Deep Chavan

T11-15

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:

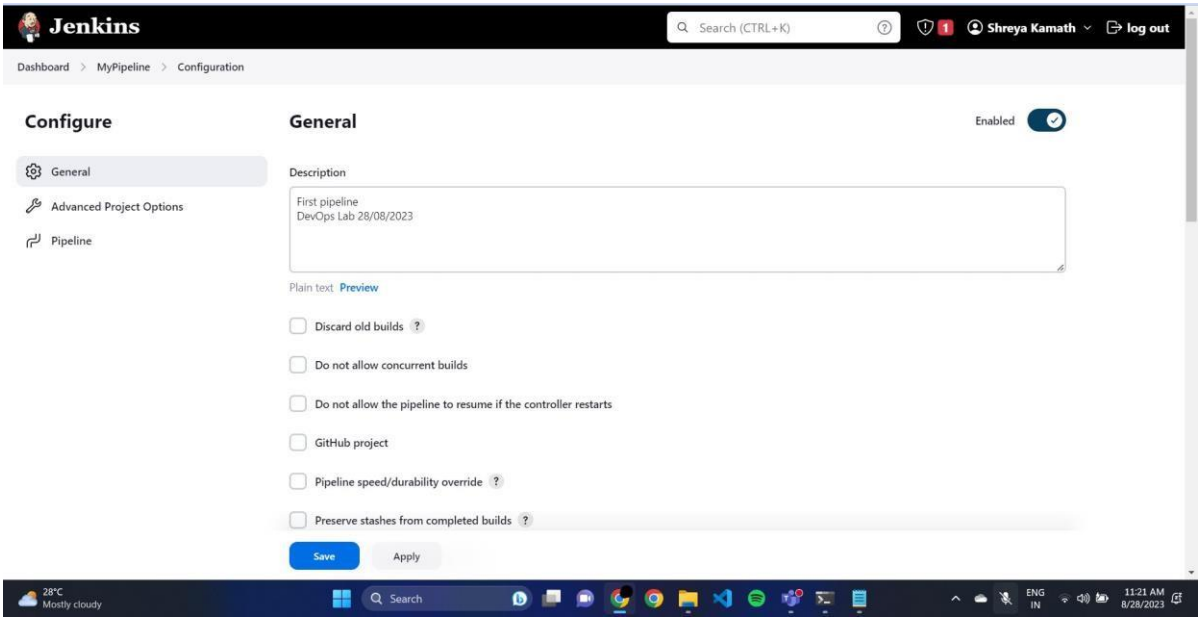
1. Consistency: Pipelines provide a uniform process for building, testing, and deploying code, reducing variability and ensuring reliability.

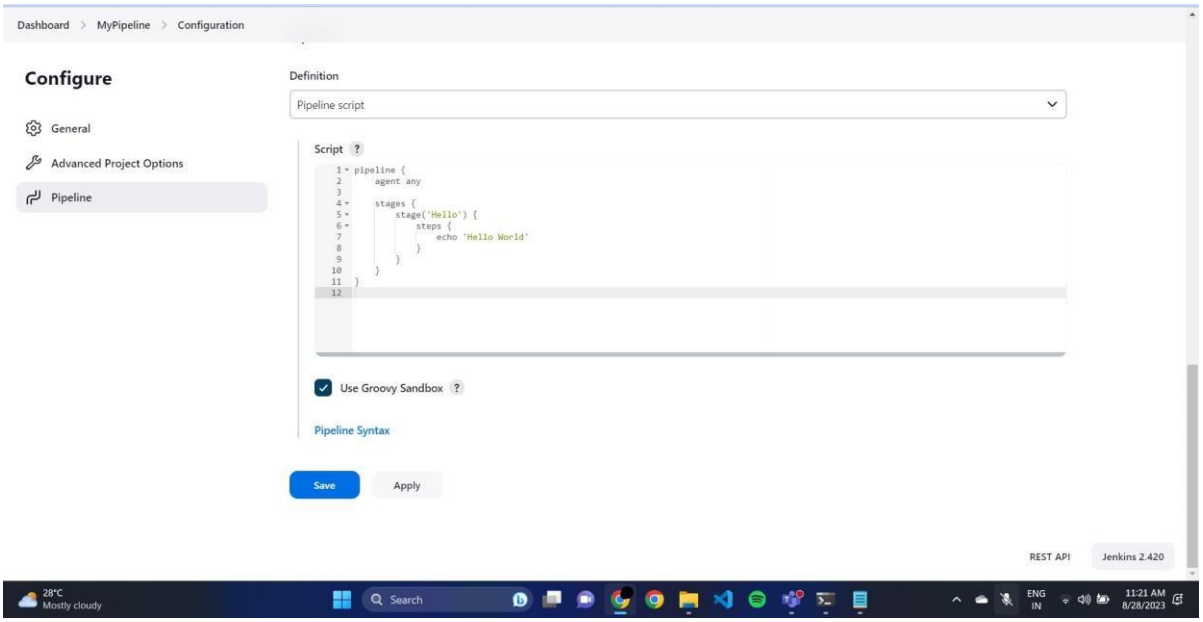
2. Efficiency: Automation streamlines tasks, minimising manual effort and accelerating software delivery.

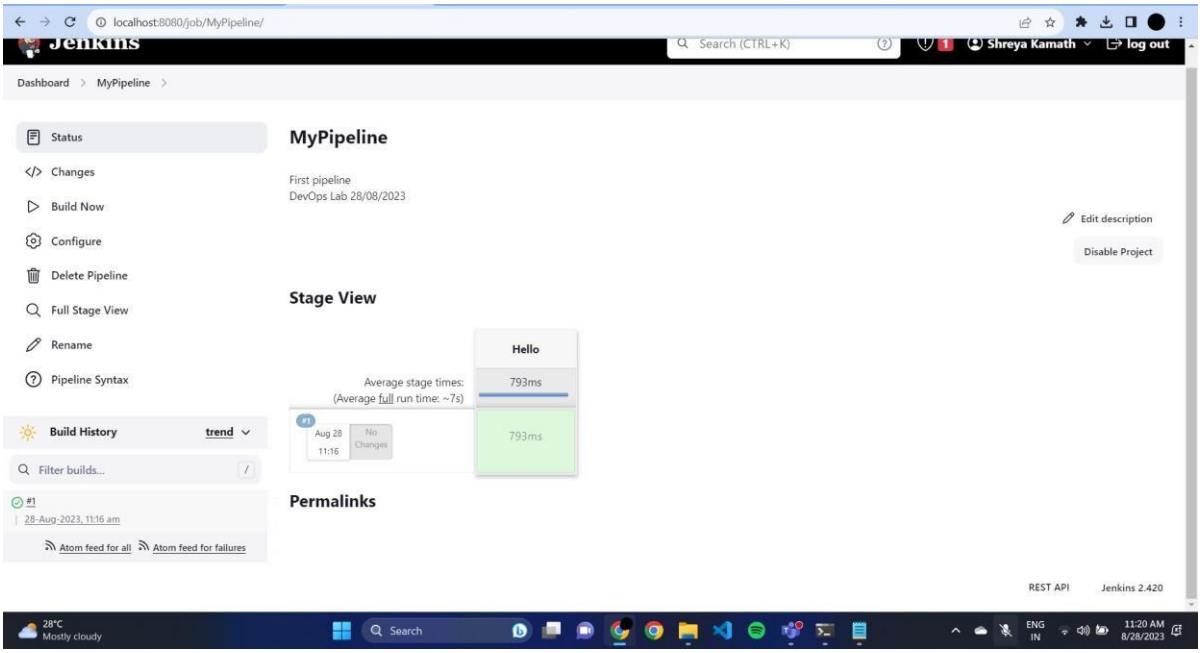
3. Versioned Control: Jenkinsfiles are versioned, keeping automation in sync with code changes and aiding collaboration.

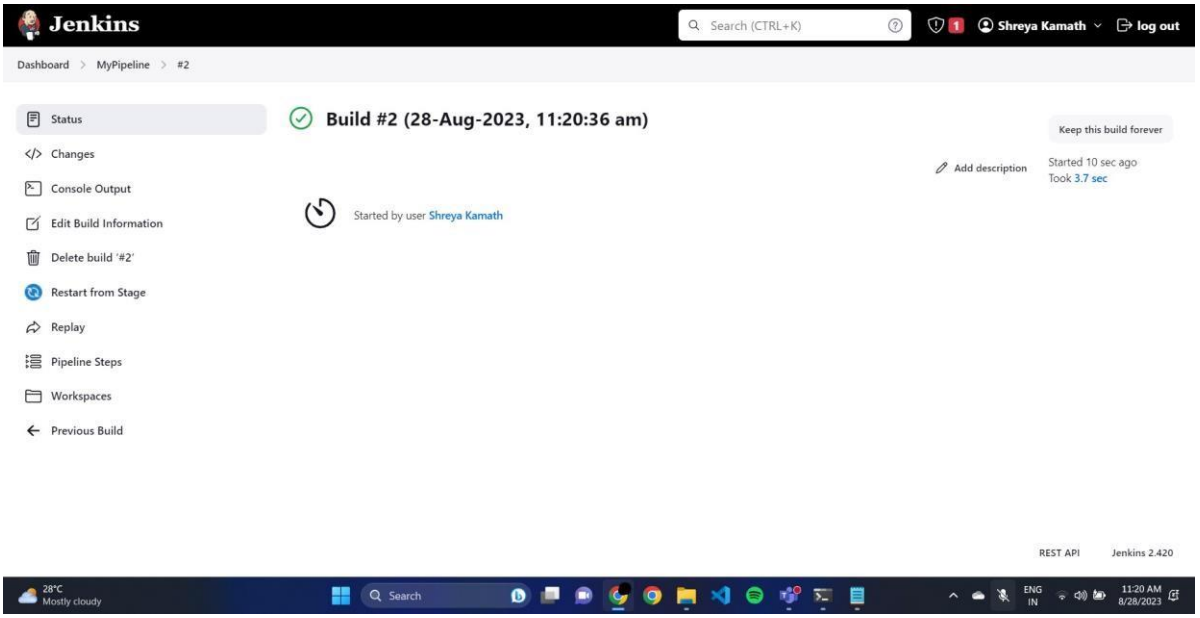
4. Adaptability: Pipelines offer adaptable syntax options (Declarative and Scripted) to match your workflow complexity.

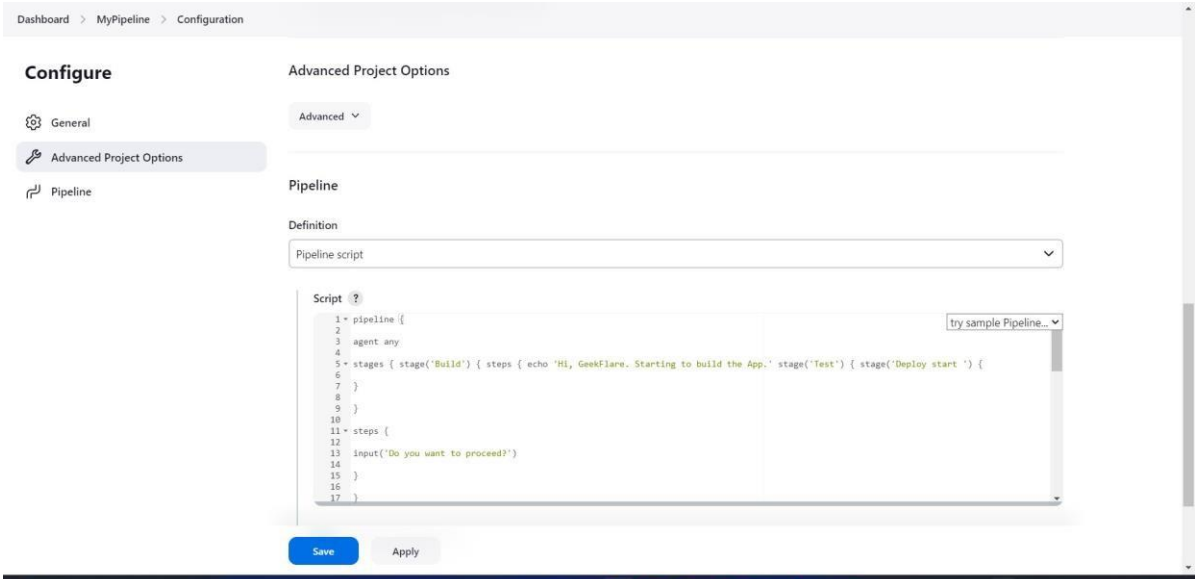
OUTPUT:



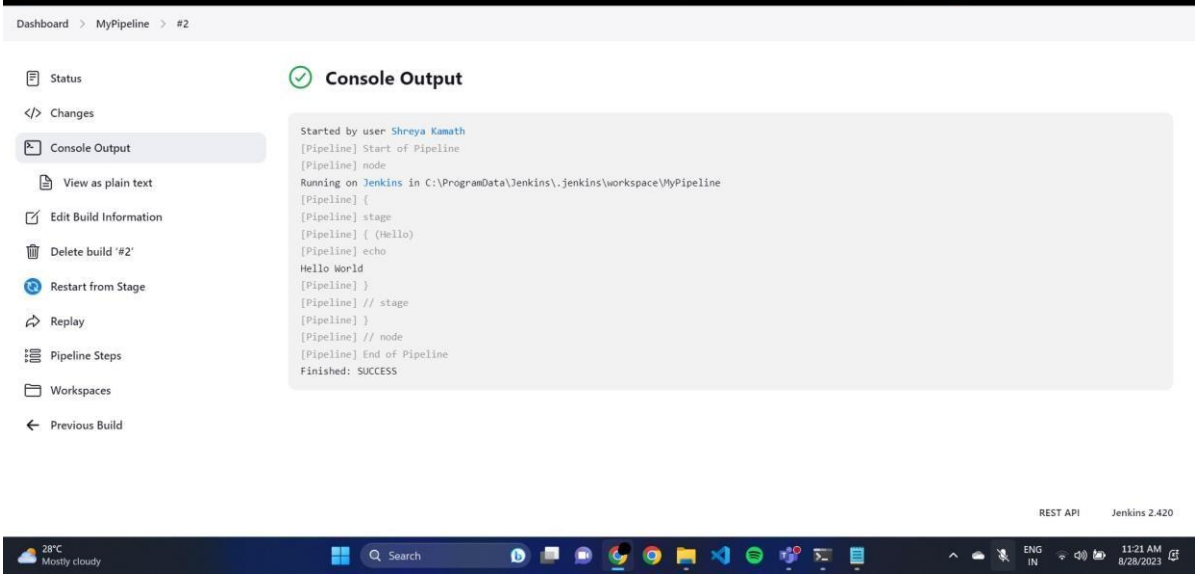

Dashboard > MyPipeline > Configuration
Configure
General
Advanced Project Options
Pipeline
Definition
Pipeline script
Script
pipeline {
agent any
stages {
stage('Hello') {
steps {
echo 'Hello World'
}
}
}
}
Use Groovy Sandbox
Pipeline Syntax
Save
Apply
REST API
Jenkins 2.420
28°C
Mostly cloudy
Search
ENG
IN
11:21 AM
8/28/2023


localhost:8080/job/MyPipeline/
Jenkins
Search (CTRL+K)
Shreya Kamath
log out
Dashboard > MyPipeline >
Status
Changes
Build Now
Configure
Delete Pipeline
Full Stage View
Rename
Pipeline Syntax
Build History
trend
Filter builds...
#1
28-Aug-2023, 11:16 am
Atom feed for all
Atom feed for failures
MyPipeline
First pipeline
DevOps Lab 28/08/2023
Edit description
Disable Project
Stage View
Hello
Average stage times:
(Average full run time: ~7s)
793ms
Aug 28
11:16
No Changes
793ms
Permalinks
REST API
Jenkins 2.420
28°C
Mostly cloudy
Search
ENG
IN
11:20 AM
8/28/2023

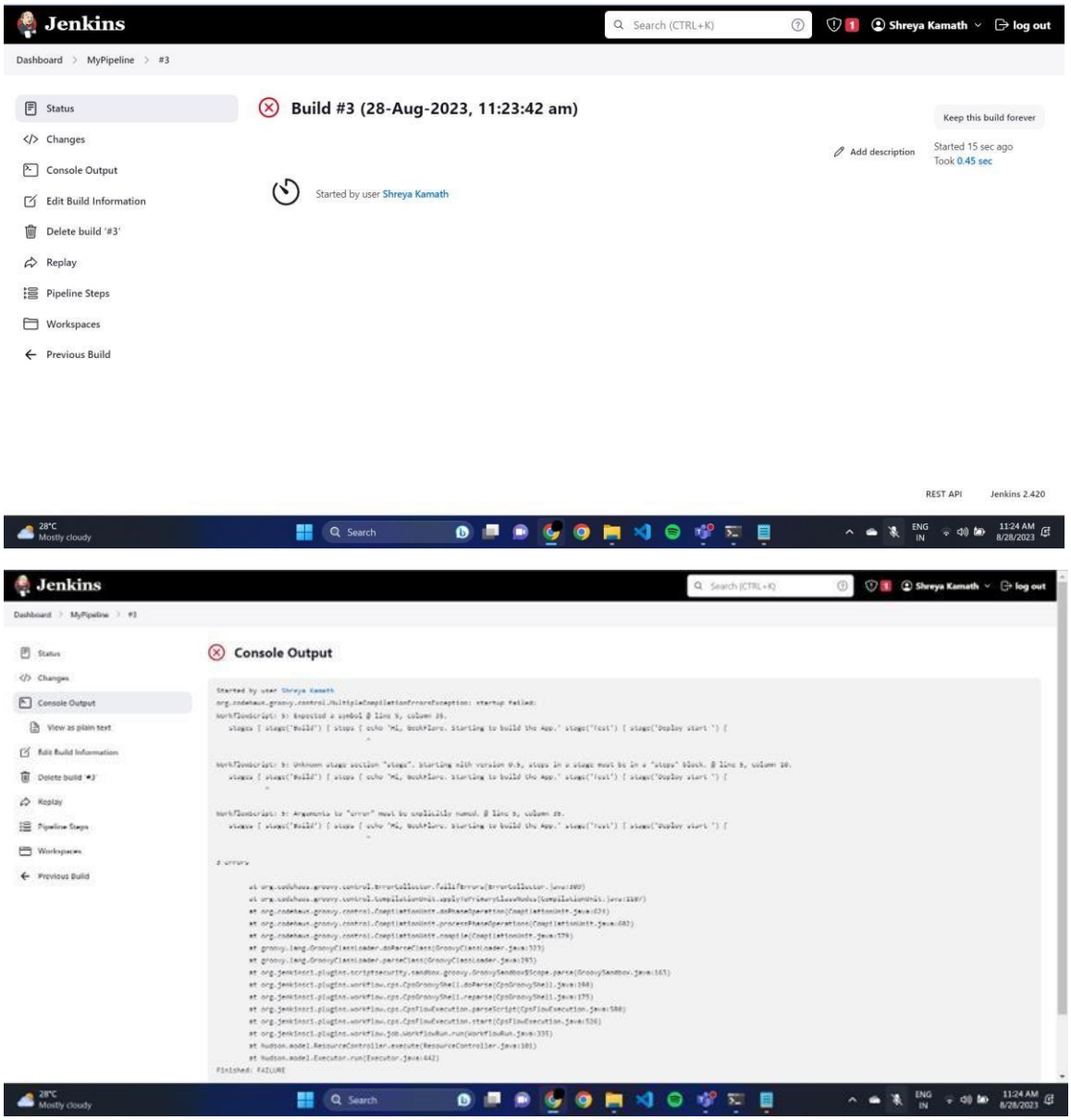

Jenkins
Search (CTRL+K)
Shreya Kamath
log out
Dashboard > MyPipeline > #2
Status
Changes
Console Output
Edit Build Information
Delete build '#2'
Restart from Stage
Replay
Pipeline Steps
Workspaces
Previous Build
Build #2 (28-Aug-2023, 11:20:36 am)
Keep this build forever
Add description
Started 10 sec ago
Took 3.7 sec
Started by user Shreya Kamath
REST API
Jenkins 2.420
28°C
Mostly cloudy
Search
ENG
IN
11:20 AM
8/28/2023


Dashboard > MyPipeline > Configuration
Configure
General
Advanced Project Options
Pipeline
Advanced Project Options
Advanced
Pipeline
Definition
Pipeline script
Script
try sample Pipeline...
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 }
Save
Apply


28°C
Mostly cloudy
Search
ENG
IN
11:23 AM
8/28/2023


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\.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



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”.