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:

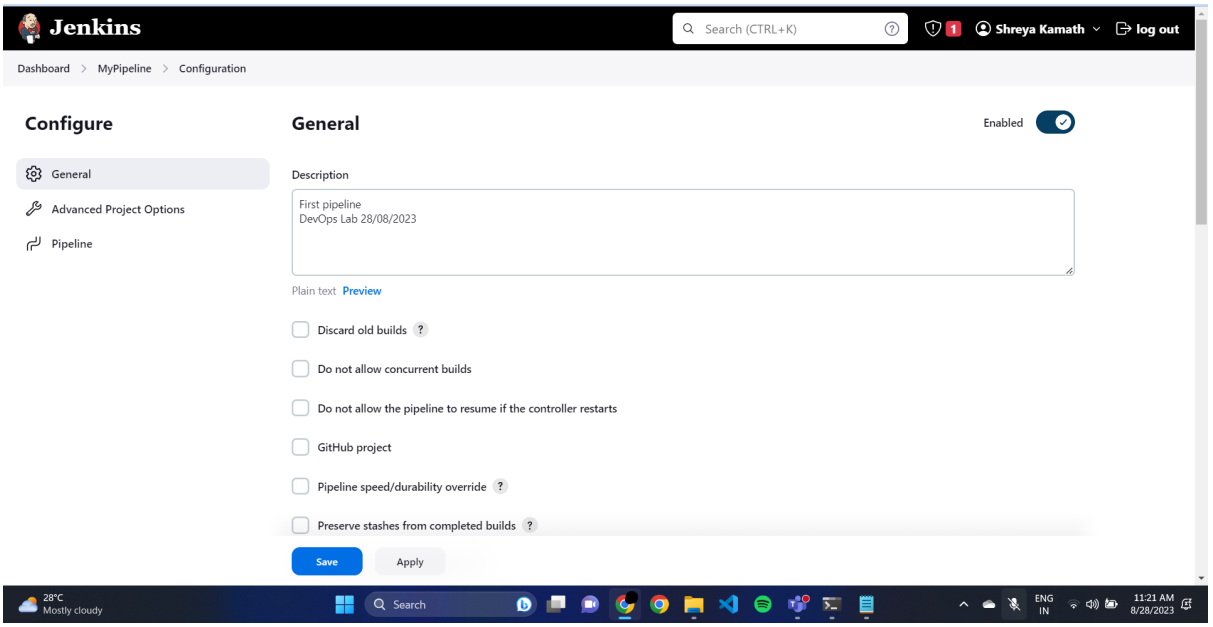
1. Consistency: Pipelines provide a uniform process for building, testing, and deploying code, reducing variability and ensuring reliability.

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

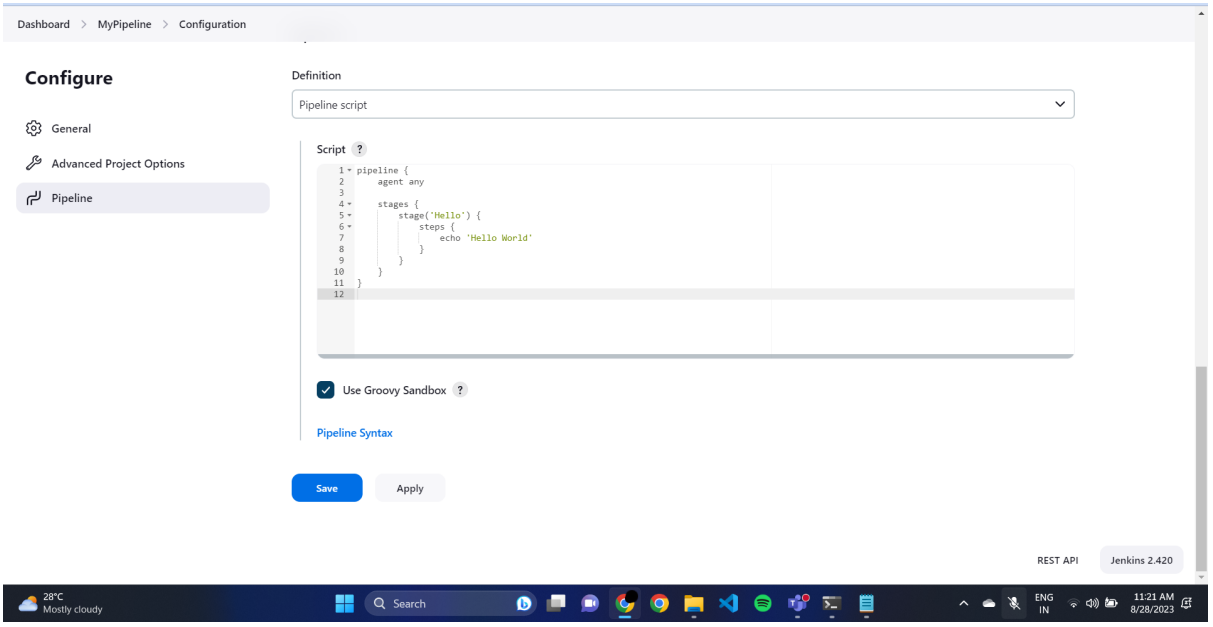
3. Versioned Control: Jenkinsfiles are versioned, keeping automation in sync with code changes and aiding collaboration.

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

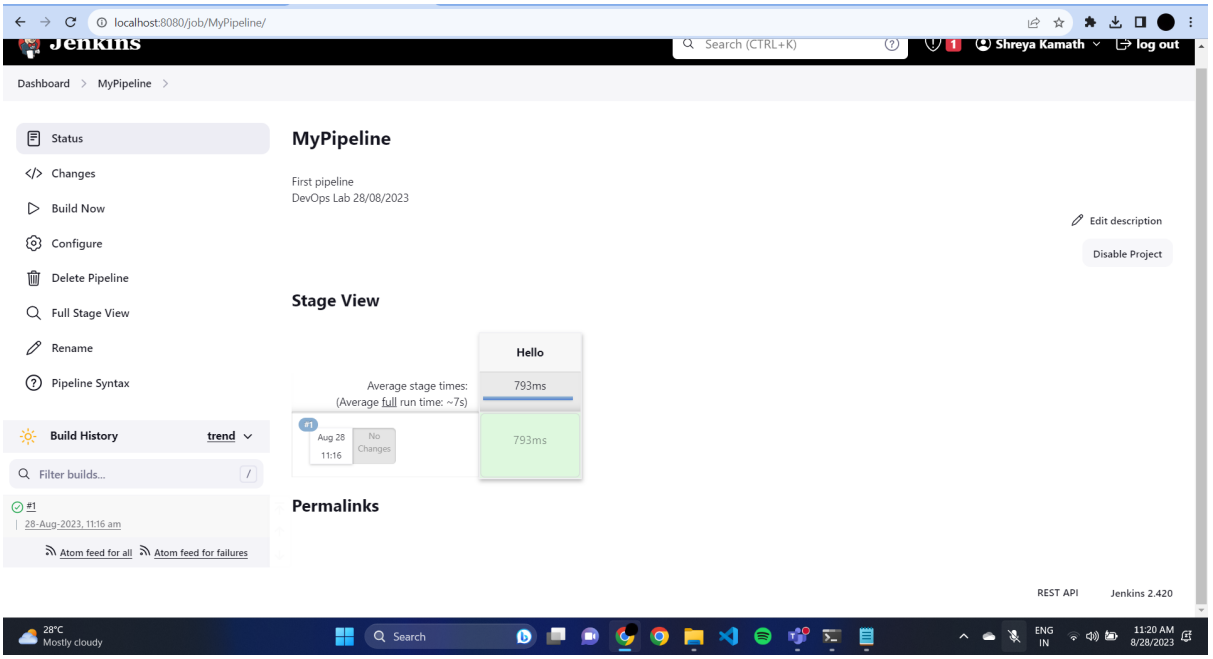
OUTPUT:

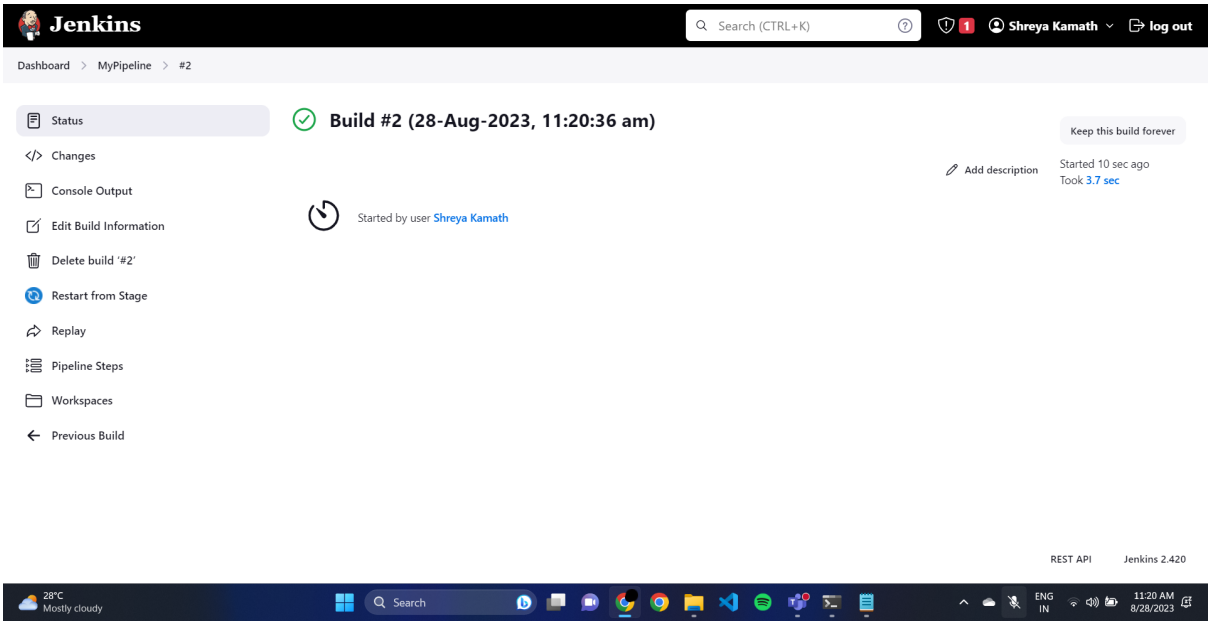


Shreya Kamath / DevOps Lab / T-13 / Roll no. 53

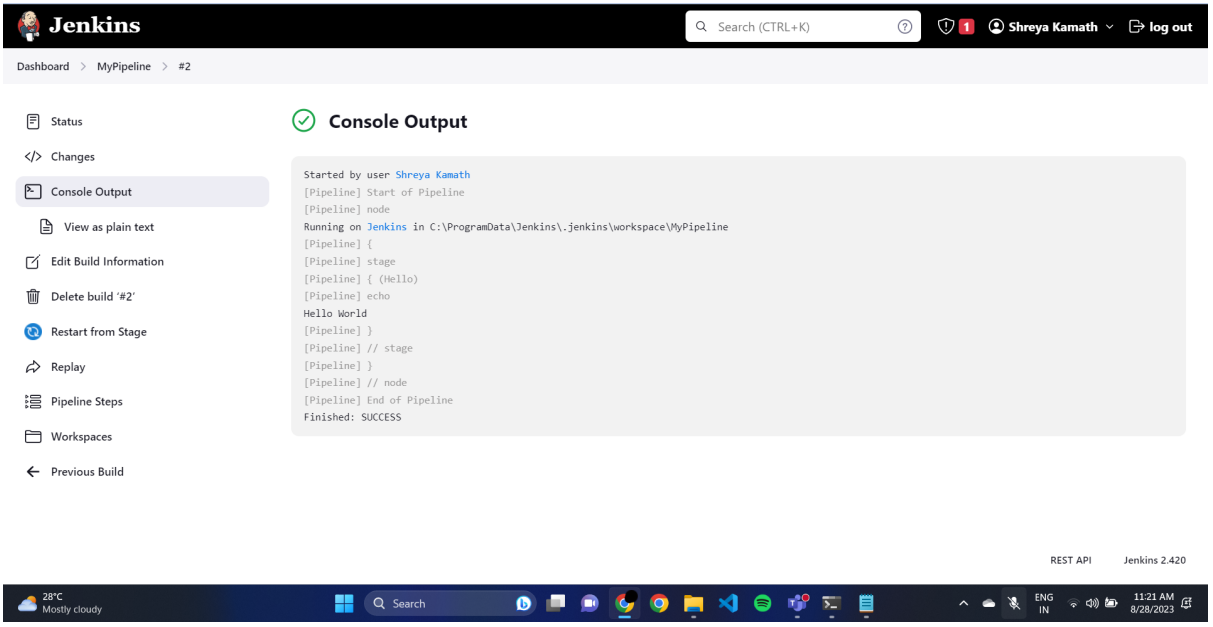


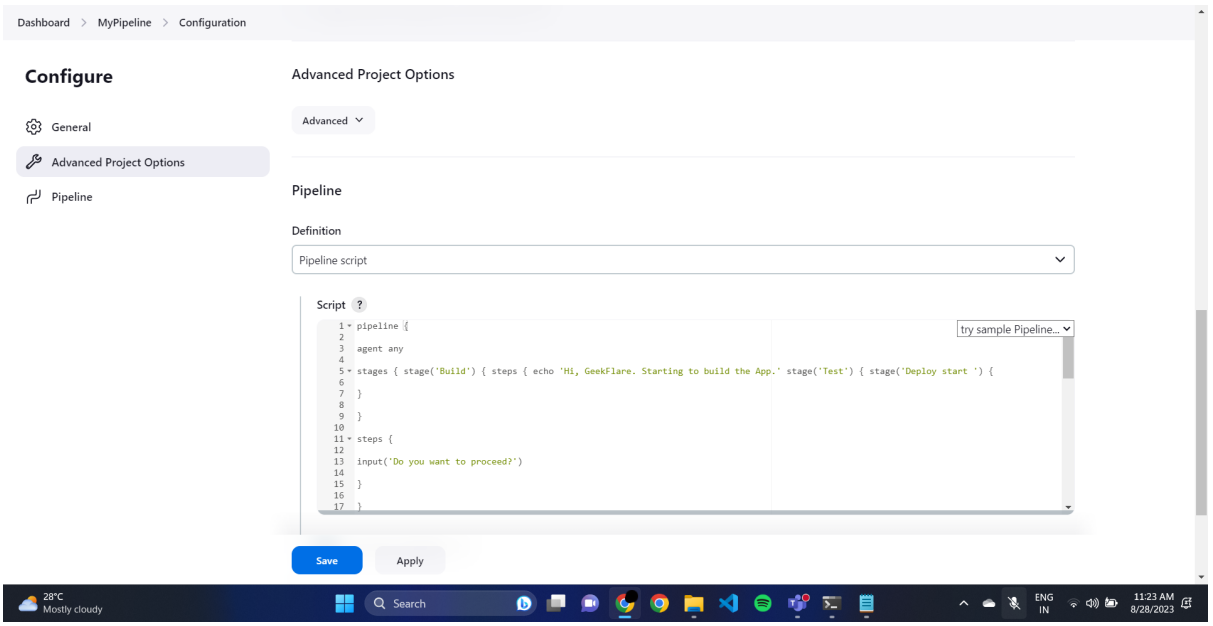
Shreya Kamath / DevOps Lab / T-13 / Roll no. 53



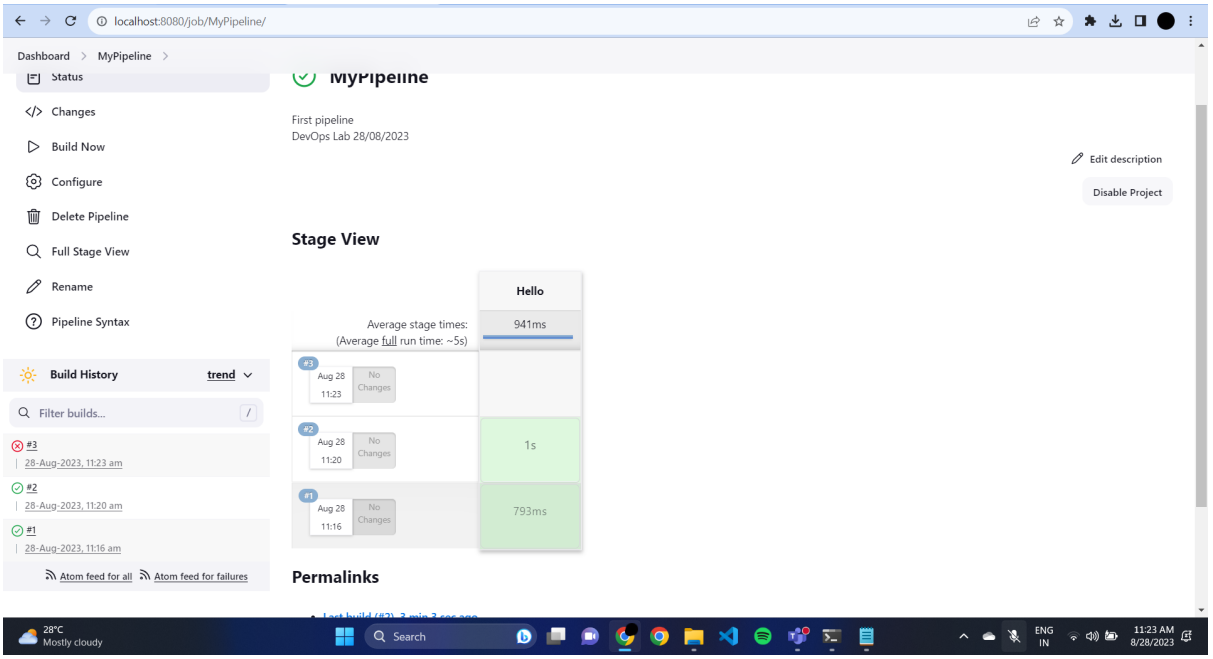


Shreya Kamath / DevOps Lab / T-13 / Roll no. 53

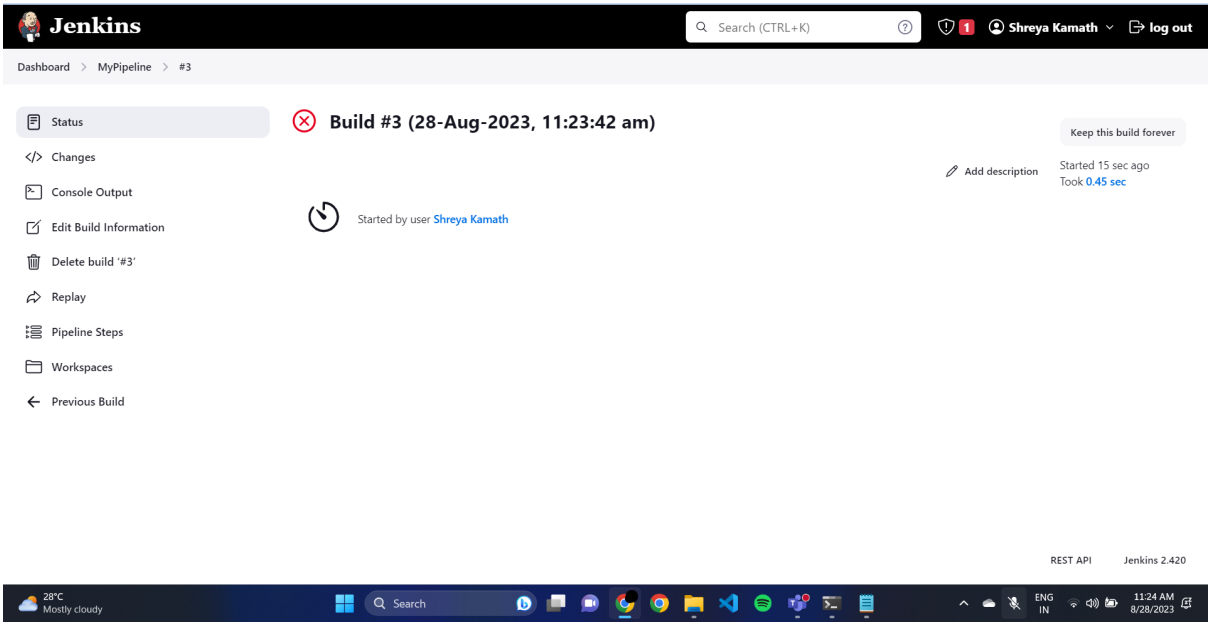


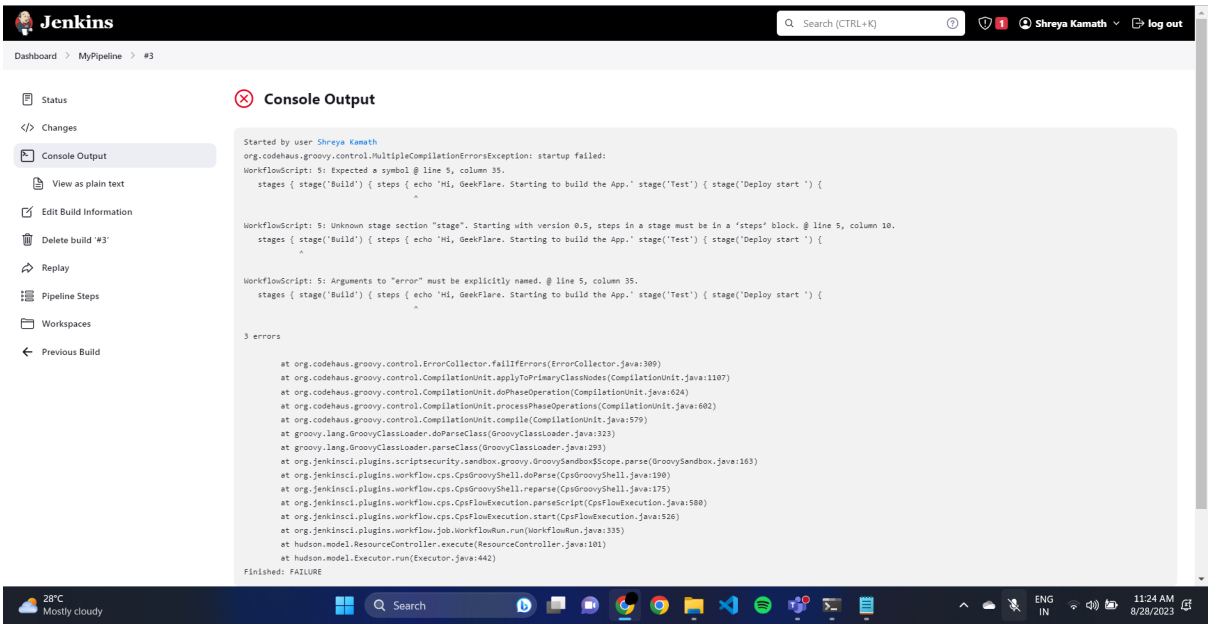


Shreya Kamath / DevOps Lab / T-13 / Roll no. 53



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





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”.