# Experiment No. : 4

**Title:** Implementation of Continuous Integration /Continuous Development with Jenkins and Java & cloning in GitHub

# Objectives:

To implement Continuous Integration /Continuous Development with Jenkins and Java.

# Theory:

Continuous Integration (CI) and Continuous Development (CD) are

important practices in software development that can be achieved using

Jenkins. Here's an example of how you can demonstrate CI/CD using

Jenkins:

Create a simple Java application

# Continuous Integration (CI) and Continuous Development (CD) are important practices in software development that can be achieved using Jenkins. Here's an example of how you can demonstrate CI/CD using Jenkins: Create a simple Java application

# Create a simple Java application that you want to integrate with Jenkins.

# • The application should have some basic functionality, such as printing "Hello World" or performing simple calculations.

# Commit the code to a Git repository:

# • Create a Git repository for the application and commit the code to the repository.

# • Make sure that the Git repository is accessible from the Jenkins server.

# Create a Jenkins job:

# • Log in to the Jenkins web interface and create a new job.

# • Configure the job to build the Java application from the Git repository.

# • Specify the build triggers, such as building after every commit to the repository. Build the application:

# • Trigger a build of the application using the Jenkins job.

# • The build should compile the code, run any tests, and produce an executable jar file.

# Monitor the build:

# • Monitor the build progress in the Jenkins web interface.

# • The build should show the build log, test results, and the status of the build.

# Deploy the application:

# • If the build is successful, configure the Jenkins job to deploy the application to a production environment.

# • The deployment could be as simple as copying the jar file to a production server or using a more sophisticated deployment process, such as using a containerization technology like Docker.

# Repeat the process:

# • Repeat the process for subsequent changes to the application.

# 

# Jenkins should automatically build and deploy the changes to the production environment.

# This is a basic example of how you can use Jenkins to demonstrate CI/CD in software development. In a real-world scenario, you would likely have more complex requirements, such as multiple environments, different types of tests, and a more sophisticated deployment process.