Assignment 6

Aim: - To build pipeline using Jenkins.

Lo Mapped: - Lo 1 and Lo 3

Objective: -

1. Understand concept of pipeline in Jenkins.
2. Students will be able to make pipeline using jenkins

Theory: -

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. 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. 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. Pipeline provides an extensible set of tools for modeling simple-to-complex delivery pipelines "as code" via the Pipeline domain-specific language (DSL) syntax. The definition of a Jenkins Pipeline is written into a text file (called a Jenkinsfile) which in turn can be committed to a project's source control repository. This is the foundation of "Pipeline-as- code"; treating the CD pipeline a part of the application to be versioned and reviewed like any other code.

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 [3] 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.

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

Many of the individual syntactical components (or "steps") written into a Jenkinsfile, however, are common to both Declarative and Scripted Pipeline. Read more about how these two types of syntax differ in Pipeline concepts and Pipeline syntax overview below

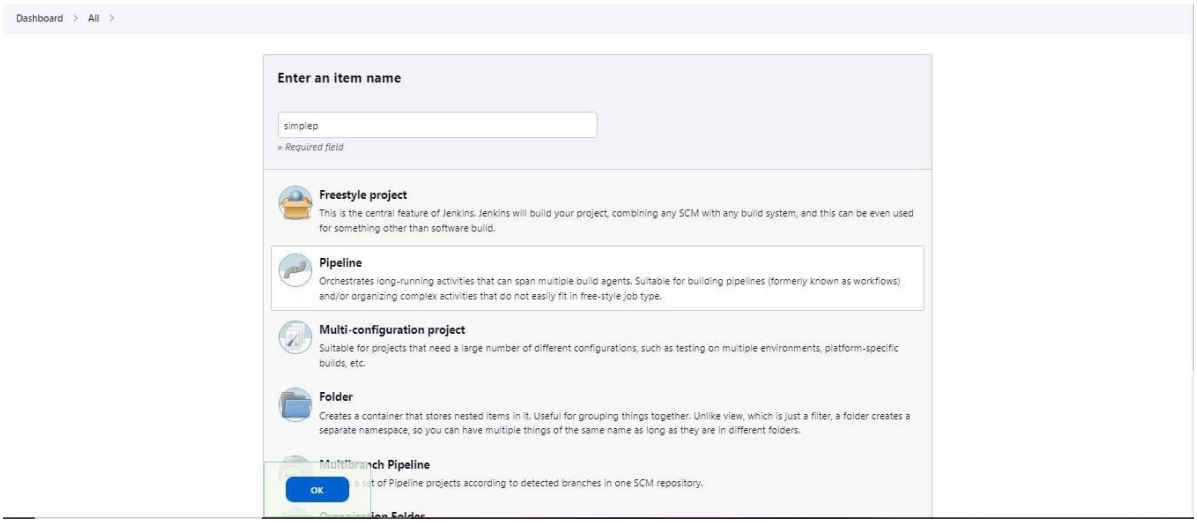
Why Pipeline?

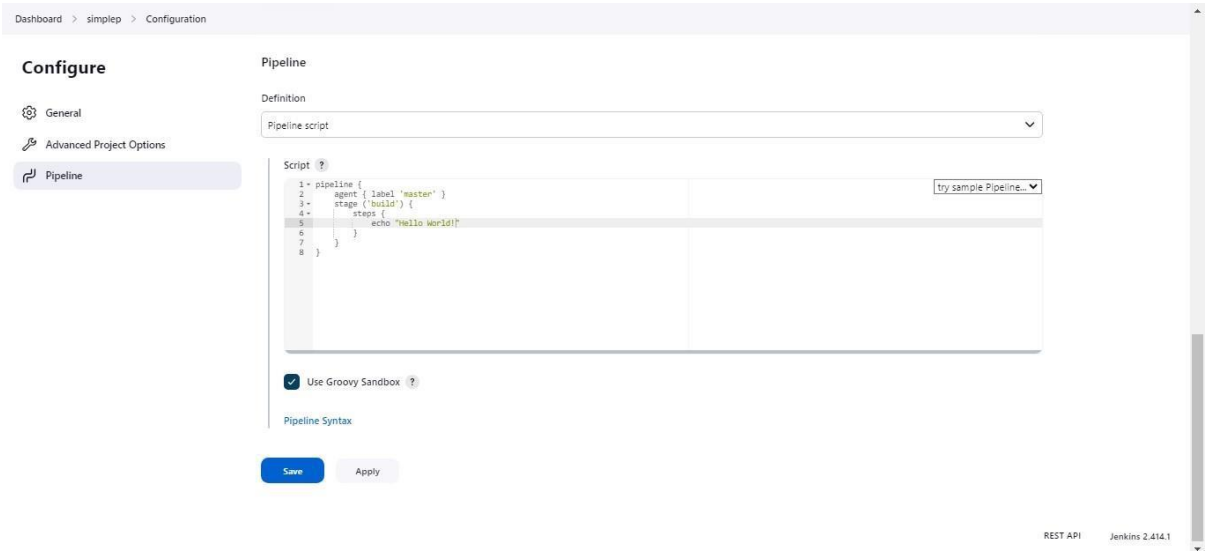
Jenkins is, fundamentally, an automation engine which supports a number of automation patterns. Pipeline adds a powerful set of automation tools onto Jenkins, supporting use cases that span from simple continuous integration to comprehensive CD pipelines. By modeling a series of related tasks, users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

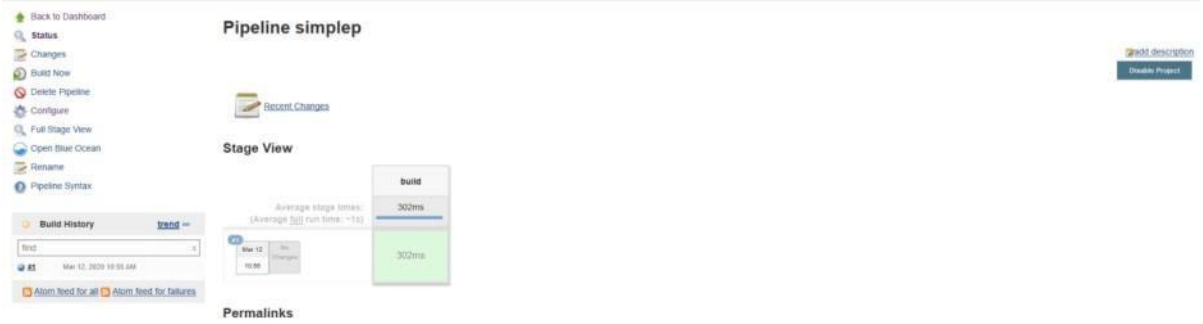
- Code: Pipelines are implemented in code and typically checked into source control, giving teams the ability to edit, review, and iterate upon their delivery pipeline.
- Durable: Pipelines can survive both planned and unplanned restarts of the Jenkins master. • Pausable: Pipelines can optionally stop 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 and multiple options for integration with other plugins.

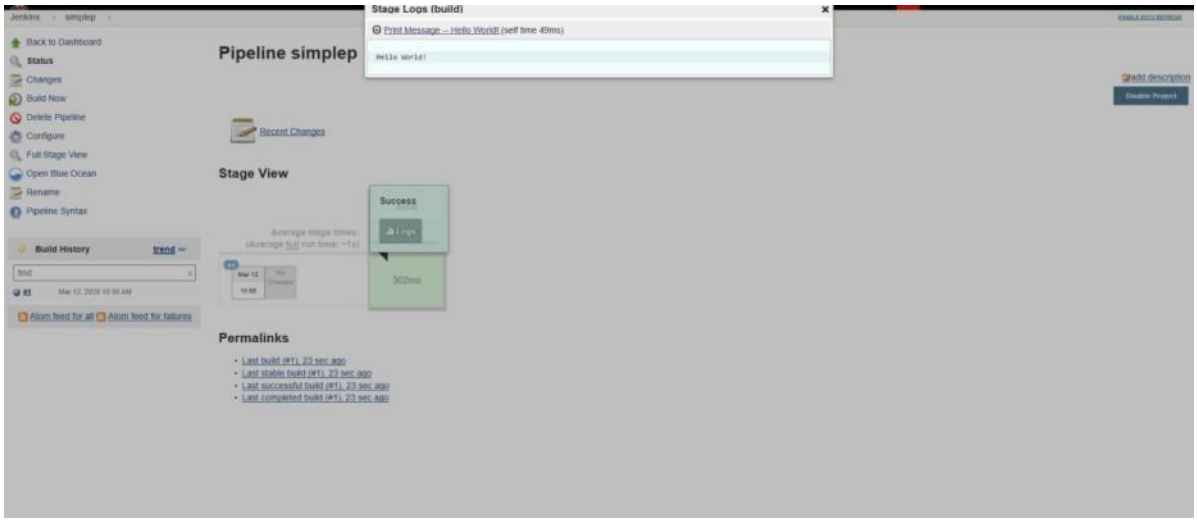
While Jenkins has always allowed rudimentary forms of chaining Freestyle Jobs together to perform sequential tasks, Pipeline makes this concept a first-class citizen in Jenkins. Building on the core Jenkins value of extensibility, Pipeline is also extensible both by users with Pipeline Shared Libraries and by plugin developers.

Output :-




Dashboard > simplep > Configuration
Configure
General
Advanced Project Options
Pipeline
Pipeline
Definition
Pipeline script
Script
pipeline {
agent { label 'master' }
stage ('build') {
steps {
echo "Hello World!"
}
}
}
try sample Pipeline...
Use Groovy Sandbox
Pipeline Syntax
Save
Apply
REST API
Jenkins 2.414.1


Back to Dashboard
Status
Changes
Build Now
Delete Pipeline
Configure
Full Stage View
Open Blue Ocean
Rename
Pipeline Syntax
Build History
trend
Atom feed for all
Atom feed for failures
Pipeline simplep
add description
Disable Project
Recent Changes
Stage View
build
302ms
Permalinks


Stage Logs (build)
Print Message – Hello World (self time 45ms)
Hello World!
Pipeline simplep
Back to Dashboard
Status
Changes
Build Now
Delete Pipeline
Configure
Full Stage View
Open Blue Ocean
Rename
Pipeline Syntax
Build History
trend
Atom feed for all
Atom feed for failures
add description
Disable Project
Recent Changes
Stage View
Success
Logs
302ms
Permalinks
Last build (#1), 23 sec ago
Last stable build (#1), 23 sec ago
Last successful build (#1), 23 sec ago
Last completed build (#1), 23 sec ago

```
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in D:\jen\workspace\simplep
[Pipeline] {
[Pipeline] stage
[Pipeline] { (build)
[Pipeline] echo
Hello World!
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Conclusion: - In this experiment we had created pipeline in Jenkins.