

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

» Required field

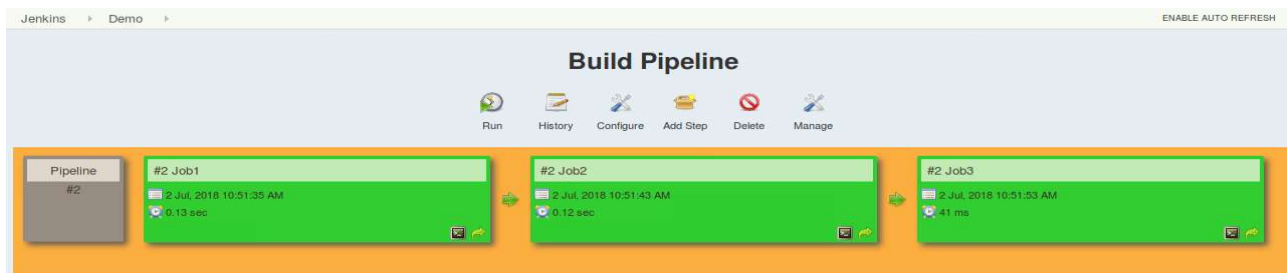
**Freestyle project**
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**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



- **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`.

```
Command Prompt
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.

The screenshot shows the Jenkins 'Tools' configuration page. The breadcrumb navigation is 'Dashboard > Manage Jenkins > Tools'. The page title is 'JDK installations'. There is a tab labeled 'JDK installations' and a link to 'Edit'. A dashed box contains the configuration for a new JDK installation. The 'Name' field is labeled 'JDK' and contains 'JAVA_HOME'. The 'JAVA_HOME' field contains 'C:\Program Files\Java\jdk-11.0.13'. There is an unchecked checkbox for 'Install automatically'. Below the dashed box are 'Save' and 'Apply' buttons.

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

The screenshot shows the Jenkins 'Tools' configuration page. The breadcrumb navigation is 'Dashboard > Manage Jenkins > Tools'. The page title is 'Maven installations'. There is a tab labeled 'Maven installations' and a link to 'Edit'. A dashed box contains the configuration for a new Maven installation. The 'Name' field is labeled 'Maven' and contains 'MAVEN_HOME'. The 'MAVEN_HOME' field contains 'E:\apache-maven-3.8.5'. There is an unchecked checkbox for 'Install automatically'. Below the dashed box are 'Save' and 'Apply' buttons.

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

The screenshot shows the Jenkins 'Tools' configuration page. The breadcrumb navigation is 'Dashboard > Manage Jenkins > Tools'. The 'Add JUK' button is visible. The 'Git installations' section contains a form for adding a new Git installation. The form has a title 'Git' with a close button. The 'Name' field is 'GIT_HOME'. The 'Path to Git executable' field is 'C:\Program Files\Git\bin\git.exe'. The 'Install automatically' checkbox is unchecked. Below the form is an 'Add Git' button. At the bottom of the page are 'Save' and 'Apply' buttons.

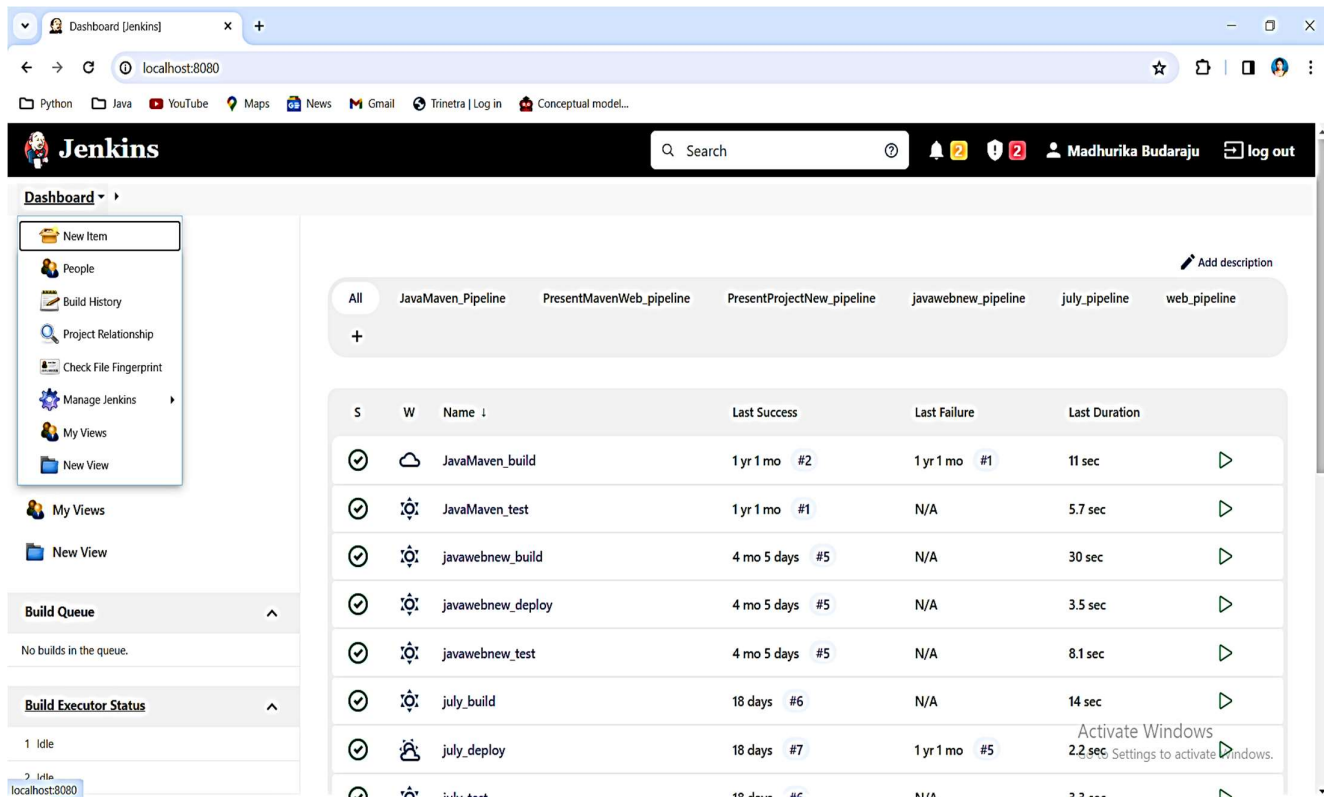
Correct configuration ensures that Jenkins can build, test, and deploy projects without errors.

2. Creating a New Pipeline in Jenkins

Setting Up a New Pipeline

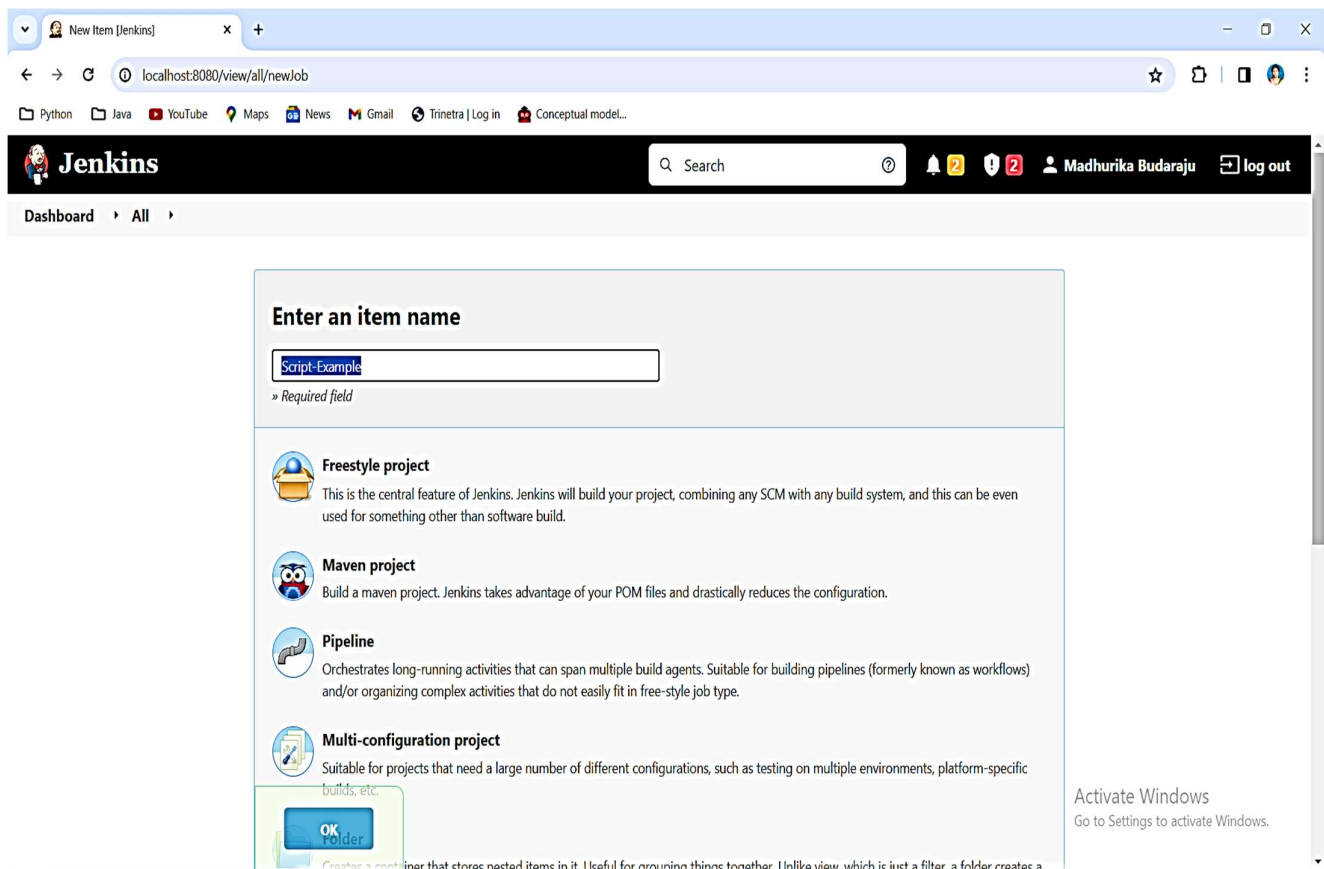
With tools configured, start setting up a new Jenkins Pipeline:

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.

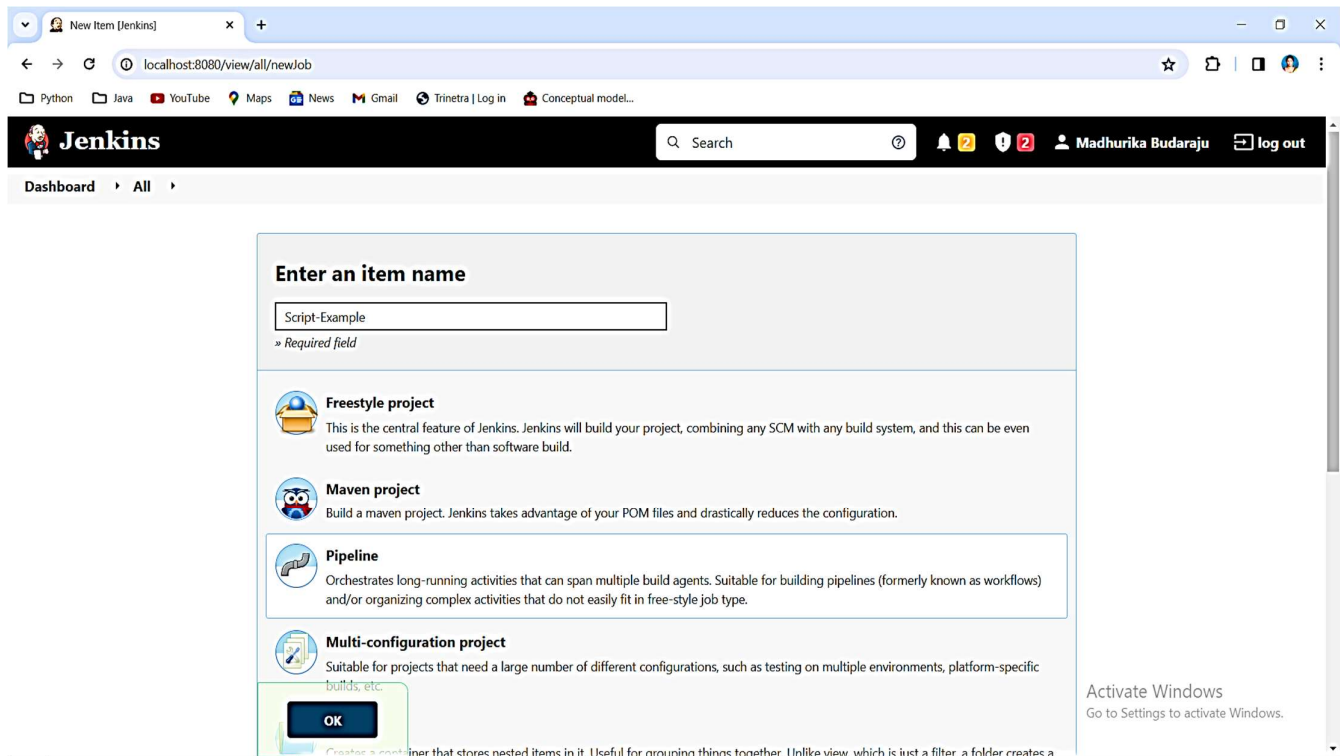


The screenshot shows the Jenkins Dashboard in a web browser. The browser address bar shows 'localhost:8080'. The Jenkins logo is in the top left, and a search bar is in the top right. The user 'Madhurika Budaraju' is logged in. The dashboard has a left sidebar with a 'New Item' button and a list of categories: People, Build History, Project Relationship, Check File Fingerprint, Manage Jenkins, My Views, and New View. The main area shows a table of builds with columns: S, W, Name, Last Success, Last Failure, and Last Duration. The table lists several builds, including 'JavaMaven_build', 'JavaMaven_test', 'javawebnew_build', 'javawebnew_deploy', 'javawebnew_test', 'july_build', and 'july_deploy'. The status of each build is indicated by a checkmark or a failure icon. The 'Last Success' column shows the time since the last successful build, and the 'Last Failure' column shows the time since the last failure. The 'Last Duration' column shows the time taken for the last build. A 'Build Queue' section on the left shows 'No builds in the queue.' and a 'Build Executor Status' section shows '1 idle' and '1 running'.

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	3.2 sec



The screenshot shows the Jenkins 'New Item' page. The browser address bar shows 'localhost:8080/view/all/newJob'. The Jenkins logo is in the top left, and a search bar is in the top right. The user 'Madhurika Budaraju' is logged in. The page has a 'Dashboard' breadcrumb and an 'All' filter. The main area is titled 'Enter an item name' and contains a text input field with 'Script-Example' entered. Below the input field is a 'Required field' label. The page lists four project types: Freestyle project, Maven project, Pipeline, and Multi-configuration project. Each project type has a description and a 'Create a new item' button. The 'Freestyle project' description states: '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.' The 'Maven project' description states: 'Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.' The 'Pipeline' description states: '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.' The 'Multi-configuration project' description states: 'Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.'



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.

The screenshot shows the Jenkins configuration page for a job named 'Script-Example'. The browser address bar indicates the URL is 'localhost:8080/job/Script-Example/configure'. The Jenkins header shows the user 'Madhurika Budaraju' is logged in. The 'General' tab is selected, displaying a 'Description' text area, a '[Plain text] Preview' section, and a list of checkboxes for build options: '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', and 'Throttle builds'. The 'Build Triggers' section includes a checkbox for 'Build after other projects are built' and a 'Build periodically' option with a text input field. 'Save' and 'Apply' buttons are at the bottom.

The screenshot shows the same Jenkins configuration page, but with the 'Pipeline' tab selected. The 'Advanced Project Options' tab is also visible. The 'Pipeline' section shows a 'Definition' dropdown menu with 'Pipeline script' selected. Below the dropdown is a 'try sample Pipeline...' button. The 'Save' and 'Apply' buttons are at the bottom.

The screenshot shows the Jenkins 'Script-Example Config' page with the 'Advanced Project Options' tab selected. The 'Pipeline' section has a 'Definition' dropdown set to 'Pipeline script'. The 'Script' area is empty, and a dropdown menu is open showing options: 'try sample Pipeline...', 'try sample Pipeline...', 'Hello World' (highlighted), 'GitHub + Maven', and 'Scripted Pipeline'. At the bottom are 'Save' and 'Apply' buttons. A watermark 'Activate Windows' is visible in the bottom right corner.

The screenshot shows the Jenkins 'Script-Example Config' page with the 'Pipeline' tab selected. The 'Definition' dropdown is set to 'Pipeline script'. The 'Script' area contains a YAML pipeline definition:

```
1 pipeline {  
2   agent any  
3  
4   stages {  
5     stage('Hello') {  
6       steps {  
7         echo 'Hello World'  
8       }  
9     }  
10  }  
11 }  
12
```

 A dropdown menu is open next to the script, showing 'Hello World' as the selected option. At the bottom are 'Save' and 'Apply' buttons. A watermark 'Activate Windows' is visible in the bottom right corner.

Script-Example Config [Jenkins] x +

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**

Pipeline script

Script ?

```
1 pipeline {
2   agent any
3
4   stages {
5     stage('Hello 1') {
6       steps {
7         echo 'Hello 1'
8       }
9     }
10    stage('Hello 2') {
11      steps {
12        echo 'Hello 2'
13      }
14    }
15  }
16 }
17
```

Use Groovy Sandbox ?

Save Apply

Activate Windows
Go to Settings to activate Windows.

Script-Example Config [Jenkins] x +

localhost:8080/job/Script-Example/configure

Python Java YouTube Maps News Gmail Trinetra | Log in Conceptual model...

✓ Saved

General Build Triggers Advanced Project Options **Pipeline**

Pipeline script

Script ?

```
1 pipeline {
2   agent any
3
4   stages {
5     stage('Hello 1') {
6       steps {
7         echo 'Hello 1'
8       }
9     }
10    stage('Hello 2') {
11      steps {
12        echo 'Hello 2'
13      }
14    }
15  }
16 }
17
```

Use Groovy Sandbox ?

Save Apply

Activate Windows
Go to Settings to activate Windows.

4. Understanding the Pipeline Script Structure

Key Elements of the Sample Script

The basic pipeline script has a few key components:

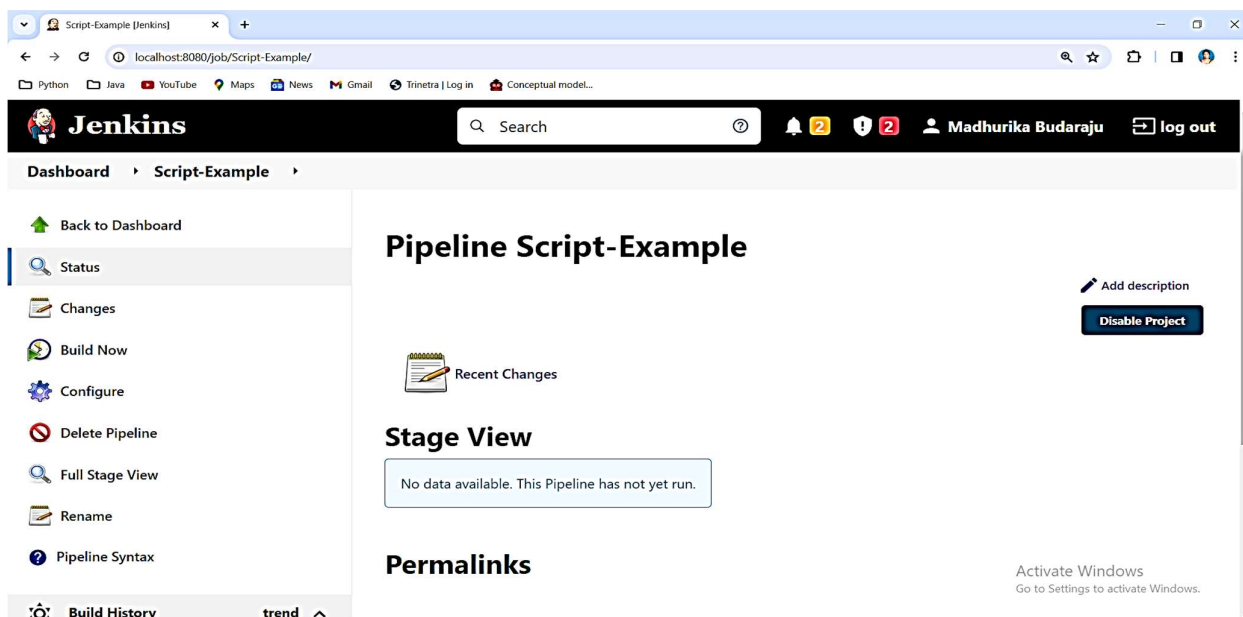
- **Agent:** Defines the machine (or slave) that runs the tasks. In a single-machine setup, the local machine acts as both the master and the slave. In more complex environments, specify the slave's ID in the script.
- **Stages:** The script is divided into stages, each representing a specific step, 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.



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.

The screenshot shows the Jenkins web interface for a pipeline named 'Script-Example'. The left sidebar contains navigation links: Dashboard, Script-Example, Back to Dashboard, Status, Changes, Build Now (highlighted), Configure, Delete Pipeline, Full Stage View, Rename, Pipeline Syntax, Build History, and a search bar. The main content area is titled 'Pipeline Script-Example' and includes a 'Recent Changes' section, a 'Stage View' table, and a 'Permalinks' section. The 'Stage View' table shows two stages: 'Hello 1' and 'Hello 2', both with a duration of 184ms. The 'Permalinks' section shows a build history entry for '27-Dec-2023_11:24 AM' with a status of 'No Changes'.

Stage View	
Stage	Duration
Hello 1	184ms
Hello 2	84ms

Build	Status	Duration
#1	No Changes	27-Dec-2023_11:24 AM

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.

The screenshot displays the Jenkins web interface for a pipeline named 'Script-Example'. The left sidebar contains navigation options: Dashboard, Script-Example, Back to Dashboard, Status, Changes, Build Now (highlighted), Configure, Delete Pipeline, Full Stage View, Rename, Pipeline Syntax, Build History, and a search bar for builds. The main content area shows the 'Pipeline Script-Example' with a 'Recent Changes' section and a 'Stage View' section. The 'Stage View' section displays a grid of stage results: 'Success Hello 1' (184ms) and 'Hello 2' (84ms). Below this, a table shows the build history for the pipeline, including the build number, date, and time. The 'Permalinks' section is also visible at the bottom.

Build	Date	Time	Status
#1	Dec 27	11:24	No Changes

The screenshot shows the Jenkins web interface for a job named 'Script-Example'. The 'Stage Logs (Hello 1)' window is open, displaying a log entry: 'Print Message -- Hello 1 (self time 16ms)' followed by the text 'Hello 1'. The main dashboard shows the 'Stage View' with a progress bar for 'Hello 1' at 184ms. The 'Build History' section shows a single build #1 from Dec 27, 11:24 AM, with a status of 'No Changes'. The 'Recent Changes' section is empty. The 'Permalinks' section is also empty. The 'Build Now' button is visible in the top left corner.

The screenshot shows the Jenkins web interface for the same job 'Script-Example'. The 'Stage Logs (Hello 2)' window is open, displaying a log entry: 'Print Message -- Hello 2 (self time 17ms)' followed by the text 'Hello 2'. The main dashboard shows the 'Stage View' with a progress bar for 'Hello 2' at 84ms. The 'Build History' section shows a single build #1 from Dec 27, 11:24 AM, with a status of 'No Changes'. The 'Recent Changes' section is empty. The 'Permalinks' section is also empty. The 'Build Now' button is visible in the top left corner.

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.