6. Pipeline using Jenkins

AIM:
To build pipeline using Jenkins.

THEORY:
What is a Pipeline in Jenkins?
A Jenkins Pipeline is a set of automated processes that help developers and DevOps teams build, test, and deploy code in a continuous integration/continuous delivery (CI/CD) environment.
In simple terms, it's a workflow that defines how software moves from code to production in an automated and repeatable manner.

Key Benefits of Jenkins Pipeline:

- Automates CI/CD processes
- Provides better visibility into the build process
- Offers version control of build logic (Pipeline as Code)
- Supports complex workflows (parallel execution, conditional logic, etc.) Types of

Jenkins Pipelines:

There are two main types of syntax used to define Jenkins pipelines:

1. Declarative Pipeline
Simpler and more structured syntax
Designed for users who want an easy-to-read format
Introduced later to make pipelines more readable and maintainable Key Features:

- Uses pipeline {} block
- Organized into stages and steps
- Easier error handling
- Better validation in Jenkins UI

2. Scripted Pipeline

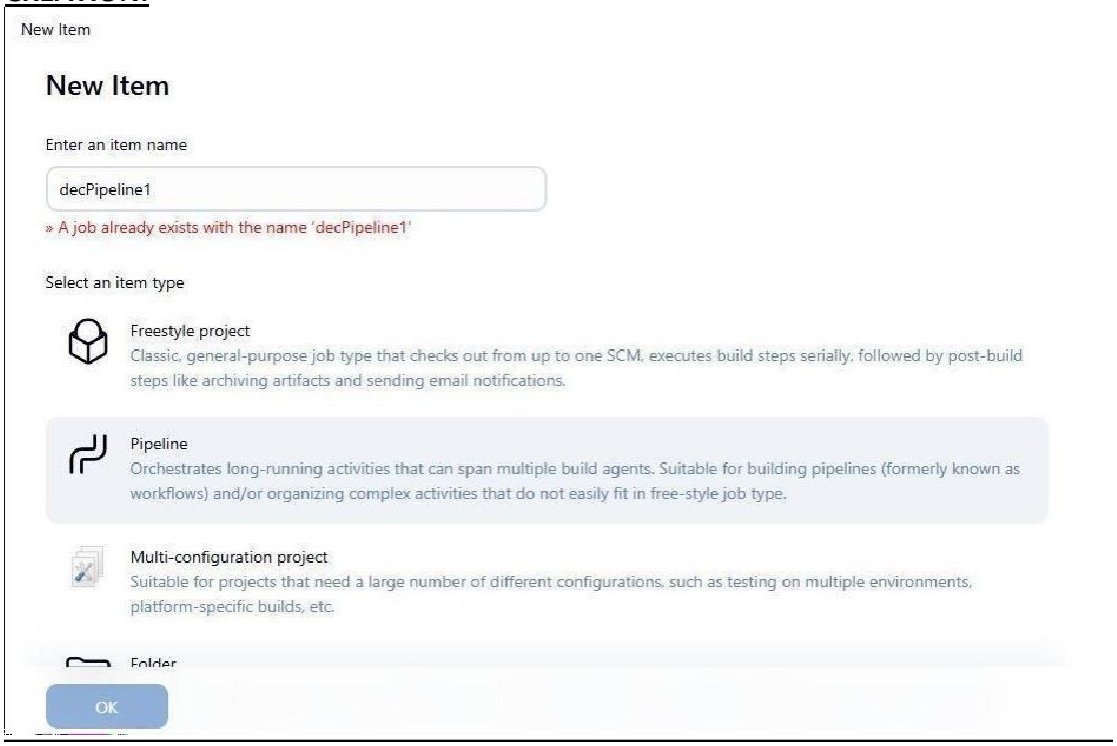
- More flexible and powerful
- Written in Groovy-based syntax
- Gives full control over the build process
- Suitable for advanced users Key

Features:

- Uses node {} block
- Everything is written as code
- Allows complex scripting and logic
- Less user-friendly for beginners

1. Declarative Pipeline

CREATION:



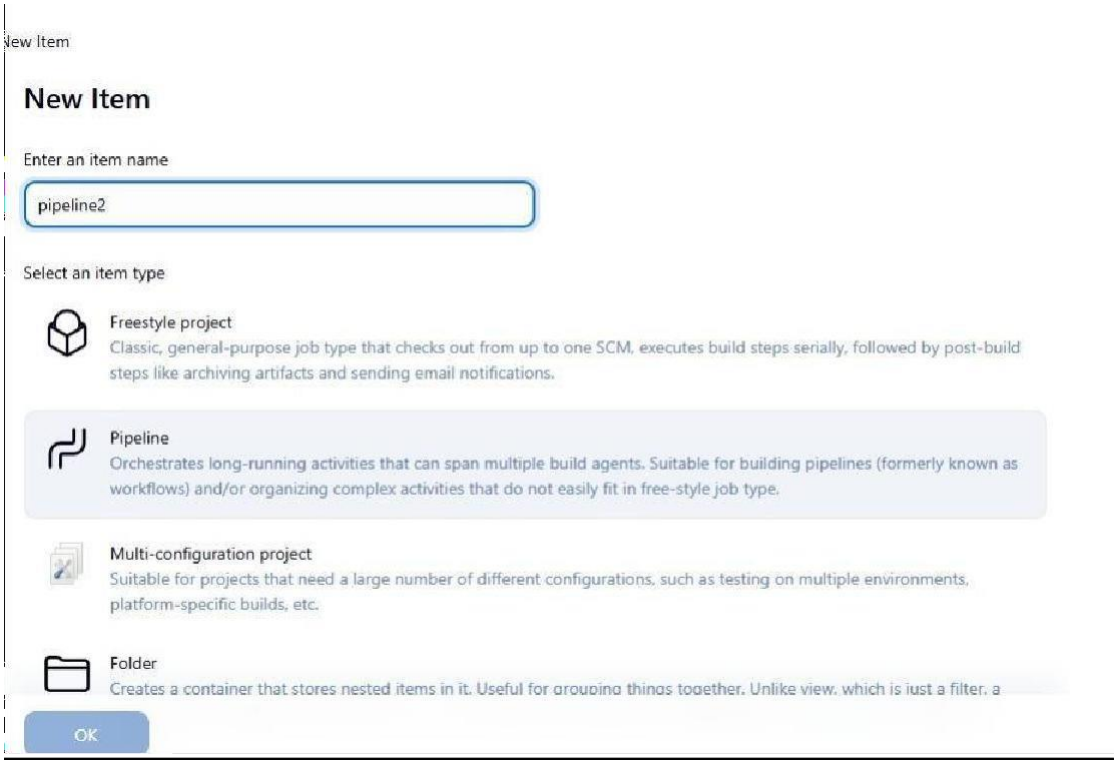
CODE:

```
pipeline {       agent any       stages {
stage('Build') {                steps {
echo 'Building..'
            }       }       stage('Test') {
   steps {                  echo 'Testing...'
          }       }       stage('Deploy') {
 steps {                  echo 'Deploying...'
         }
      }
   }
}
```

OUTPUT:

```
Running on Jenkins in C:\ProgramData\Jenkins\.jenkins\workspace\decPipeline1
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] echo
Building..
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Test)
[Pipeline] echo
Testing...
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy)
[Pipeline] echo
Deploying...
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

2. SCRIPTED CREATION:



CODE:

```
node {
stage('Build') {
//      }
stage('Test') {
        //      }
 stage('Deploy') {
//
    }
}
```

OUTPUT:

CONCLUSION: LO1, LO3

In this assignment, we learned how to create Jenkins Pipelines to automate the build, test, and deployment process. We explored both Declarative and Scripted pipelines, understanding their structure, benefits, and use cases. By implementing different pipeline examples, we demonstrated how Jenkins enables efficient, repeatable, and reliable CI/CD workflows.

LO MAPPING: LO1,LO3 Mapped.