3-12-19

Jenkins pipeline Jobs:

* We write pipeline jobs using code
* It is also referred as pipe line as a code.
* We use Groovy DSL (domain specific language) – under line scripting is groovy.
* We can directly use groovy scripts inside pipeline.

**Jenkins pipeline supports 2 different syntaxes:**

* Declarative pipeline (New feature)
* Scripted pipe line

**Why to use pipeline**

Jenkins is, fundamentally, an automation engine which supports a number of automation patterns. Pipeline adds a powerful set of automation tools onto Jenkins, supporting use cases that span from simple continuous integration to comprehensive CD pipelines. By modeling a series of related tasks, users can take advantage of the many features of Pipeline:

* **Code**: Pipelines are implemented in code and typically checked into source control, giving teams the ability to edit, review, and iterate upon their delivery pipeline.
* **Durable**: Pipelines can survive both planned and unplanned restarts of the Jenkins master.
* **Pausable**: Pipelines can optionally stop and wait for human input or approval before continuing the Pipeline run.
* **Versatile**: Pipelines support complex real-world CD requirements, including the ability to fork/join, loop, and perform work in parallel.
* **Extensible**: The Pipeline plugin supports custom extensions to its DSL [[1](https://jenkins.io/doc/book/pipeline/#_footnotedef_1)] and multiple options for integration with other plugins.

**Example of scripted pipe line:**

* Scripted pipe line contains node
* Node ( can be master or slave)
* Stages (stage comes within a node)

node{

stage(‘SCM checkout’){

//checkout code from git

}

stage(‘maven build’){

//build using maven

}

}