

LAB ASSIGNMENT 6

AIM: To build a Pipeline using Jenkins.

LAB OUTCOME:

LO1, LO3 Mapped.

THEORY:

A Jenkins Pipeline is a method of defining software development workflows as code. It lets you describe your entire build, test, and deployment process in a structured manner. With Jenkins Pipelines, you can choose between Declarative and Scripted syntax to specify your automation steps. These pipelines are versioned alongside your application code in your version control system, ensuring consistency and reproducibility. Pipelines offer features like parallel execution of tasks, integration with numerous plugins, and the ability to encapsulate reusable components. They bring transparency through detailed logs, visualise the flow of tasks, and facilitate advanced automation practices like Continuous Integration and Continuous Delivery.

Continuous Delivery (CD) pipelines automate the steps involved in getting software changes from development to production. They include building code, running tests, deploying to staging, and potentially deploying to production. CD pipelines ensure that software updates are thoroughly tested, reducing errors and allowing for rapid and reliable delivery. They involve stages such as automated testing, staging environment validation, and production deployment, all supported by automation. By automating these steps, CD pipelines streamline software delivery, enhance collaboration, and enable faster response to user needs.

A Jenkinsfile is a text file used to define and describe the stages and steps of a Jenkins Pipeline. It's written in Groovy, a versatile scripting language that allows you to express complex automation workflows. The Jenkinsfile is stored within your version control repository alongside your application code, enabling versioning and consistency between code and pipeline changes.

There are two main syntax options for writing Jenkinsfiles: Declarative and Scripted.

Declarative Pipeline Syntax:

Declarative pipelines offer a simplified way to define pipelines. They focus on describing the high-level structure of the pipeline and its stages. Declarative pipelines are less verbose and are designed to be easy to read and understand. They follow a structured syntax and provide predefined directives for commonly used stages and steps.

Example:

```
pipeline {  
    agent any  
    stages {  
        stage('Build') {
```

```
    steps {
        sh 'echo "Building"'
    }
}

stage('Test') {
    steps {
        sh 'echo "Testing"'
    }
}

stage('Deploy') {
    steps {
        sh 'echo "Deploying"'
    }
}
}
```

Scripted Pipeline Syntax:

Scripted pipelines provide more flexibility and control over your pipeline workflow. With Scripted pipelines, you write custom Groovy scripts to define each stage and step. This syntax is more powerful but can be more verbose and complex compared to the Declarative syntax. Scripted pipelines are suitable for scenarios where you need precise control over the order and execution of tasks.

Example:

```
node {
    stage('Build') {
        echo "Building"
    }

    stage('Test') {
        echo "Testing"
    }

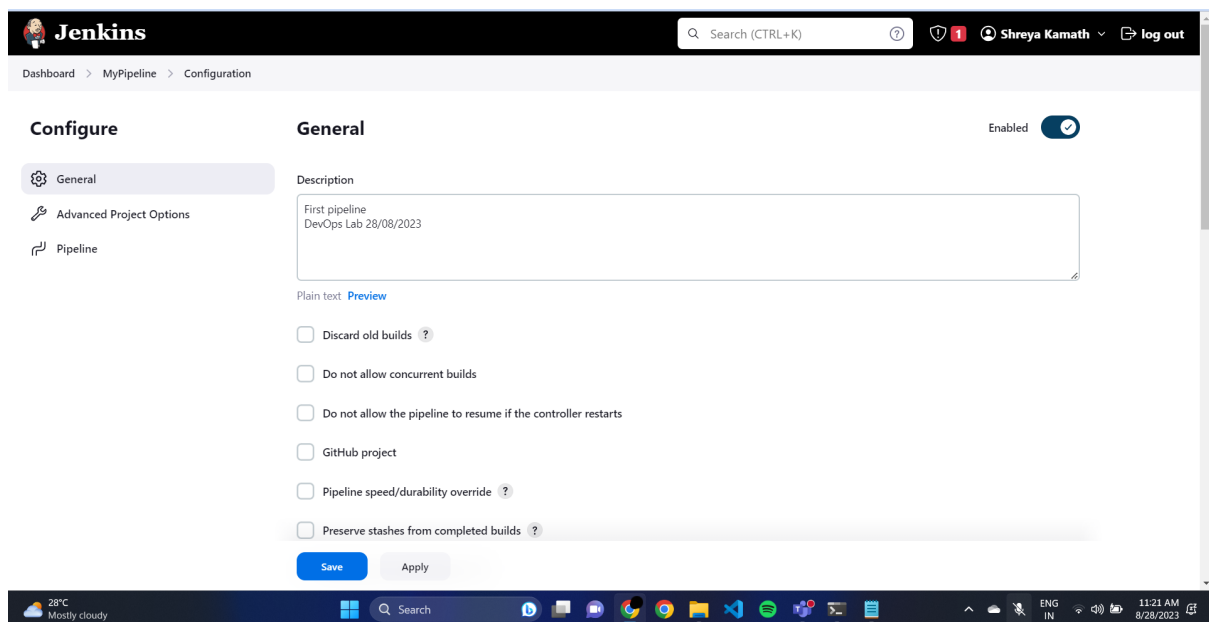
    stage('Deploy') {
        echo "Deploying"
    }
}
```

```
}  
  
}
```

Benefits of Jenkins Pipeline:

1. Consistency: Pipelines provide a uniform process for building, testing, and deploying code, reducing variability and ensuring reliability.
2. Efficiency: Automation streamlines tasks, minimising manual effort and accelerating software delivery.
3. Versioned Control: Jenkinsfiles are versioned, keeping automation in sync with code changes and aiding collaboration.
4. Adaptability: Pipelines offer adaptable syntax options (Declarative and Scripted) to match your workflow complexity.

OUTPUT:



Shreya Kamath / DevOps Lab / T-13 / Roll no. 53

The screenshot shows the Jenkins web interface for a pipeline named 'MyPipeline'. The left sidebar contains navigation links: Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, Rename, and Pipeline Syntax. The main content area displays the 'MyPipeline' details, including its first run on 28/08/2023. A 'Stage View' section shows a build history table with columns for build number, time, and status. The table shows build #1 on Aug 28 at 11:16 with a status of 'No Changes'. A 'Hello' stage is highlighted with a green box, showing an average stage time of 793ms. Below the stage view, there are 'Permalinks' and a 'Build History' section with a 'trend' dropdown and a 'Filter builds...' input. The bottom of the page shows a Windows taskbar with the date and time as 11:20 AM on 8/28/2023.

Dashboard > MyPipeline >

MyPipeline

First pipeline
DevOps Lab 28/08/2023

Edit description
Disable Project

Stage View

Average stage times:
(Average full run time: ~7s)

Build	Time	Status
#1	Aug 28 11:16	No Changes

793ms

Permalinks

REST API Jenkins 2.420

The screenshot shows the Jenkins web interface for a specific build, 'Build #2 (28-Aug-2023, 11:20:36 am)'. The left sidebar contains navigation links: Status, Changes, Console Output, Edit Build Information, Delete build '#2', Restart from Stage, Replay, Pipeline Steps, Workspaces, and Previous Build. The main content area displays the build details, including its status (green checkmark), started by user 'Shreya Kamath', and a 'Keep this build forever' button. The 'Console Output' section shows the build log, which includes the text 'Started by user Shreya Kamath' and 'Took 3.7 sec'. The bottom of the page shows a Windows taskbar with the date and time as 11:20 AM on 8/28/2023.

Dashboard > MyPipeline > #2

Build #2 (28-Aug-2023, 11:20:36 am)


Keep this build forever





Add description Started 10 sec ago Took 3.7 sec

Started by user Shreya Kamath


REST API Jenkins 2.420


Shreya Kamath / DevOps Lab / T-13 / Roll no. 53


 **Jenkins**


Search (CTRL+K)    Shreya Kamath  log out


Dashboard > MyPipeline > #2


 Status


 Changes


 Console Output


 View as plain text


 Edit Build Information


 Delete build '#2'


 Restart from Stage

 Replay

 Pipeline Steps

 Workspaces

 Previous Build

 **Console Output**

Started by user [Shreya Kamath](#)

[Pipeline] Start of Pipeline

[Pipeline] node

Running on [Jenkins](#) in C:\ProgramData\Jenkins\workspace\MyPipeline

[Pipeline] {

[Pipeline] stage

[Pipeline] { (Hello)

[Pipeline] echo

Hello World

[Pipeline] }

[Pipeline] // stage

[Pipeline] }


[Pipeline] // node


[Pipeline] End of Pipeline

Finished: SUCCESS

REST API Jenkins 2.420

28°C
Mostly cloudy


 Search





ENG
IN 11:21 AM
8/28/2023

Dashboard > MyPipeline > Configuration


Configure

 General

 Advanced Project Options


 Pipeline


Advanced Project Options

Advanced 

Pipeline

Definition

Pipeline script 

Script 

```
1 * pipeline {
2
3   agent any
4
5   stages { stage('Build') { steps { echo 'Hi, GeekFlare. Starting to build the App.' } } stage('Test') { stage('Deploy start ') {
6
7   }
8   }
9   }
10  }
11  steps {
12
13    input('Do you want to proceed?')
14
15  }
16
17 }
```

try sample Pipeline...

Save

Apply

28°C
Mostly cloudy

 Search



ENG
IN 11:23 AM
8/28/2023

Dashboard > MyPipeline > Status

mypipeline

First pipeline
DevOps Lab 28/08/2023

Edit description
Disable Project

Stage View

Average stage times: (Average full run time: ~5s)		Hello
#3 Aug 28 11:23 No Changes		941ms
#2 Aug 28 11:20 No Changes		1s
#1 Aug 28 11:16 No Changes		793ms

Permalinks

Next build (#3) in 2 minutes

28°C Mostly cloudy

Search

11:23 AM 8/28/2023

Shreya Kamath / DevOps Lab / T-13 / Roll no. 53

The top screenshot shows the Jenkins dashboard for 'MyPipeline' > '#3'. The 'Status' tab is selected, showing a red 'X' icon and the text 'Build #3 (28-Aug-2023, 11:23:42 am)'. The build was started by user 'Shreya Kamath' 15 seconds ago and took 0.45 seconds. The console output is visible, showing a Groovy script and a stack trace of errors.

The bottom screenshot shows the 'Console Output' tab for the same build. The console output displays a Groovy script and a stack trace of errors. The script defines a pipeline with two stages: 'Build' and 'Test'. The 'Build' stage contains a 'Hello World' step. The 'Test' stage contains a 'Deploy' step. The stack trace shows an error in the 'Test' stage, specifically in the 'Deploy' step.

CONCLUSION:

Through this assignment, I have learnt the concept of Pipeline and Continuous Delivery (CD) on Jenkins, and successfully executed a pipeline printing “Hello World”.