clone Code') {
        git 'GithubRespositoryURL'
        // OR
        sh 'git clone https://github.com/fewaitconsulting/maven-web-app'
    }

    stage('2 Test & Build') {
        sh "${mavenHome}/bin/mvn clean package"
    }

    stage('3 Code Quality') {
        sh "${mavenHome}/bin/mvn sonar:sonar"
    }

    stage('4 Upload Artifacts') {
        sh "${mavenHome}/bin/mvn deploy"
    }

    stage('5 Deploy to UAT') {
        // Generated pipeline script from Pipeline Syntax
        // Deploy war/ear to a container
        // WAR/EAR files: target/*war
        // Credentials: Select or add tomcat credentials (username and password)
        // Tomcat URL: Copy and paste the URL of your tomcat from the browser
    }

    stage('6 Approval Gate') {
        sh "echo 'Ready for review'"
        timeout(time: 5, unit: 'DAYS') {
            input message: 'Application ready for deployment, please review and
approve'
        }
    }

    stage('7 Deploy to Prod') {
        // Reuse the generated pipeline script from stage 5
    }

    stage('8 Email Notification') {
        // Generated pipeline script from Pipeline Syntax
        // Extended Email Recipients
        // To: we add the email of the intended recipient(s)
        // Subject: E.g. Build Status
        // Body: We input the message we intend to send to the recipient(s)
    }
}
// The above script can be reused in different similar pipeline projects
```