Module 6- Continuous Delivery using Pipelines

What are Pipelines

- Top-level component of continuous integration, delivery, and deployment.

Pipelines comprise:

- Jobs that define what to run. For example, code compilation or test runs.
- Stages that define when and how to run. For example, that tests run only after code compilation.
- Multiple jobs in the same stage are executed by Runners in parallel, if there are enough concurrent Runners.

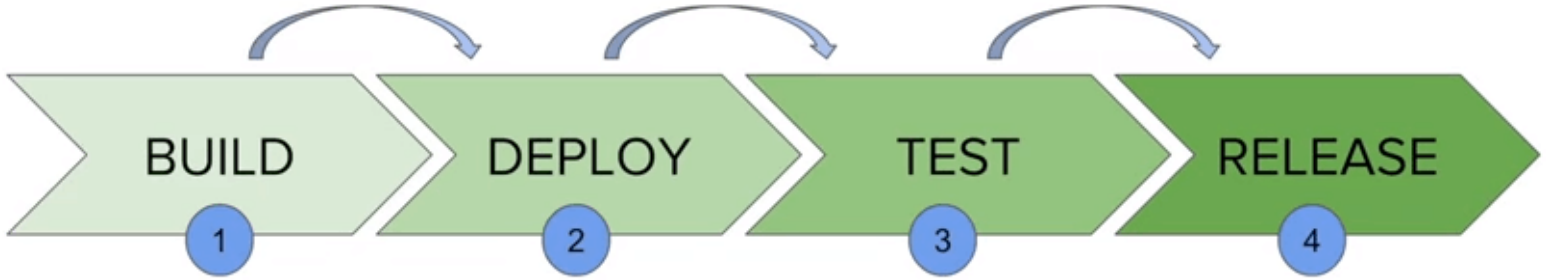
If all the jobs in a stage:

- Succeed, the pipeline moves on to the next stage.
- Fail, the next stage is not (usually) executed and the pipeline ends early.

Benefits of Pipelines

- Jenkins pipeline is implemented as a code which allows multiple users to edit and execute the pipeline process.
- Pipelines are robust. So if your server undergoes an unforeseen restart, the pipeline will be automatically resumed.
- You can pause the pipeline process and make it wait to resume until there is an input from the user.
- Jenkins Pipelines support big projects. You can run multiple jobs, and even use pipelines in a loop.

How Build Pipeline Works



- It contains a group of states called build, deploy, test and release.
- These events are interlinked with each other. Every state has its events, which work in a sequence called a continuous delivery pipeline.
- A continuous delivery pipeline is an automated expression to display your process for getting software for version control. Thus, every change made in your software goes through a number of complex processes on its way to being released.
- It also involves developing the software in a reliable and repeatable manner, and progression of the built software through multiple stages of testing and deployment.

Jenkins Build Pipeline

Overview of Pipeline as a Code

- Describes a set of features that allow Jenkins users to define pipelined job processes with code, stored and versioned in a source repository.

- These features allow Jenkins to discover, manage, and run jobs for multiple source repositories and branches — eliminating the need for manual job creation and management.

- To use Pipeline as Code, projects must contain a file named JenkinsFile in the repository root, which contains a "Pipeline script."

Overview of Jenkinsfile

- Text file that stores the entire workflow as code and it can be checked into a SCM on your local system. This enables the developers to access, edit and check the code at all times.

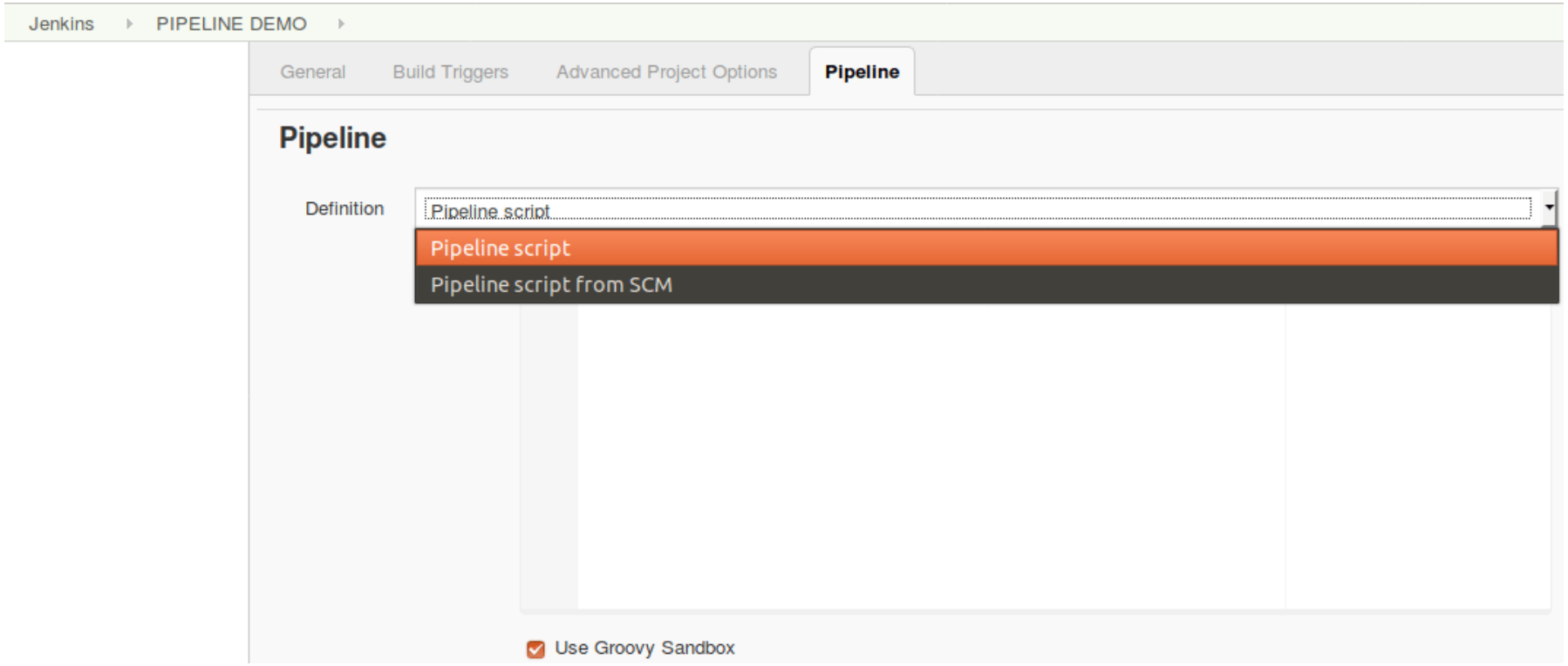
- Written using the Groovy DSL and it can be created through a text/groovy editor or through the configuration page on the Jenkins instance. It is written based on two syntaxes, namely:

  1. Declarative pipeline syntax
  2. Scripted pipeline syntax

- **Declarative pipeline** is a relatively new feature that supports the pipeline as code concept. It makes the pipeline code easier to read and write. This code is written in a Jenkinsfile which can be checked into a source control management system such as Git.

- **Scripted pipeline** is a traditional way of writing the code. In this pipeline, the Jenkinsfile is written on the Jenkins UI instance

Pipeline Script



Docker Plugin for Jenkins

- This plugin allows containers to be dynamically provisioned as Jenkins nodes using Docker. It is a Jenkins Cloud plugin for Docker.

- The aim of this docker plugin is to be able to use a Docker host to dynamically provision a docker container as a Jenkins agent node, let that run a single build, then tear-down that node, without the build process .

- The Jenkins administrator configures Jenkins with knowledge of one or more docker hosts (or swarms), knowledge of one or more "templates" and Jenkins can then run docker containers to provide Jenkins (slave agent) Nodes on which Jenkins can run builds.

Ref: https://plugins.jenkins.io/docker-plugin/

Using Docker with Pipeline

- Pipeline is designed to easily use Docker images as the execution environment for a single Stage or the entire Pipeline.
- A user can define the tools required for their Pipeline, without having to manually configure agents.
- Any tool which can be packaged in a Docker container can be used with ease by making only minor edits to a Jenkinsfile.

Jenkinsfile (Declarative Pipeline)

```
pipeline {
    agent {
        docker { image 'node:7-alpine' }
    }
    stages {
        stage('Test') {
            steps {
                sh 'node --version'
            }
        }
    }
}
```

Ref: https://www.jenkins.io/doc/book/pipeline/docker/

Working with Jenkins Promotions

- Promoted Build plugin is used to distinguish good builds from bad builds by introducing the notion of 'promotion’.

- A promoted build is a successful build that passed additional criteria . The typical situation in which you use promotion is where you have multiple 'test' jobs hooked up as downstream jobs of a 'build' job. You'll then configure the build job so that the build gets promoted when all the test jobs passed successfully. This allows you to keep the build job run fast.

- Promoted builds will get a star in the build history view, and it can be then picked up by other teams, deployed to the staging area, etc., as those builds have passed additional quality criteria

Ref: https://plugins.jenkins.io/promoted-builds/

THANKS