CI/CD Pipeline:
Continuous Integration and Contionuous Deployment

in devops,there are two teams like development team and operations team.
Four steps:
Build
Deploy
Testing

above three are Integration steps..CI integration steps
Release

and All those 4 teams are develop team

How to setup environment of Jenkins:

Requirements:
1.Download javersion of any 11,17 or 21
2.Edit the environment variables for java latest download of above
3.Download jenkins latest version
4.Give the source folder as C:\program files\jenkins
5.Give the option as local system for free usage
6.Test ur port number as valid
7.Give the Copied environment variable as the source folder for jenkins
8.click the next upto install jenkins

Open browser and give "localhost:8081"
And the copy the link as seen above and paste it inthe filemanager search bar
using notepad activate it.
so give the password in the localhost website.
click the continue button
and u appeared two suggested features as
1.install suggested plugins
2.selct the plugins u wish to install

select the first option since we are beginers for this and we can see some plugins are installing.....

In CICD pipeline,we have two types
1.build pipeline-individual monitoring of current and done projects
2.delivery pipeline-pictorial presentation of pipeline

Give the appropriate details and continue it.

And u appeared main page of Jenkins
Click new and give the name of project and choose as freestyle project and click ok
give the buildstep as window batch command and give echo command to save for a reference

click apply and save

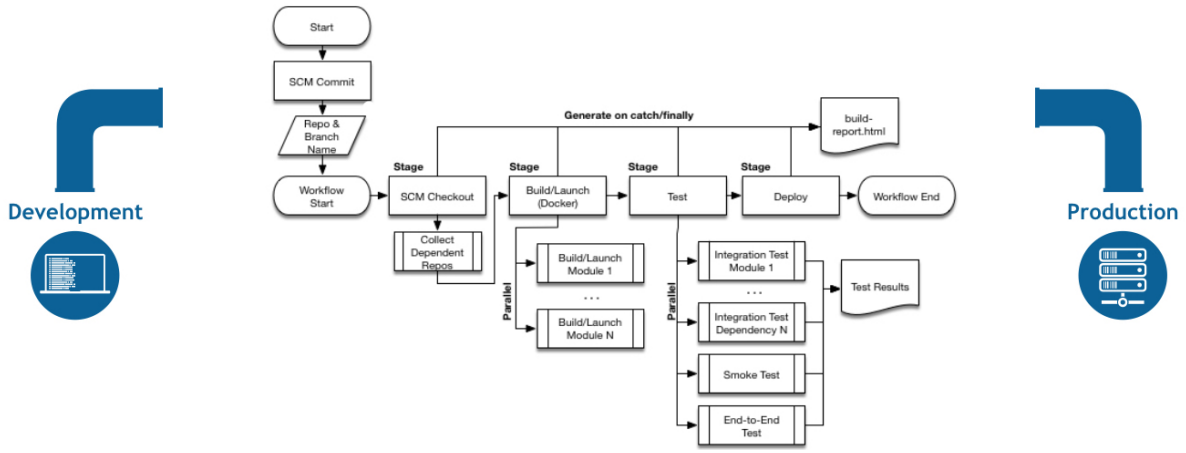
And u appeared build now option...then it is tested and gives the data at console output build steps are used to save theproject as individually and use the build now to test the errors in the project

build actions are used to link the one project to another project so that we can run the projects in one step that which collabs for same output

we can see two terms as upstream projects and downstreamprojects.

how to install plugins:
manage jenkins->select plugins->click on available plugins->search for build and delivery pipeline->select those and install it

Creating build pipeline:
-In dashboard,click +



Pipeline concepts

The following concepts are key aspects of Jenkins Pipeline, which tie in closely to Pipeline syntax (see the overview below).

Pipeline

A Pipeline is a user-defined model of a CD pipeline. A Pipeline's code defines your entire build process, which typically includes stages for building an application, testing it and then delivering it.

Also, a `pipeline` block is a key part of Declarative Pipeline syntax.

Node

A node is a machine which is part of the Jenkins environment and is capable of executing a Pipeline.

Also, a `node` block is a key part of Scripted Pipeline syntax.

Stage

A `stage` block defines a conceptually distinct subset of tasks performed through the entire Pipeline (e.g. "Build", "Test" and "Deploy" stages), which is used by many plugins to visualize or present Jenkins Pipeline status/progress. [6]

Step

A single task. Fundamentally, a step tells Jenkins *what* to do at a particular point in time (or "step" in the process). For example, to execute the shell command `make` use the `sh` step: `sh 'make'`. When a plugin extends the Pipeline DSL, [1] that typically means the plugin has implemented a new *step*.

Pipeline syntax overview

The following Pipeline code skeletons illustrate the fundamental differences between Declarative Pipeline syntax and Scripted Pipeline syntax.

Be aware that both stages and steps (above) are common elements of both Declarative and Scripted Pipeline syntax.

Declarative Pipeline fundamentals

In Declarative Pipeline syntax, the `pipeline` block defines all the work done throughout your entire Pipeline.

Two types of pipelines are there….
Declarative pipeline
Scripted pipeline

Creating a maven project by clicking new button at corner
_Give group id and artifact id and click finish then
_give right click on src/main/java and create a package name as com.lbrce.finalmodule
_give a right click on package and create class name as UserVerification
_write the code as same as below

Code:

```
package com.lbrce.finalmodule;
import java.util.ResourceBundle;
public class UserVerification {
		public boolean check(String uname,String pswd) {
			ResourceBundle rb=ResourceBundle.getBundle("config");
			String username=rb.getString("username");
			String password=rb.getString("password");
			if(uname.equals(username)&&pswd.equals(password))
				return true;
			else
				return false;
		}
}
```

_and save this file or code

_create a new file at right corner option and named as “configure.properties”

Code:

```
username=usha
password=usha@2020
```

_ save this file

_ create a new test file by create its package from src/test/java
_and also create a class by package and write the code as below

Code:

```
package com.lbrce.finalmodule;
import org.testng.Assert;
import org.testng.annotations.Test;
public class UserVerificationTest {
	UserVerification uv=new UserVerification();
	@Test
	public void testcase1() {
		Assert.assertEquals(true,uv.check("usha","usha@2020"));
	}
	@Test
	public void testcase2() {
		Assert.assertEquals(true,uv.check("usha","usha@2020"));
	}
	@Test
	public void testcase3() {
		Assert.assertEquals(false,uv.check("usha1","usha@2022"));
	}
	@Test
	public void testcase4() {
		Assert.assertEquals(false,uv.check("ushabajjuri","usha@2022"));
	}
}
```

_save the file

_open the pom.xml file and give the dependencies code as below
Code:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>lbrce</groupId>
<artifactId>finalmodule</artifactId>
<version>0.0.1-SNAPSHOT</version>
<dependencies>
        <!-- https://mvnrepository.com/artifact/org.testng/testng -->
<dependency>
<groupId>org.testng</groupId>
<artifactId>testng</artifactId>
<version>7.8.0</version>
<scope>test</scope>
</dependency>
</dependencies>
</project>
```

Then the errors are recovered through above file

_right click on the main module i.e., and click on the properties then u can see icon “->”
_then u can see the final module file in file explorer
_drag those files like src,target and pom.xml in a new repository named “finaldevopsmodule”
_click on “addfile” then give the name as “Jenkinsfile”
_then write the code as below

Code:_

```
pipeline{
  agent any
  stages{
    stage('Deploy'){
      steps{
        echo "Deploy successful"
        bat "mvn compile"
      }
    }
    stage('Build'){
      steps{
        echo "Build successful"
        bat "mvn clean"
      }
    }
    stage('Test'){
      steps{
        echo "Test successful"
        bat "mvn test"
      }
    }
  }
}
```

_run the main project i.e., final module by right click on the module
_then options are appeared by maven build, maven clean, maven test

_install the git for windows and edit the environment variables for suitable git as
C;/program files/Git/
_at the repository view of “finaldevopsmodule” click on “code” then u can see the url and copy it

Open the jenkins
_click new item and create a pipeline
_choose the option pipeline as “pipeline script from scm”
_choose option as none to Git
_paste the copied url from github
_apply and save it
_ then u get a pipeline created
_run the pipeline by using “build now” option