EXPERIMENT NO. 5

AIM :-
To Build the pipeline of jobs using Maven / Gradle / Ant in Jenkins, create a pipeline script to Test and deploy an application over the tomcat server.

THEORY :-

Jenkins Pipeline
In Jenkins, a pipeline is a collection of events or jobs which are interlinked with one another in a sequence.

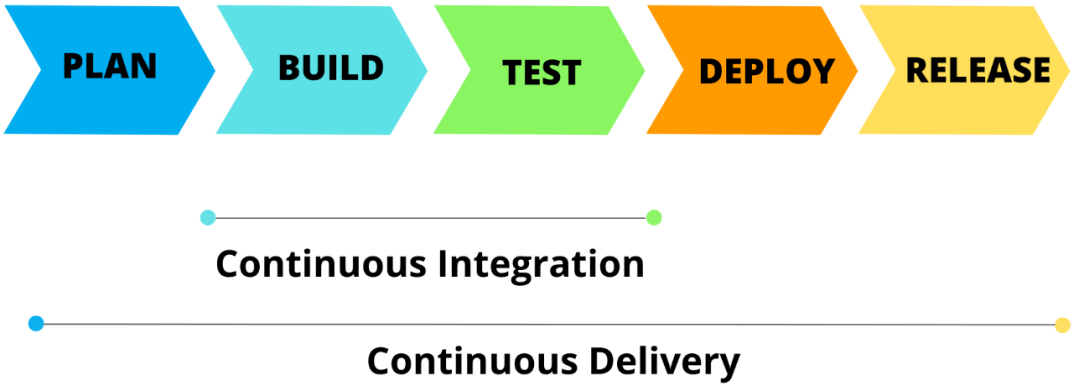
It is a combination of plugins that support the integration and implementation of continuous delivery pipelines using Jenkins.

In other words, a Jenkins Pipeline is a collection of jobs or events that brings the software from version control into the hands of the end users by using automation tools.

It is used to incorporate continuous delivery in our software development workflow. A pipeline has an extensible automation server for creating simple or even complex delivery pipelines "as code", via DSL (Domain-specific language).

Continuous Delivery Pipeline:

In a Jenkins Pipeline, every job has some sort of dependency on at least one or more jobs or events.



The above diagram represents a continuous delivery pipeline in Jenkins. It contains a collection of states such as build, deploy, test and release. These jobs or events are interlinked with each other. Every state has its jobs, which work in a sequence called a continuous delivery pipeline.

A continuous delivery pipeline is an automated expression to show your process for getting software for version control. Thus, every change made in your software goes through a number of complex processes on its manner to being released. It also involves developing the software in a repeatable and reliable manner, and progression of the built software through multiple stages of testing and deployment.

Jenkins – Build Jobs

a) Go to **New Items** → Give a project name in "Item name" field → select **Maven project** → click **OK**

b) Now configure the job

- Provide the description
- In **Source Code Management**, there are options for CVS project, Git etc, select the one which is required
- In **Build Triggers**, there are multiple options like "Build when a change is pushed to GitHub", "Poll SCM", "Build whenever a SNAPSHOT dependency is built" etc, select the required one
- Give the path of your **pom.xml** file in **Build Root POM**
- Give "Goals and options", take a use case where the requirement is to install the code then give **clean install**

c) Configure the job as in the screenshot and don't forget to **Save**

d) In the same way, create another job (**4503_deploy**)

e) Suppose the requirement is creating two jobs, one (**4502_deploy**) for deploying the code on author instance 4502 and if its build is successful then it should run its downstream job (**4503_deploy**) for deploying code on publishing instance.

f) Now configure the job **4502_deploy**, in **Post Steps** select **Run only if build succeeds**

g) In **Post Build Action**, add post build action and give the job name (**4503_deploy**) in "Projects to build"

Step 6 – Build the jobs
Now build the job **4502_deploy**, on successful completion **4503_deploy** will trigger automatically and hopefully, jobs will be successful !!!

Programs :

Script 1 : Jenkins Declarative Pipeline script that runs on any agent and prints "Hello World".

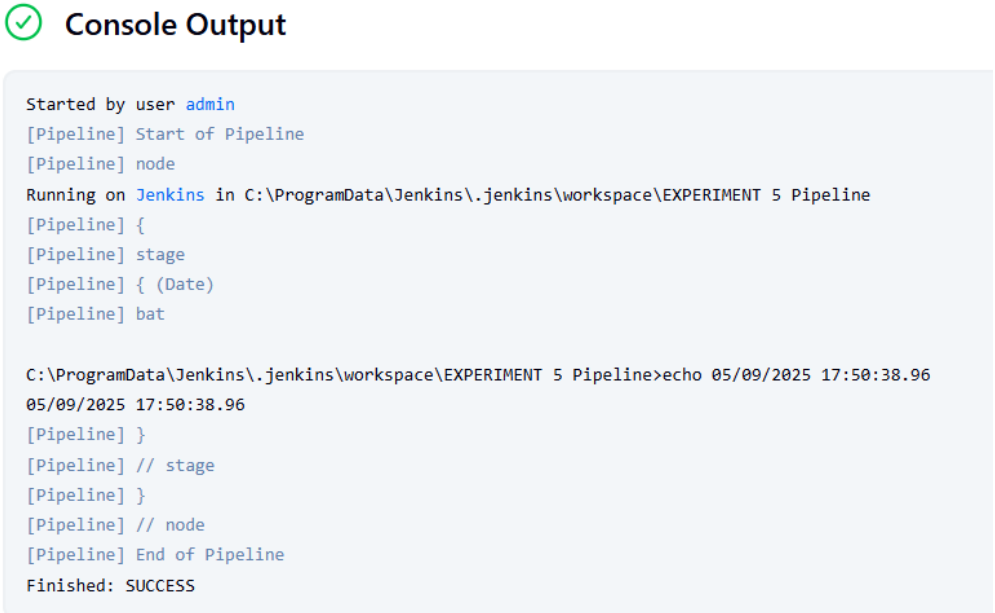
```
pipeline {
    agent any           // Run on any available Jenkins agent

    stages {
        stage('Hello') {    // Define a stage named "Hello"
            steps {         // Inside the stage, define the steps
                echo 'Hello World'  // Print "Hello World" in the console log
            }
        }
    }
}
```

Output :

Script 2 : Show Date and Time.

```
pipeline {
    agent any
    stages {
        stage('Date') {
            steps {
                bat 'echo %DATE% %TIME%'
            }
        }
    }
}
```

Output :



Script 3 : Add two numbers and print the result.

```
pipeline {
    agent any
    stages {
        stage('Add Numbers') {
            steps {
                powershell '''
                # two numbers
                $a = 5
                $b = 7
                $sum = $a + $b
                Write-Output "The sum is $sum"
                '''
            }
        }
    }
}
```

Output :

Script 4 : Multiply two numbers and print the result.

```
pipeline {
    agent any
    stages {
        stage('Multiply Numbers') {
            steps {
                bat '''
                set /a a=8
                set /a b=3
                set /a product=%a%*%b%
                echo %a% times %b% equals %product%
```

```
            '''
        }
    }
  }
}
```

Output :

Conclusion :

The pipeline for building, testing, and deploying an application was successfully created in Jenkins using Maven/Gradle/Ant.
It automates the build and deployment to the Tomcat server, enabling faster and consistent delivery.