## What is Jenkins Pipeline?

Jenkins Pipeline (or simply "Pipeline" with a capital "P") is a suite of plugins which supports implementing and integrating continuous delivery pipelines into Jenkins.

Pipeline provides an extensible set of tools for modeling simple-to-complex delivery pipelines "as code" via thePipeline domain-specific language (DSL) syntax.

### What are Domain Specific Languages (DSLs)?

A Domain Specific Language is a programming language with a higher level of abstraction optimized for a specific class of problems. A DSL uses the concepts and rules from the field or domain.

Creating a Jenkinsfile and committing it to source control provides a number of immediate benefits:

* Automatically creates a Pipeline build process for all branches and pull requests.
* Code review/iteration on the Pipeline (along with the remaining source code).
* Audit trail for the Pipeline.
* Single source of truth for the Pipeline, which can be viewed and edited by multiple members of the project.

While the syntax for defining a Pipeline, either in the web UI or with a Jenkinsfile is the same, it is generally considered best practice to define the Pipeline in a Jenkinsfile and check that in to source control.

### Pipeline concepts

The following concepts are key aspects of Jenkins Pipeline, which tie in closely to Pipeline syntax

#### 

#### Pipeline

A Pipeline is a user-defined model of a CD pipeline. A Pipeline’s code defines your entire build process, which typically includes stages for building an application, testing it and then delivering it.

Also, a pipeline block is a key part of Declarative Pipeline syntax.

#### 

#### 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'. When a plugin extends the Pipeline DSL, that typically means the plugin has implemented a new step.

### Pipeline syntax overview

The following Pipeline code skeletons illustrate the fundamental differences between Declarative Pipeline syntax and Scripted Pipeline syntax.

Be aware that both stages and steps (above) are common elements of both Declarative and Scripted Pipeline syntax.

### Declarative Pipeline fundamentals

In Declarative Pipeline syntax, the pipeline block defines all the work done throughout your entire Pi

## Create your initial Pipeline as a Jenkinsfile

You’re now ready to create your Pipeline that will automate building your Java application with Maven in Jenkins. Your Pipeline will be created as a Jenkinsfile, which will be committed to your locally cloned Git repository.

This is the foundation of "Pipeline-as-Code", which treats the continuous delivery pipeline as a part of the application to be versioned and reviewed like any other code.

First, create an initial Pipeline and add a "Build" stage to the Pipeline that begins orchestrating this whole process.

| pipeline {  agent { node { label 'linux'} }  options {  timestamps()  disableConcurrentBuilds()  }    environment {  JAVA\_TOOL\_NAME = 'Zulu JDK 8'  MAVEN\_TOOL\_NAME = 'Maven 3'  JAVA\_HOME = tool('Zulu JDK 8')  IS\_BUILD\_SERVER = "true"  MAVEN\_HOME = tool('Maven 3')  }  stages {  stage(Build) {  steps {  script {  sh 'mvn clean install deploy -DskipTests'  }  }  }  } } |
| --- |

Agent - We at Nomura use the agent where the pipeline execution is done, are pre-stored images from Nexus Repository or pre-saved nodes in Jenkins.

Environment - As we are deploying for a Maven Project, we need to set the “Linux” Agent with the following Environment.

Java JDK is taken from Zulu JDK 8

Maven Version - 3.x.x

Java Home is taken from Zulu JDK

Maven Home is taken from Maven 3 Tool

Stages:

We are initializing the maven build on the pom.xml from our repository and builds it and deploy to Nexus Repository as a SnapShot.