JENKINS - Pipeline

Jenkins Pipeline is an automation solution that lets you create simple or complex pipelines.
Jenkins Pipeline is a combination of Plugins which automates number of tasks and makes the **CI/CD pipeline** efficient, high in quality and reliable.

Jenkins provides two ways of developing a pipeline
1) Scripted
2) Declarative

- Traditionally, Jenkins jobs were created using Jenkins UI called FreeStyle jobs.
- In Jenkins 2.0, Jenkins introduced a new way to create jobs using the technique called pipeline as code.
- In pipeline as code technique, jobs are created using a script file that contains the steps to be executed by the job.

In Jenkins, that scripted file is called **Jenkinsfile**.

What is Jenkinsfile?

- Jenkinsfile is nothing but a simple text file which is used to write the Jenkins Pipeline and to automate the Continuous Integration process.
- Jenkinsfile usually checked in along with the project's source code in Git repo. Ideally, every application will have its own Jenkinsfile.

Jenkinsfile can be written in two ways -

1) Scripted pipeline syntax
2) Declarative pipeline syntax

1. What is Jenkins Scripted Pipeline?

Jenkins pipelines are traditionally written as scripted pipelines. Ideally, the scripted pipeline is stored in Jenkins web-UI as a Jenkins file. The end-to-end scripted pipeline script is written in Groovy.

It requires knowledge of Groovy programming as a prerequisite.
Jenkinsfile starts with the word node.
Can contain standard programming constructs like if-else block, try-catch block, etc.

Sample Scripted Pipeline

```
node {
    stage('Stage 1') {
        echo 'hello'
    }
}
```

What is Jenkins Declarative Pipeline?

- ✧ The Declarative Pipeline subsystem in Jenkins Pipeline is relatively new, and provides a simplified, opinionated syntax on top of the Pipeline subsystems.
- ✧ The latest addition in Jenkins pipeline job creation technique.
- ✧ Jenkins declarative pipeline needs to use the predefined constructs to create pipelines. Hence, it is not flexible as a scripted pipeline.
- ✧ Jenkinsfile starts with the word pipeline.
- ✧ Jenkins declarative pipeline should be the **preferred way** to create a Jenkins job as they offer a rich set of features, come with *less learning curve* & *no prerequisite to learn a programming language* like Groovy just for the sake of writing pipeline code.
- ✧ We can also validate the syntax of the Declarative pipeline code before running the job. It helps to avoid a lot of runtime issues with the build script.

Our First Declarative Pipeline

```
pipeline {
    agent any
    stages {
        stage('Welcome Step') {
            steps {
                echo 'Welcome to Jenkins Scripting'
            }
        }
    }
}
```

pipeline : Entire Declarative pipeline script should be written inside the pipeline block. It's a mandatory block.
agent : Specify where the Jenkins build job should run. agent can be at pipeline level or stage level. It's mandatory to define an agent.
stages : stages block constitutes different executable stage blocks. At least one stage block is mandatory inside stages block.
stage : stage block contains the actual execution steps. Stage block has to be defined within stages block. It's mandatory to have at least one stage block inside the stages block. Also its mandatory to name each stage block & this name will be shown in the Stage View after we run the job.
steps : steps block contains the actual build step. It's mandatory to have at least one step block inside a stage block.

Depending on the Agent's operating system (where Jenkins job runs), we can use shell, bat, etc., inside the steps command.

Build Pipeline Script

```
pipeline{
    agent any
    environment {
        PATH = "$PATH:/opt/apache-maven-3.6.3/bin"
    }
    stages{
          stage('GetCode'){
            steps{
               git branch: 'main',
               url:
                `https://github.com/ashokitschool/maven_web_app_jenkins_pipeline.git'
             }
          }
          stage('Build'){
            steps{
                sh 'mvn clean package'
            }
        }
    }
}
```

Build Pipeline + Sonar Qube Server - Script

```
pipeline{
    agent any
    environment {
        PATH = "$PATH:/opt/apache-maven-3.6.3/bin"
    }
    stages{
       stage('GetCode'){
            steps{
                git 'https://github.com/ashokitschool/maven-web-app.git'
            }
         }
       stage('Build'){
            steps{
                sh 'mvn clean package'
            }
         }
        stage('SonarQube analysis') {
            steps{
                withSonarQubeEnv('Sonar-Server-7.8') {
                    sh "mvn sonar:sonar"
                }
            }
        }

    }
```

}

JENKINS PIPELINE (JENKINS + MAVEN + GIT HUB + SONAR + TOMCAT)

Note: Install ssh-agent plugin and generate code using pipeline syntax

```
pipeline{
    agent any
    environment {
        PATH = "$PATH:/opt/apache-maven-3.6.3/bin"
    }
    stages{
       stage('GetCode'){
            steps{
                git 'https://github.com/ashokitschool/maven-web-app.git'
            }
         }
       stage('Build'){
            steps{
                sh 'mvn clean package'
            }
         }
        stage('SonarQube Analysis') {
            steps{
                withSonarQubeEnv('Sonar-Server-7.8') {
                    sh "mvn sonar:sonar"
                }
            }
        }
          stage('Code deploy') {
            steps{
                    sshagent(['Tomcat-Server-Agent']) {
                      sh 'scp -o StrictHostKeyChecking=no target/01-maven-web-app.war
ec2-user@13.235.68.29:/home/ec2-user/apache-tomcat-9.0.63/webapps'
                    }
            }
        }
    }
}
```
