Jenkins Pipeline

What is Jenkins Pipeline?

- Jenkins Pipeline is a suite of plugins which supports implementing and integrating *continuous delivery pipelines* into Jenkins.
- A *continuous delivery (CD) pipeline* is an automated expression of your process for getting software from version control right through to your users and customers.

What is Jenkins Pipeline?

- Every change to your software (committed in source control) goes through a complex process on its way to being released.
- This process involves building the software in a reliable and repeatable manner, as well as progressing the built software (called a "build") through multiple stages of testing and deployment.

Declarative versus Scripted Pipeline syntax

- A Jenkinsfile can be written using two types of syntax - **Declarative and Scripted.**
- Declarative and Scripted Pipelines are constructed fundamentally differently. Declarative Pipeline is a more recent feature of Jenkins Pipeline which:
- provides richer syntactical features over Scripted Pipeline syntax, and
- is designed to make writing and reading Pipeline code easier.

Pipeline concepts

Node : A node is a machine which is part of the Jenkins environment and is capable of executing a Pipeline.

Also, a node block is a key part of Scripted Pipeline syntax.

Stage: A stage block defines a conceptually distinct subset of tasks performed through the entire Pipeline (e.g. "Build", "Test" and "Deploy" stages), which is used by many plugins to visualize or present Jenkins Pipeline status/progress.

Step: A single task. Fundamentally, a step tells Jenkins *what* to do at a particular point in time (or "step" in the process). For example, to execute the shell command make use the sh step: sh 'make'.

Declarative Pipeline fundamentals

Jenkinsfile (Declarative Pipeline)

```
pipeline {
    agent any 1
    stages {
        stage('Build') { 2
            steps {
                // 3
            }
        }
        stage('Test') { 4
            steps {
                // 5
            }
        }
        stage('Deploy') { 6
            steps {
                // 7
            }
        }
    }
}
```

Declarative Pipeline fundamentals

1. Execute this Pipeline or any of its stages, on any available agent.
2. Defines the "Build" stage.
3. Perform some steps related to the "Build" stage.
4. Defines the "Test" stage.
5. Perform some steps related to the "Test" stage.
6. Defines the "Deploy" stage.
7. Perform some steps related to the "Deploy" stage.

Example : Declarative Pipeline

```
pipeline {
  agent any
  stages {
    stage('clone repo') {
      steps {
        sh "rm -rf my-app"
        sh "git clone https://github.com/satsbit/my-app.git"
        sh "mvn clean -f my-app"
      }
    }
    stage('Test') {
      steps {
        sh "mvn test -f my-app"
      }
    }
    stage('package') {
      steps {
        sh "mvn package -f my-app"
      }
    }
  }
}
```

Scripted Pipeline fundamentals

Jenkinsfile (Scripted Pipeline)

```
node {  (1)
    stage('Build') { (2)
        // (3)
    }
    stage('Test') { (4)
        // (5)
    }
    stage('Deploy') { (6)
        // (7)
    }
}
```

Scripted Pipeline fundamentals

1. Execute this Pipeline or any of its stages, on any available agent.
2. Defines the "Build" stage. `stage` blocks are optional in Scripted Pipeline syntax. However, implementing `stage` blocks in a Scripted Pipeline provides clearer visualization of each `stage's subset of tasks/steps in the Jenkins UI.
3. Perform some steps related to the "Build" stage.
4. Defines the "Test" stage.
5. Perform some steps related to the "Test" stage.
6. Defines the "Deploy" stage.
7. Perform some steps related to the "Deploy" stage.

Example : Scripted Pipeline

```
pipeline {
    agent any
    stages {
        stage('---clean---') {
            steps {
                sh "mvn clean"
            }
        }
        stage('--test--') {
            steps {
                sh "mvn test"
            }
        }
        stage('--package--') {
            steps {
                sh "mvn package"
            }
        }
    }
}
```