SOFTWARE ENGINEERING

UNIT – 3

TOPIC – 11

BUILD PIPELINE PROJECT USING JENKIN SCRIPT

Description

Jenkins is a leading tool for automating software development processes, especially Continuous Integration (CI) and Continuous Delivery (CD). A Jenkins Pipeline defines and automates the stages needed to build, test, and deploy software projects.

Introduction to Jenkins Script

A Jenkins script is a set of commands that automates tasks in Jenkins. These scripts are used to manage various stages of software development, from retrieving code to deploying the final application. Jenkins scripts are crucial for creating a CI/CD pipeline, which helps in automating repetitive tasks, reducing errors, and ensuring consistency.

Common Tasks Performed by a Jenkins Script:

1. Retrieve the latest source code from a version control system like Git.
2. Compile the code into executable files.
3. Run automated tests to ensure code quality.
4. Package the application for deployment.
5. Deploy the application to a testing or production environment.

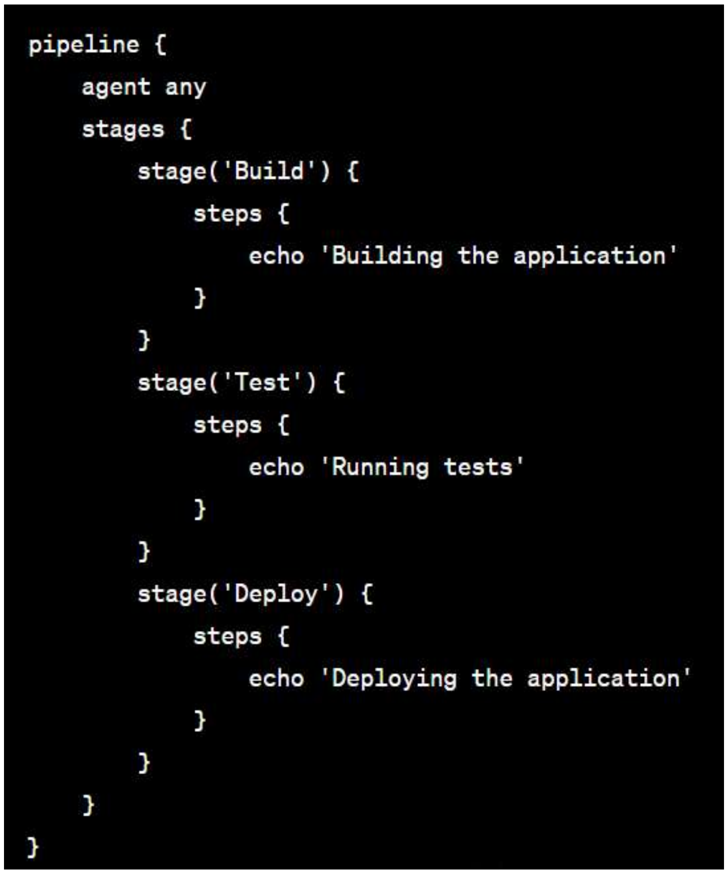
Using scripts streamlines the software delivery process, saves time, and maintains consistent workflows.

Types of Jenkins Scripts

There are two primary types of scripts used in Jenkins for defining pipelines:

1. Declarative Pipeline Syntax

- A structured and straightforward way of writing pipelines.
- Focuses on defining stages and steps in a readable format.
- **Key Features**:
    - Easy to read and understand.
    - Suitable for those new to scripting.
- **Use Case**:
    - Best for simple pipelines that don't require complex logic.

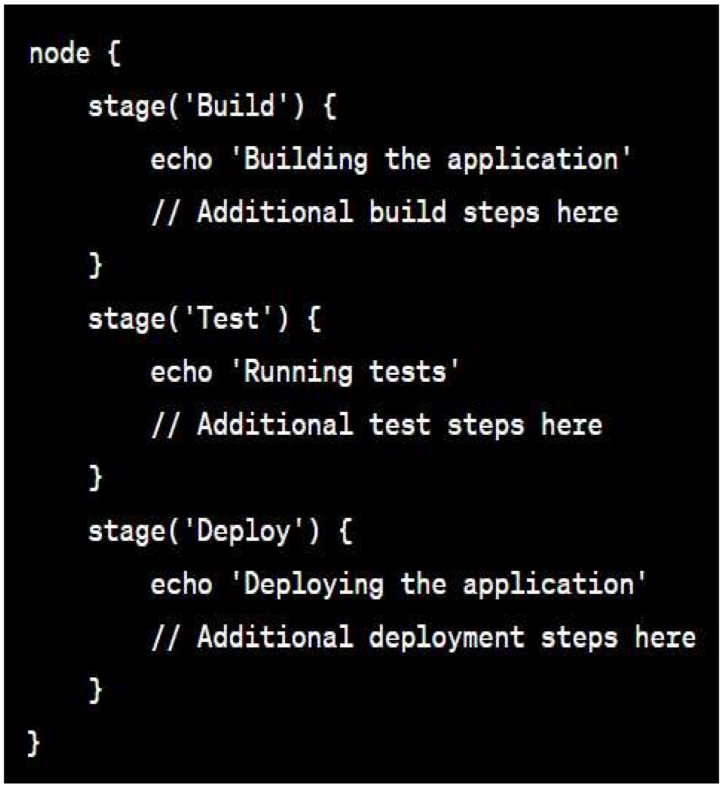


```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                echo 'Building the application'
            }
        }
        stage('Test') {
            steps {
                echo 'Running tests'
            }
        }
        stage('Deploy') {
            steps {
                echo 'Deploying the application'
            }
        }
    }
}
```

- **Example**:
    - Configuration-style scripting that defines what needs to happen in each stage.

2. Scripted Pipeline Syntax

- A more flexible and powerful way of writing pipelines using a Groovy-based scripting language.
- Offers advanced control over pipeline behavior.
- **Key Features**:
    - Flexibility and extensive customization.
    - Suitable for users who need detailed control over the pipeline.
- **Use Case**:
    - Ideal for complex projects requiring advanced CI/CD features.



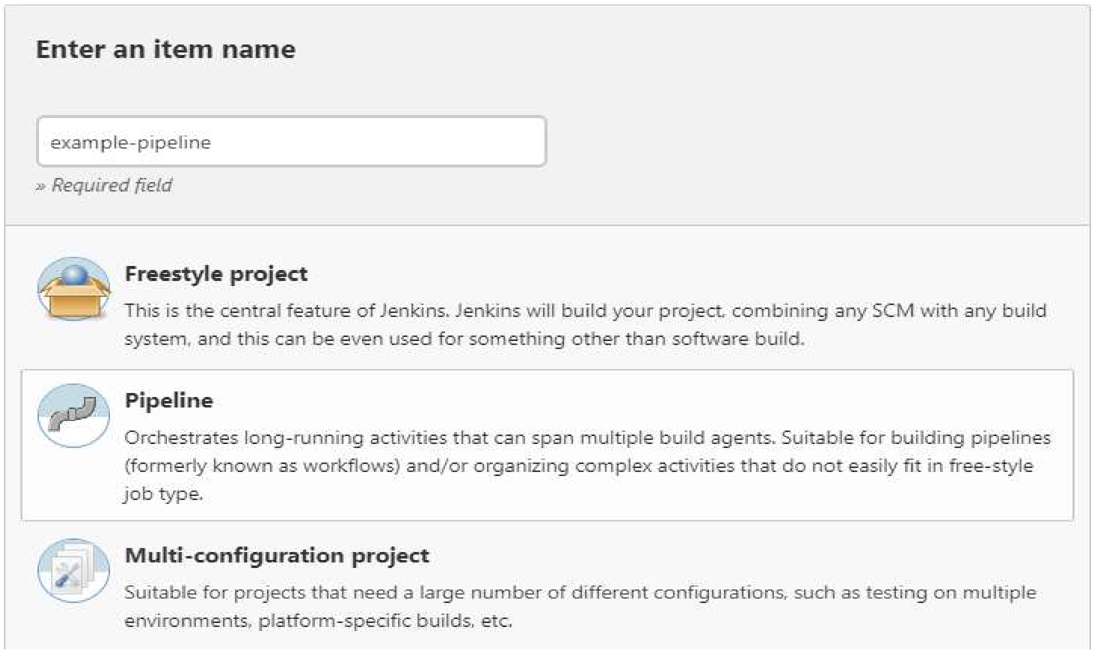
```
node {
    stage('Build') {
        echo 'Building the application'
        // Additional build steps here
    }
    stage('Test') {
        echo 'Running tests'
        // Additional test steps here
    }
    stage('Deploy') {
        echo 'Deploying the application'
        // Additional deployment steps here
    }
}
```

- **Example**:
    - Code-like scripting that allows conditional logic and custom workflows.

The choice between **Declarative** and **Scripted** syntax depends on the project's complexity and the level of customization needed.

Types of Jenkins Pipelines

Jenkins supports two major types of pipelines, which manage the CI/CD process:



Enter an item name

example-pipeline

» Required field

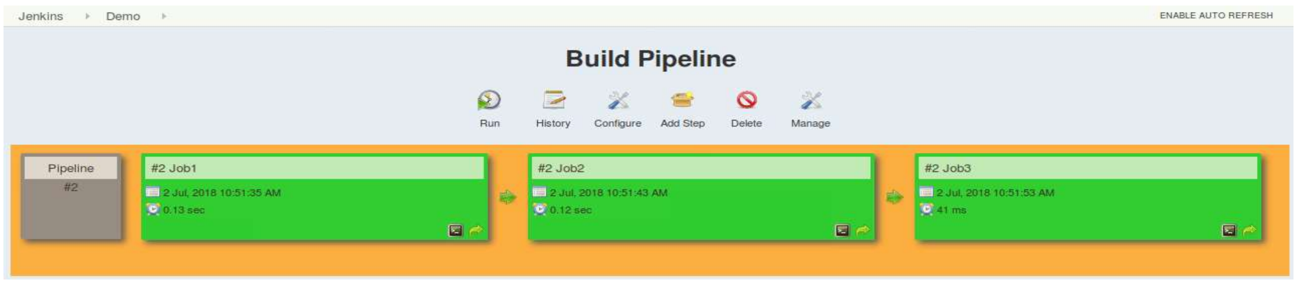
Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

1. Freestyle Projects

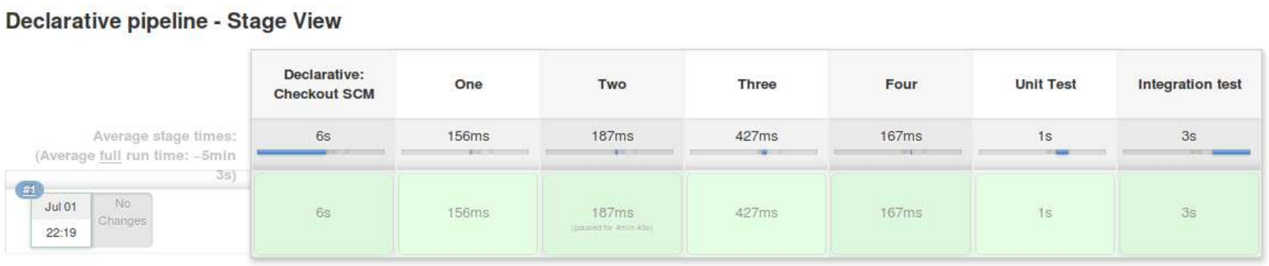
- The traditional and simpler way to create jobs in Jenkins using a graphical interface.
- **Key Features**:
  - Easy to set up using the web interface.
  - Suitable for simple build and deployment tasks.
- **Use Case**:
  - Ideal for basic CI/CD projects with straightforward requirements.



- **Example**:
  - Creating a job using a visual interface without writing code.

2. Pipeline Projects

- A more advanced and flexible method introduced with the Pipeline plugin.
- **Key Features**:
  - Uses Groovy-based scripts to define the pipeline.
  - Supports version control and complex workflows.
- **Use Case**:
  - Perfect for advanced projects that require dynamic and code-based pipeline configurations.


Declarative pipeline - Stage View

	Declarative: Checkout SCM	One	Two	Three	Four	Unit Test	Integration test
Average stage times: (Average full run time: ~5min 3s)	6s	156ms	187ms	427ms	167ms	1s	3s
#1 Jul 01 22:19 No Changes	6s	156ms	187ms (paused for 4min 43s)	427ms	167ms	1s	3s

- **Example**:
  - Writing Groovy scripts to define each step of the build and deployment process.

Freestyle Projects are intuitive and visual, while **Pipeline Projects** provide greater control and automation through scripting. The choice between them depends on project complexity.

Building a Jenkins Pipeline Using a Script

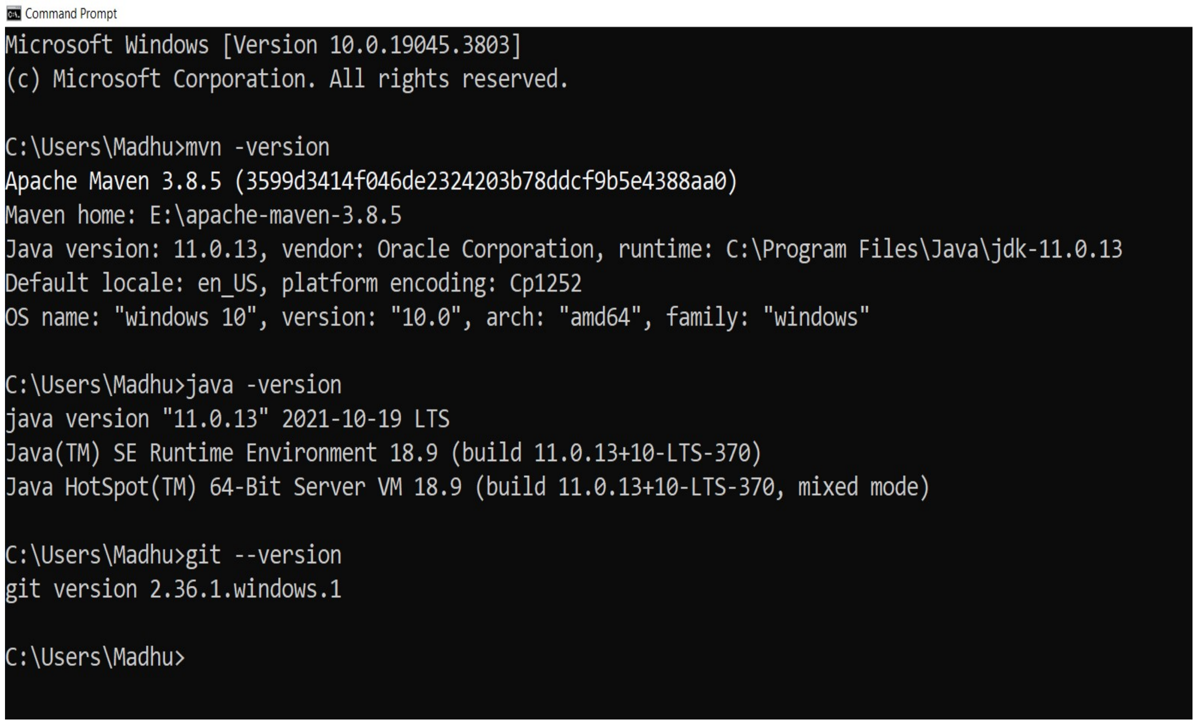
Detailed instructions on creating a Jenkins Pipeline with scripting, a crucial process in software development for automating Continuous Integration (CI) and Continuous Delivery (CD).

1. Preparing the Environment

Checking Tool Versions

Before starting with Jenkins, it's essential to verify that the required tools are installed and working:

- **Maven**: Use `mvn -version` to confirm Maven's installation.
- **Java**: Run `java -version` to check if the Java Development Kit (JDK) is correctly set up.
- **Git**: Verify Git's availability with `git --version`.



```
Microsoft Windows [Version 10.0.19045.3803]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Madhu>mvn -version
Apache Maven 3.8.5 (3599d3414f046de2324203b78ddcf9b5e4388aa0)
Maven home: E:\apache-maven-3.8.5
Java version: 11.0.13, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-11.0.13
Default locale: en_US, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"

C:\Users\Madhu>java -version
java version "11.0.13" 2021-10-19 LTS
Java(TM) SE Runtime Environment 18.9 (build 11.0.13+10-LTS-370)
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.13+10-LTS-370, mixed mode)

C:\Users\Madhu>git --version
git version 2.36.1.windows.1

C:\Users\Madhu>
```

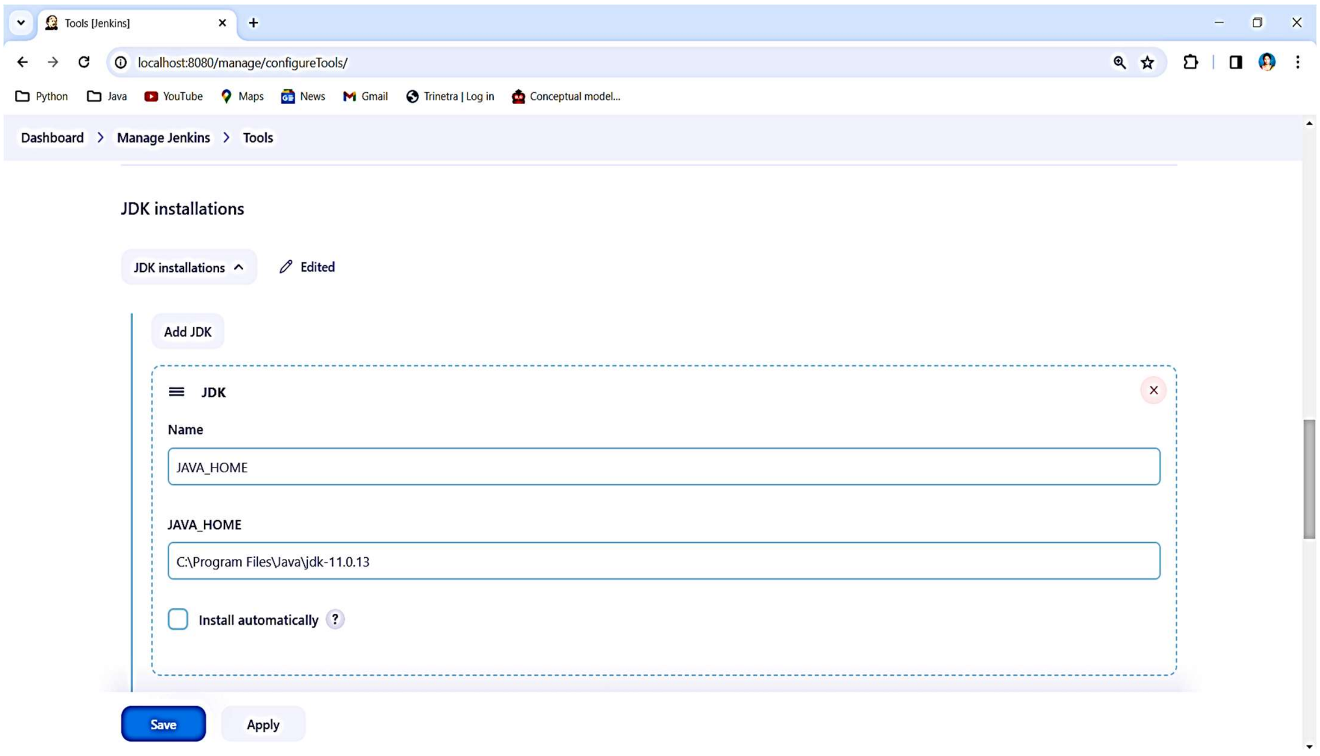
Confirming these tools are correctly installed and accessible helps ensure that Jenkins will function properly.

Configuring Jenkins with Required Tools

Once the tools are verified, configure Jenkins to recognize them:

1. Log into Jenkins and navigate to **Manage Jenkins → Global Tool Configuration**.

2. Specify paths for:
   - **Java**: Set the JDK path so Jenkins can compile Java projects.



JDK installations

JDK installations Edited

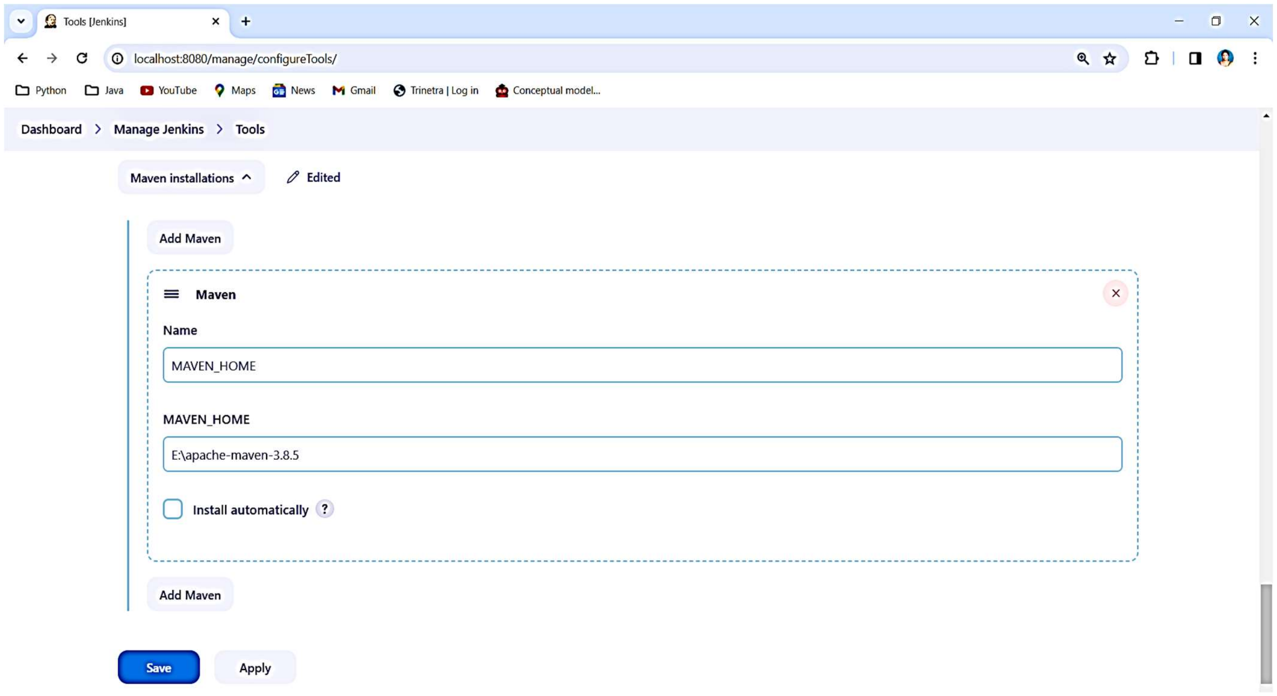
JDK

Name: JAVA_HOME

JAVA_HOME: C:\Program Files\Java\jdk-11.0.13

Install automatically ?

   - **Maven**: Define the Maven installation path for managing project builds.



Maven installations Edited

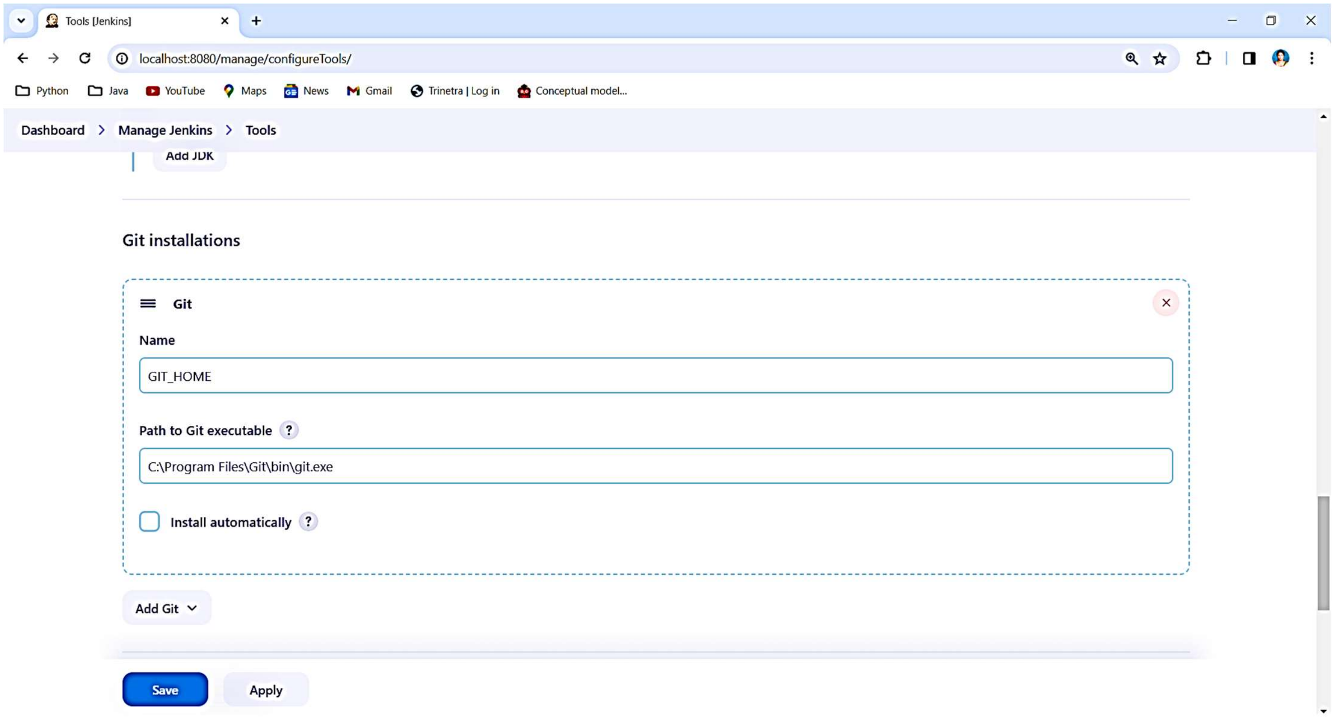
Maven

Name: MAVEN_HOME

MAVEN_HOME: E:\apache-maven-3.8.5

Install automatically ?

- **Git**: Configure the Git path for source control integration.



Git installations

Git

Name: GIT_HOME

Path to Git executable: C:\Program Files\Git\bin\git.exe

Install automatically: ☐

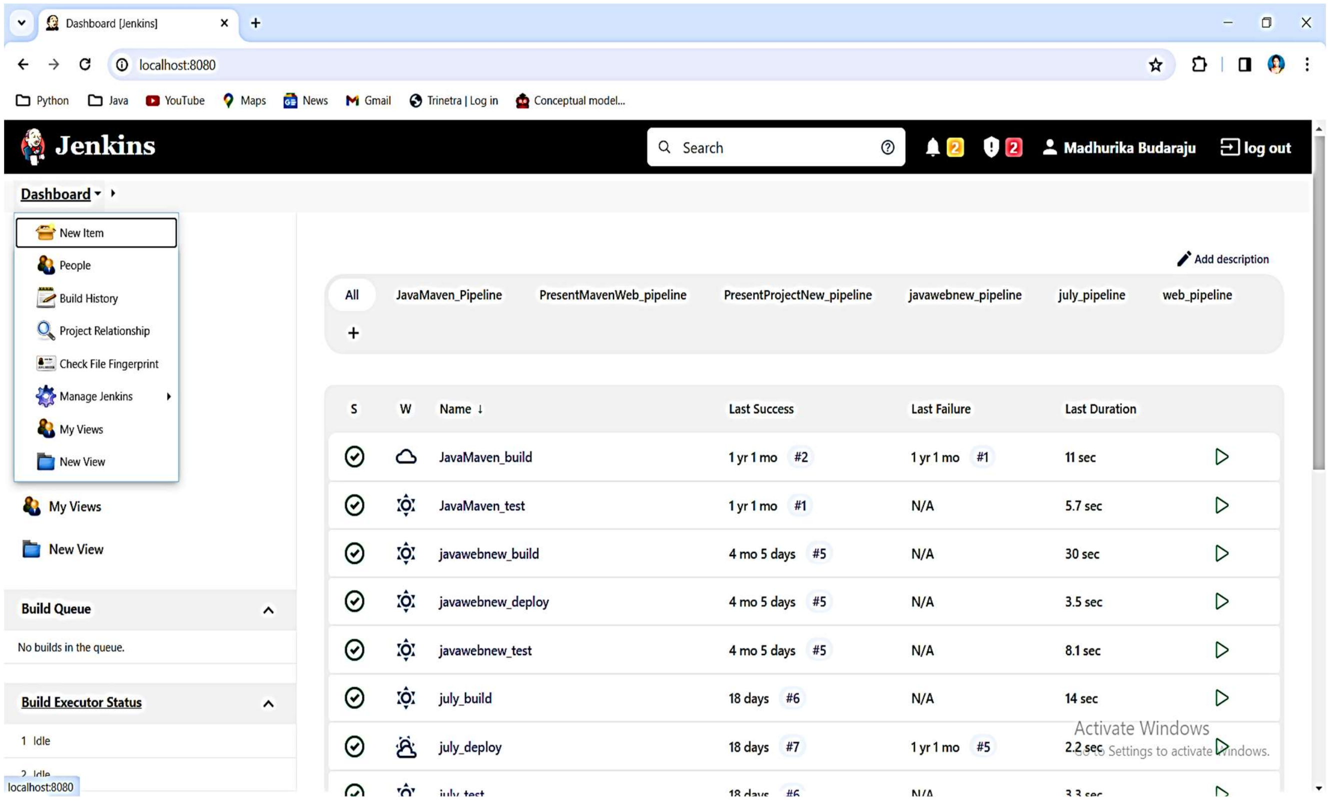
Correct configuration ensures that Jenkins can build, test, and deploy projects without errors.

2. Creating a New Pipeline in Jenkins

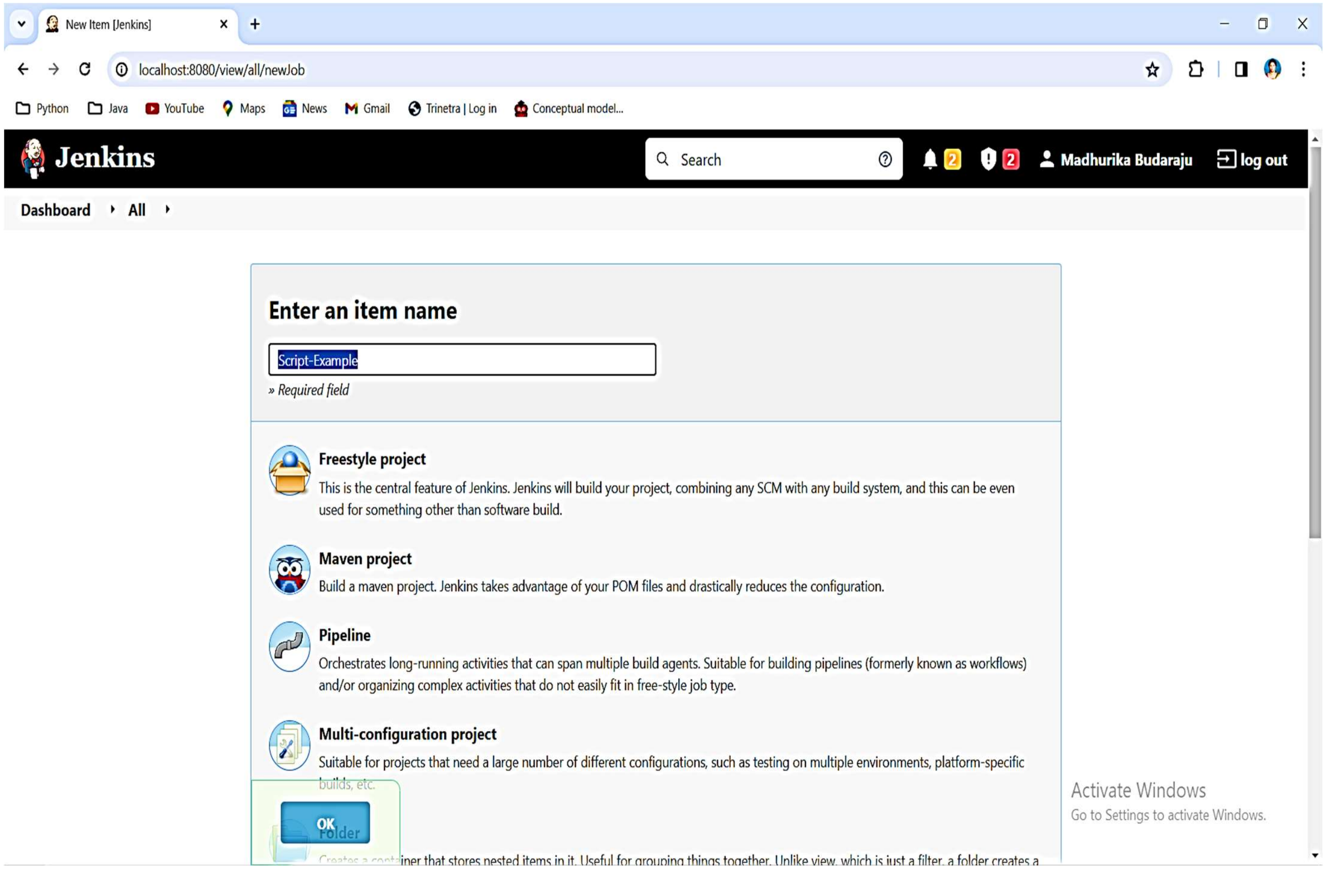
Setting Up a New Pipeline

With tools configured, start setting up a new Jenkins Pipeline:

1. Access the Jenkins dashboard.
2. Click on **New Item** in the main menu.
3. Enter a descriptive name for the pipeline.
4. Choose the **Pipeline** option and click **OK** to create the new pipeline structure.



S	W	Name ↓	Last Success	Last Failure	Last Duration
✓		JavaMaven_build	1 yr 1 mo #2	1 yr 1 mo #1	11 sec
✓		JavaMaven_test	1 yr 1 mo #1	N/A	5.7 sec
✓		javawebnew_build	4 mo 5 days #5	N/A	30 sec
✓		javawebnew_deploy	4 mo 5 days #5	N/A	3.5 sec
✓		javawebnew_test	4 mo 5 days #5	N/A	8.1 sec
✓		july_build	18 days #6	N/A	14 sec
✓		july_deploy	18 days #7	1 yr 1 mo #5	2.2 sec
✓		july_test	18 days #6	N/A	3.3 sec



Enter an item name

Script-Example

» Required field

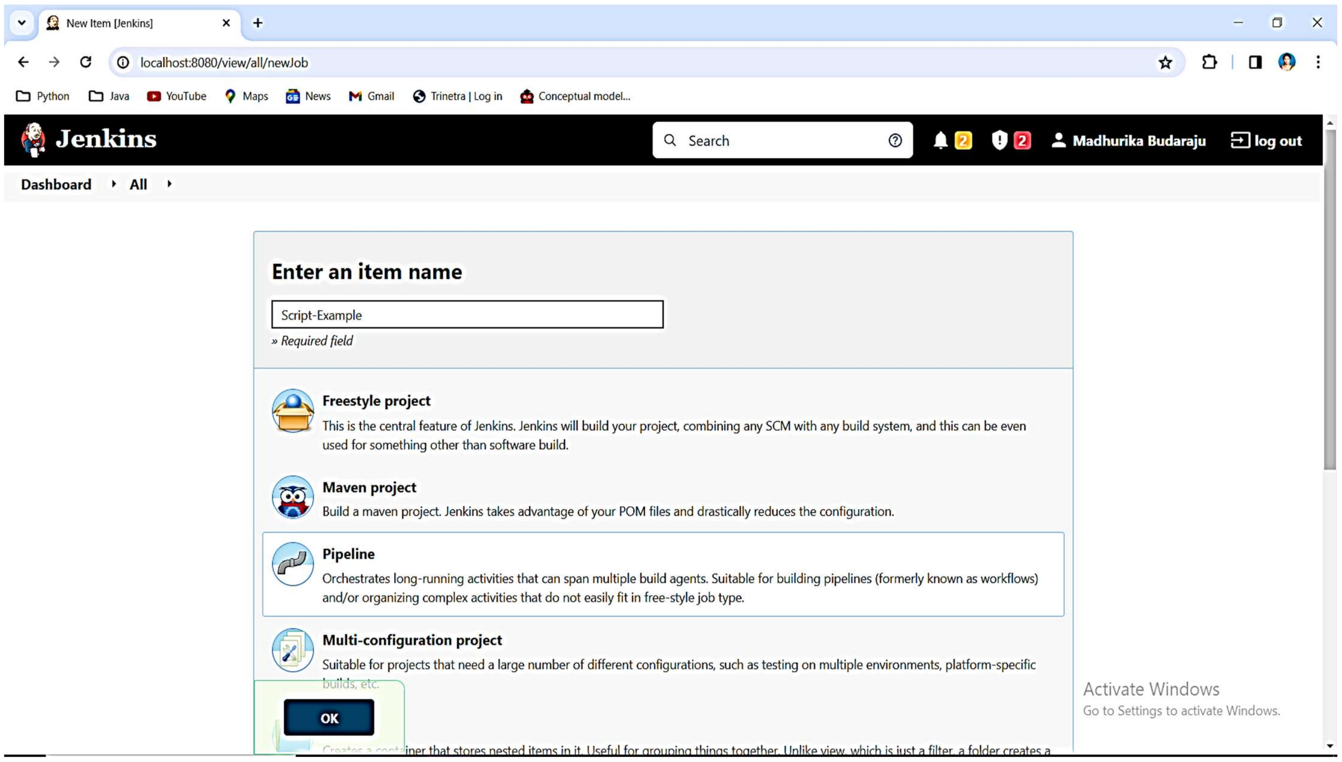
Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a



Enter an item name

Script-Example

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a

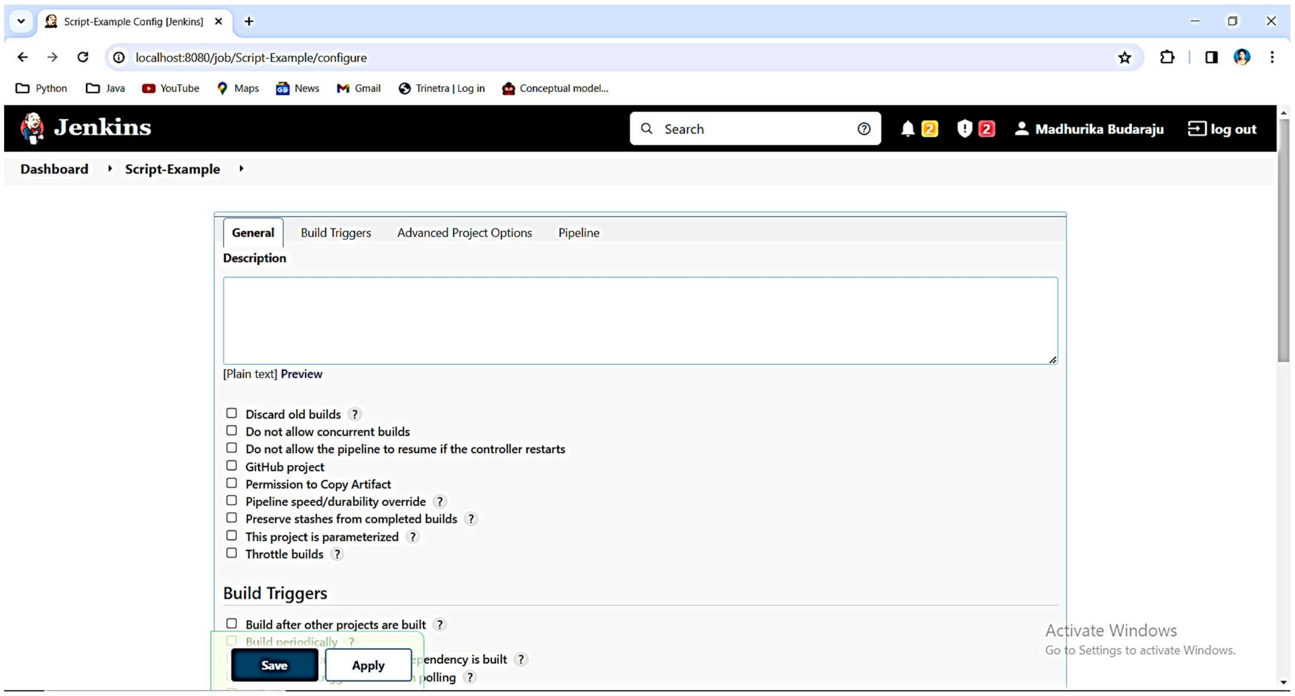
This setup forms the foundation of the pipeline, ready for script input.

3. Configuring the Pipeline Script

Accessing the Pipeline Script Section

After creating the pipeline, it's time to define the workflow:

1. Scroll to the **Pipeline** section in the project configuration screen.
2. Input or paste the script that will dictate the steps of the pipeline.
3. Utilize templates like the "Hello World" example to quickly start if needed. Templates offer a simple and accessible way to get familiar with Jenkins scripts.


Script-Example Config [Jenkins]
localhost:8080/job/Script-Example/configure
Python Java YouTube Maps News Gmail Trinetra | Log in Conceptual model...
Jenkins
Search
Madhurika Budaraju
log out
Dashboard › Script-Example ›
General Build Triggers Advanced Project Options Pipeline
Save
Apply
Activate Windows
Go to Settings to activate Windows.

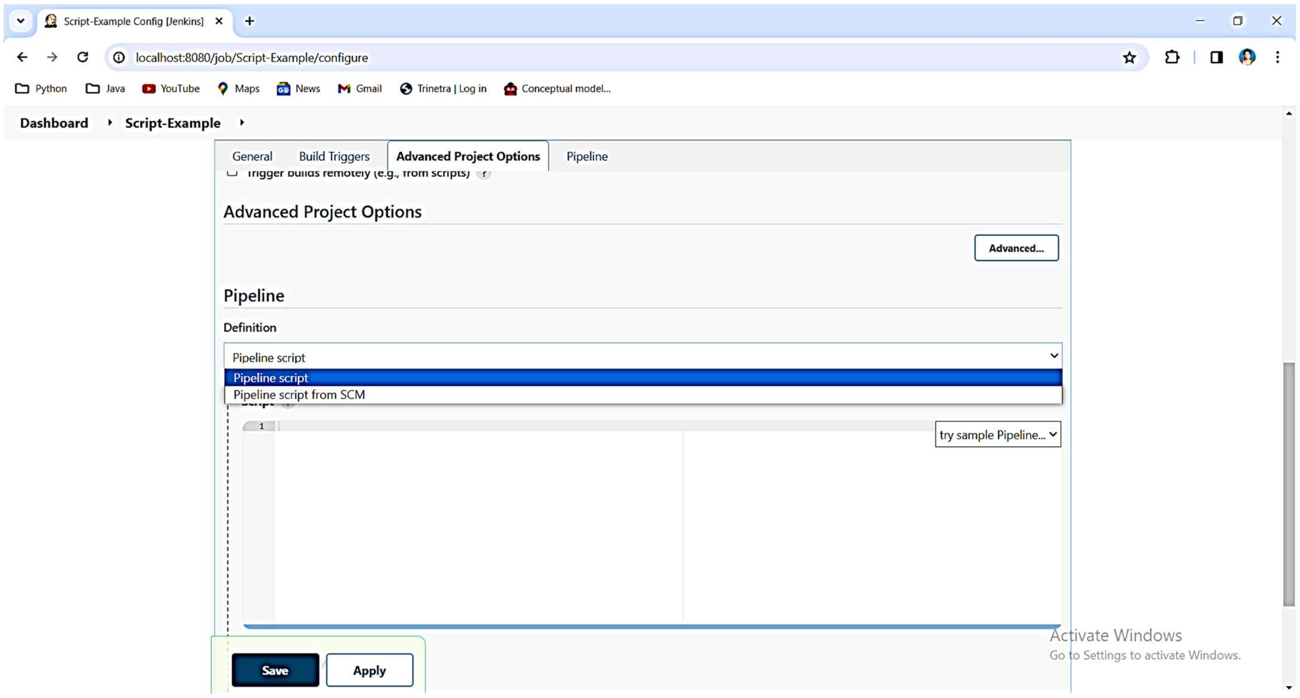
Description

[Plain text] Preview

- ☐ Discard old builds
- ☐ Do not allow concurrent builds
- ☐ Do not allow the pipeline to resume if the controller restarts
- ☐ GitHub project
- ☐ Permission to Copy Artifact
- ☐ Pipeline speed/durability override
- ☐ Preserve stashes from completed builds
- ☐ This project is parameterized
- ☐ Throttle builds

Build Triggers

- ☐ Build after other projects are built
- ☐ Build periodically
- ☐ …endency is built
- ☐ …polling


Script-Example Config [Jenkins]
localhost:8080/job/Script-Example/configure
Python Java YouTube Maps News Gmail Trinetra | Log in Conceptual model...
Dashboard › Script-Example ›
General Build Triggers Advanced Project Options Pipeline
Advanced...
try sample Pipeline...
Save
Apply
Activate Windows
Go to Settings to activate Windows.

☐ Trigger builds remotely (e.g., from scripts)

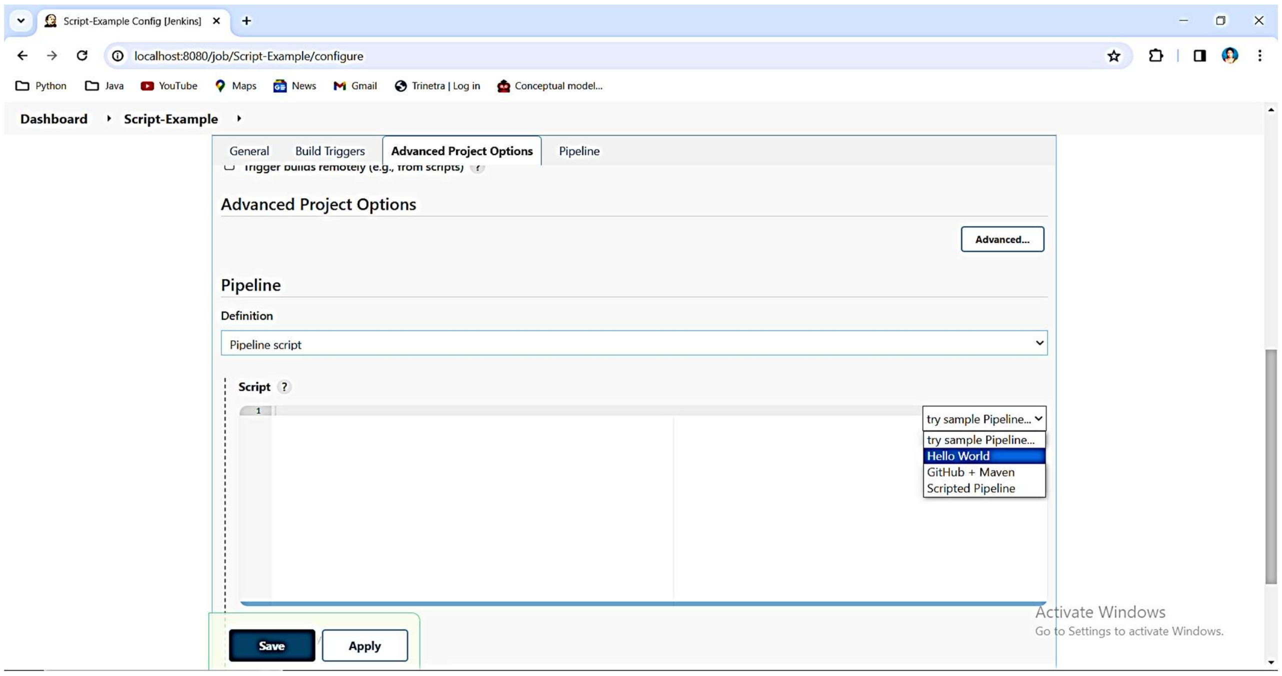
Advanced Project Options

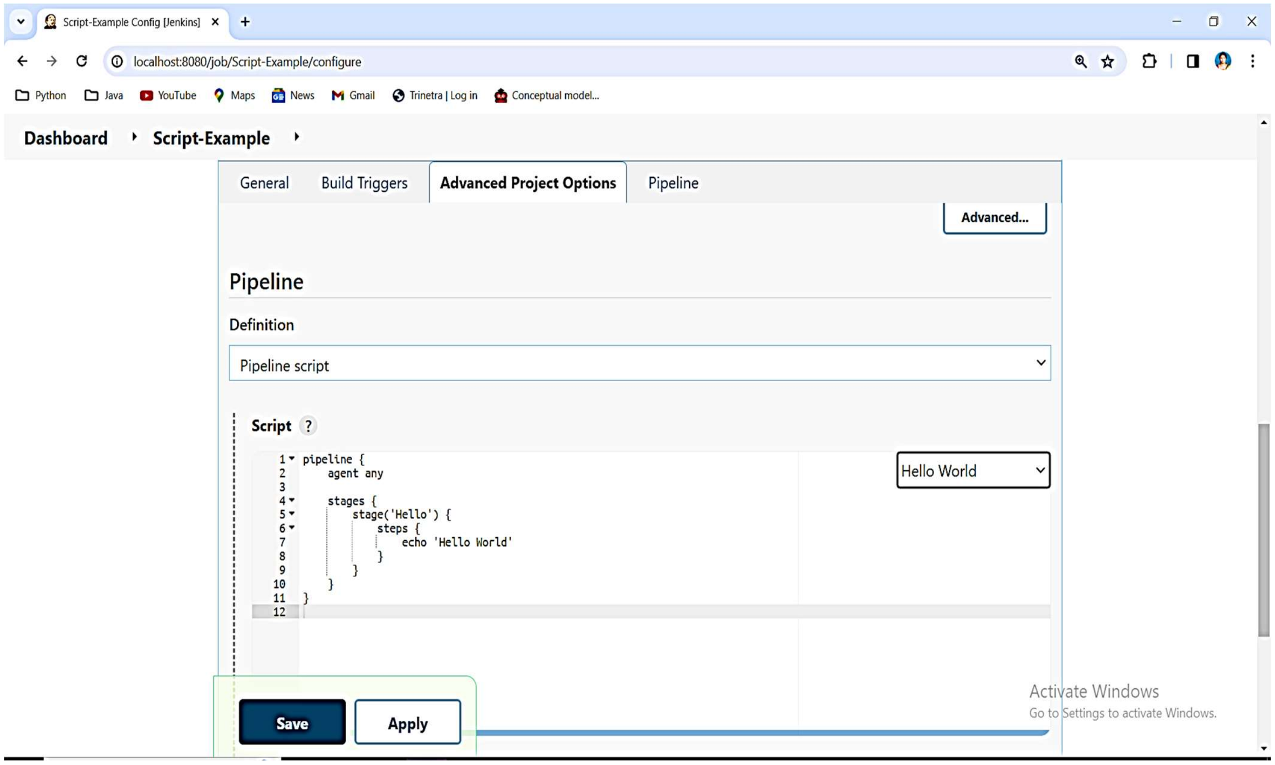
Pipeline

Definition

Pipeline script

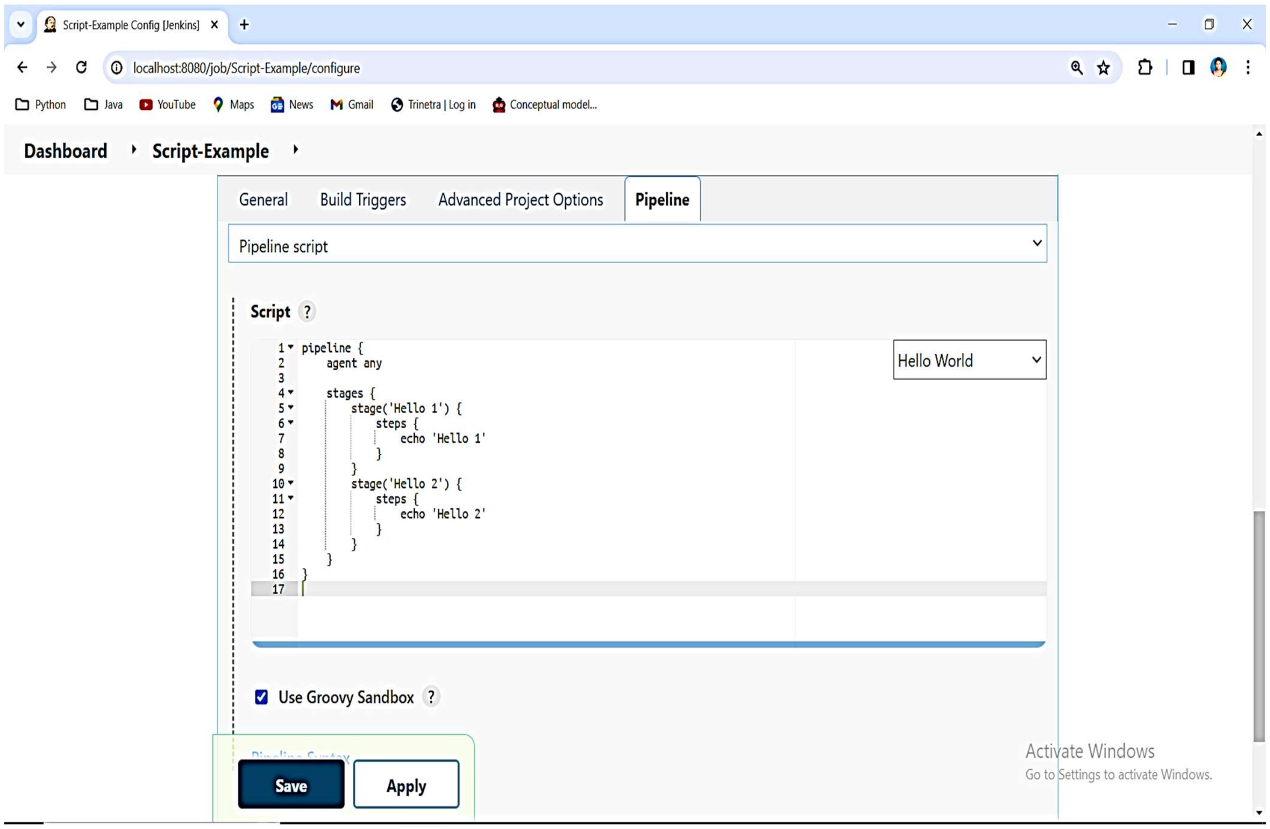
- Pipeline script
- Pipeline script from SCM





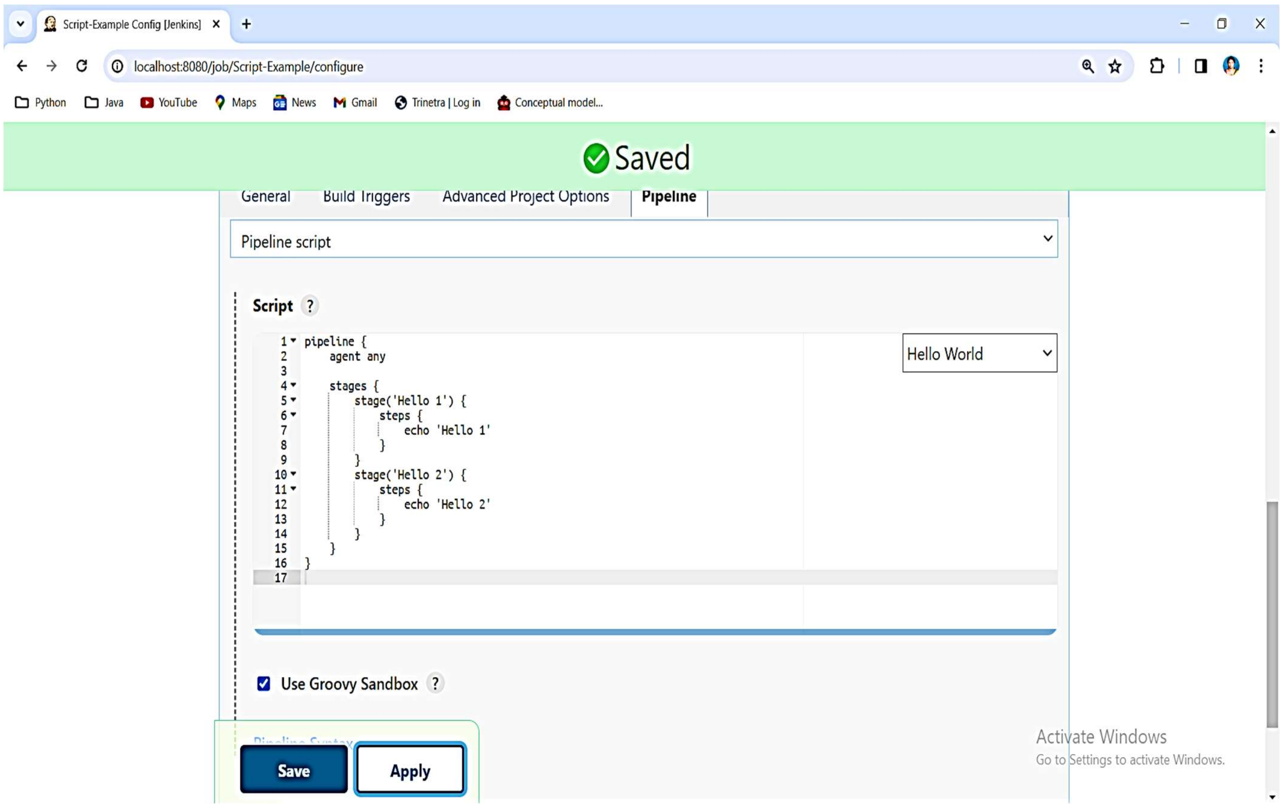
```
pipeline {
    agent any

    stages {
        stage('Hello') {
            steps {
                echo 'Hello World'
            }
        }
    }
}
```



```
pipeline {
    agent any

    stages {
        stage('Hello 1') {
            steps {
                echo 'Hello 1'
            }
        }
        stage('Hello 2') {
            steps {
                echo 'Hello 2'
            }
        }
    }
}
```



```
pipeline {
    agent any

    stages {
        stage('Hello 1') {
            steps {
                echo 'Hello 1'
            }
        }
        stage('Hello 2') {
            steps {
                echo 'Hello 2'
            }
        }
    }
}
```

4. Understanding the Pipeline Script Structure

Key Elements of the Sample Script

The basic pipeline script has a few key components:

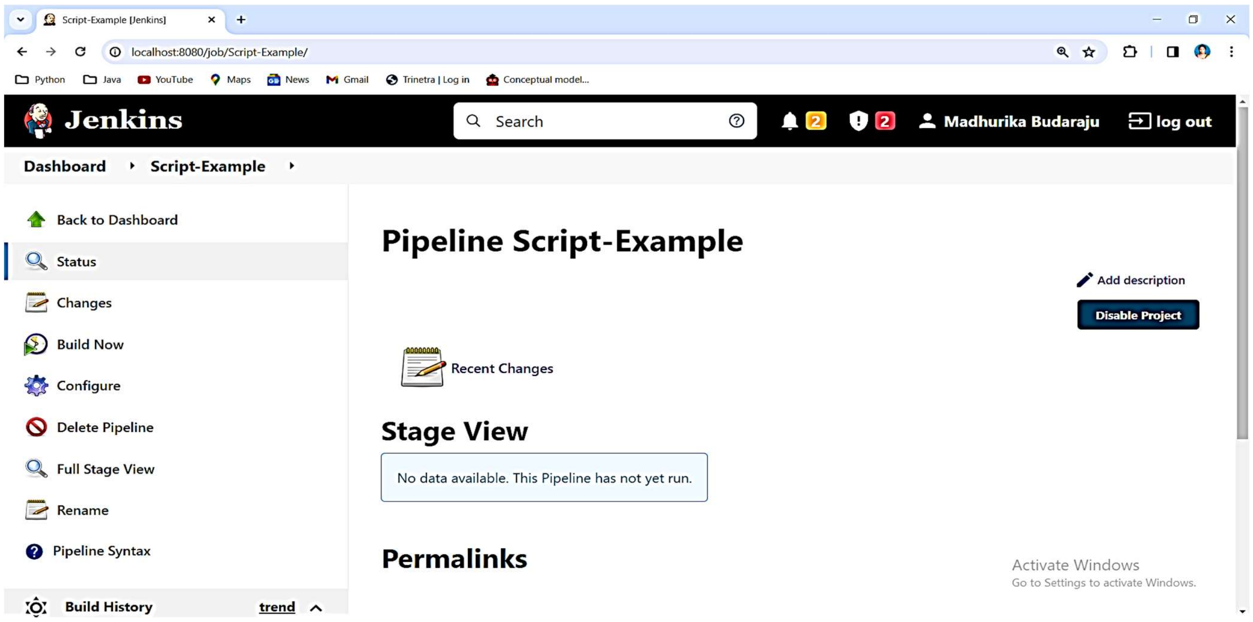
- **Agent**: Defines the machine (or slave) that runs the tasks. In a single-machine setup, the local machine acts as both the master and the slave. In more complex environments, specify the slave's ID in the script.
- **Stages**: The script is divided into stages, each representing a specific step, like building, testing, or deploying the code. Proper indentation is important to maintain clarity and script functionality.

5. Expanding the Pipeline with Additional Stages

Creating Multiple Stages for Workflow Flexibility

For detailed workflows, add multiple stages:

- Define additional stages like "Hello 1" and "Hello 2" to handle different tasks. Each stage can perform a unique function, aiding in the management of complex pipelines.
- Save changes by clicking **Apply** and **Save** to ensure the updated configuration is stored.



Pipeline Script-Example

Stage View

No data available. This Pipeline has not yet run.

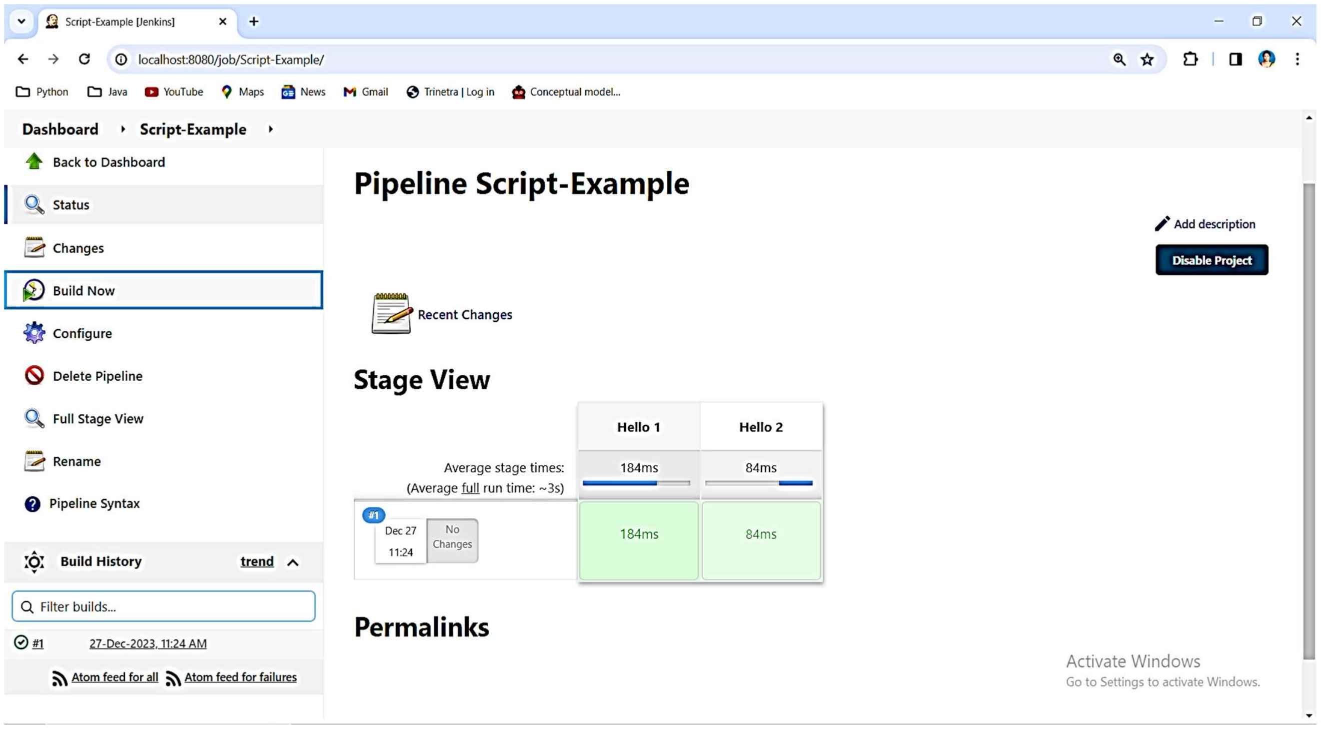
Permalinks

6. Running the Pipeline

Executing the Scripted Pipeline

Once the pipeline script is set:

- Return to the Jenkins dashboard.
- Click **Build Now** to trigger the execution. Jenkins will follow the sequence outlined in the script, processing each stage step-by-step.



	Hello 1	Hello 2
Average stage times: (Average full run time: ~3s)	184ms	84ms
#1 Dec 27 11:24 No Changes	184ms	84ms

7. Monitoring Pipeline Progress

Tracking Execution in Real-Time

Jenkins provides a visual display of the pipeline's execution:

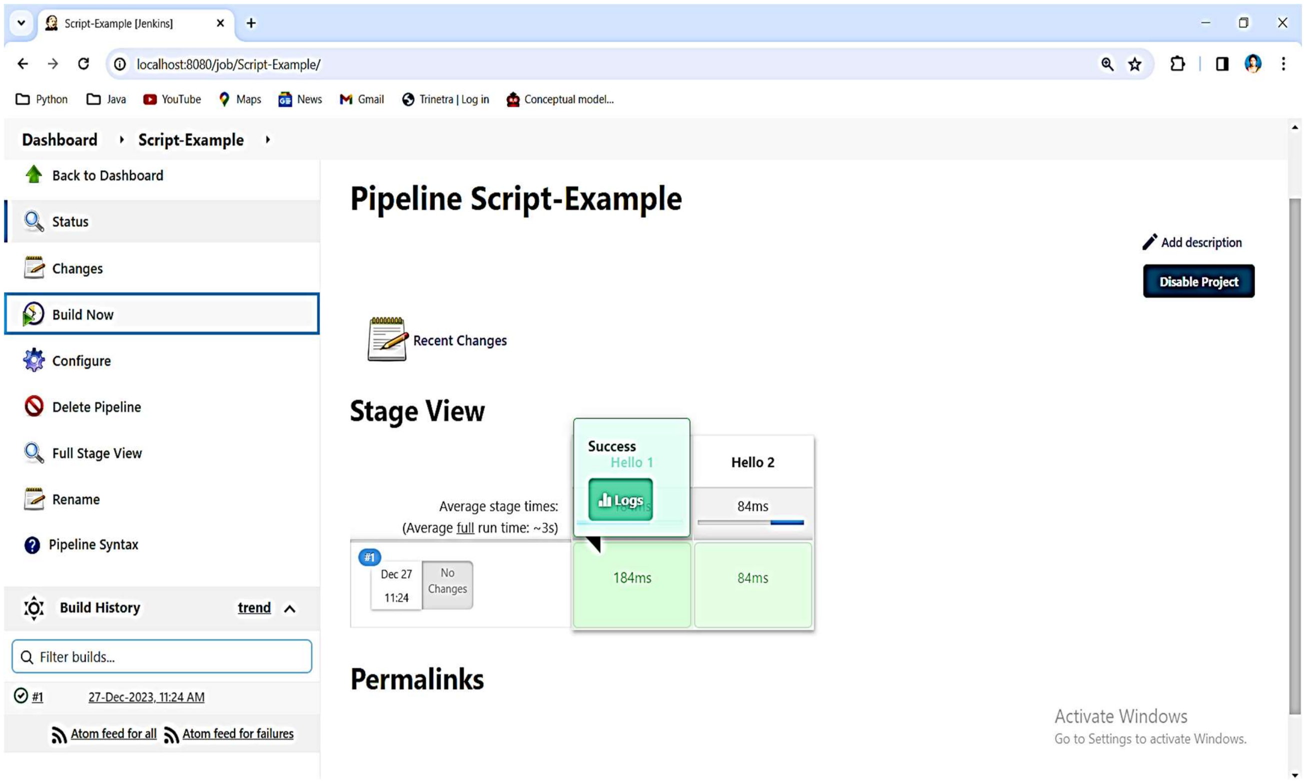
- Each stage's status is clearly shown, indicating whether it's in progress, completed, or has encountered an error.
- Hover over any stage to access more detailed information about its execution.

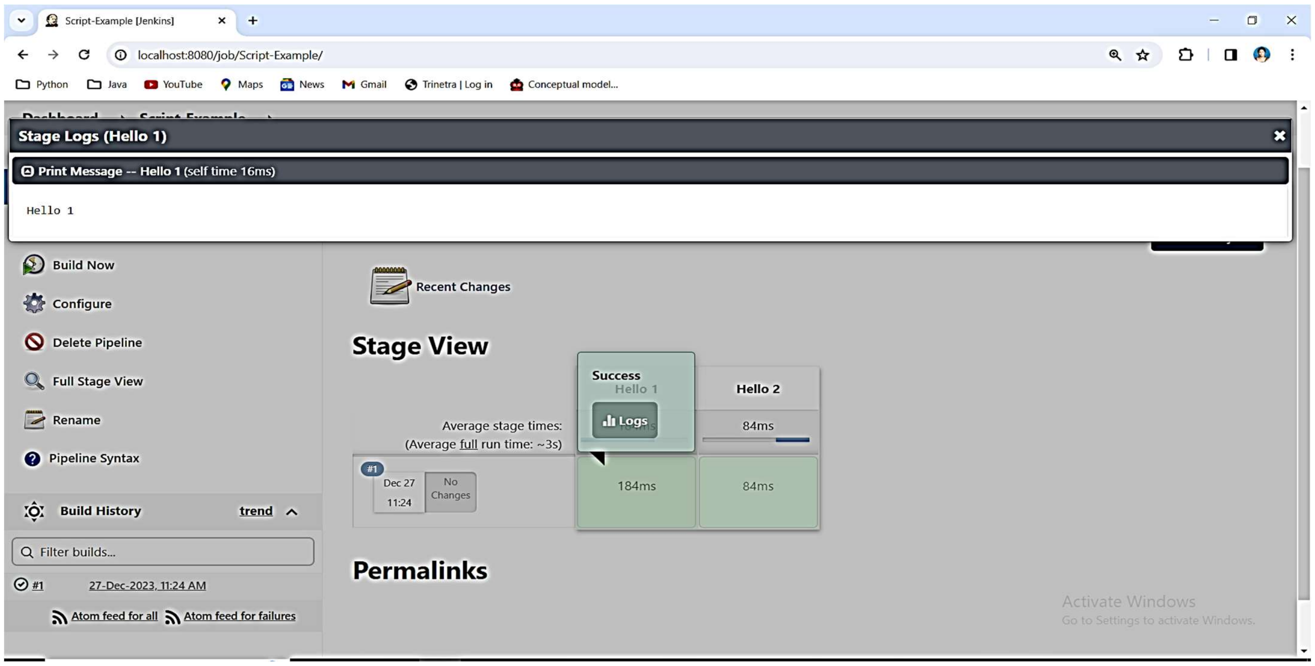
8. Reviewing Logs for Each Stage

Accessing Detailed Logs for Troubleshooting

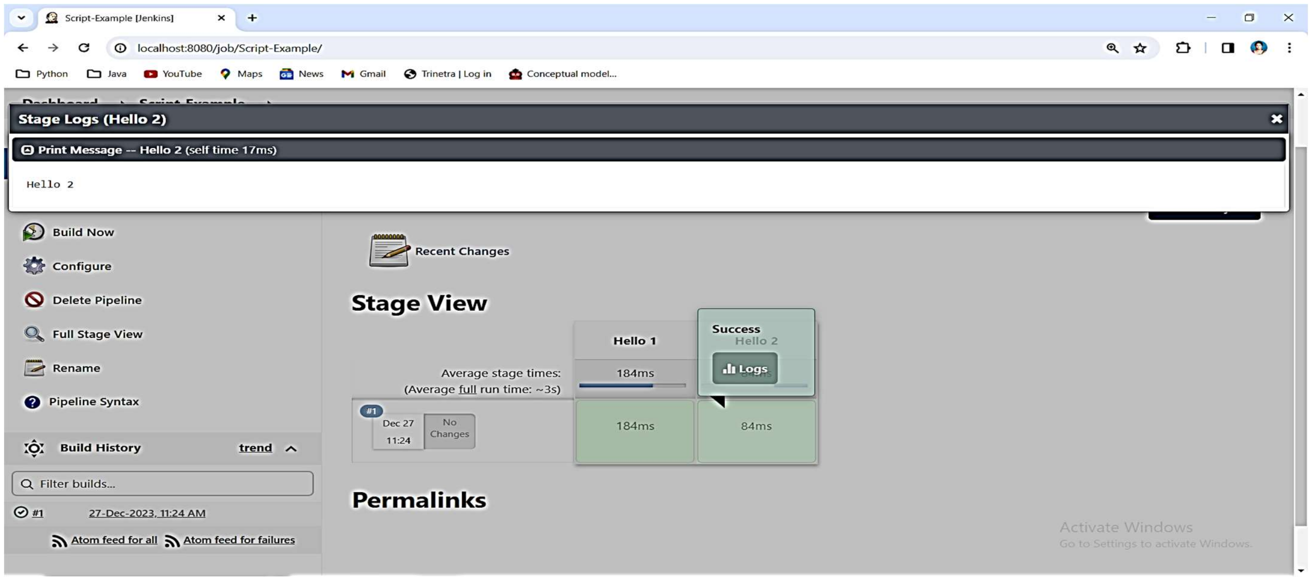
Every stage generates logs that provide a breakdown of actions taken:

- Early stages might show information on environment setup or code compilation.
- Later stages might include the results from testing or packaging processes.
- Analyzing these logs helps to ensure that each part of the pipeline functions correctly and assists in pinpointing any issues.





```
Hello 1
```



```
Hello 2
```

Note: If this topic comes as a Long Answer Question, instead of using the basic "Hello World" script, write a scripted pipeline for a Maven project. For a detailed example, refer to the video of LAB EXERCISE - 6-C: BUILDING THE CI/CD SCRIPTED PIPELINE USING JENKINS FOR A MAVEN JAVA PROJECT WITH POLL SCM. This example will help explain the Jenkins scripted pipeline in more detail.