**EXPERIMENT NO: 5. Demonstrate continuous integration and development using Jenkins.**

Aim: Demonstrate continuous integration and development using Jenkins.

DESCRIPTION

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.

**EXPERIMENT NO.: 6. Explore Docker commands for content management.**

AIM: Explore Docker commands for content management.

DESCRIPTION:

Docker is a containerization technology that is widely used for managing application containers. Here are some commonly used Docker commands for content management:

Docker run: Run a command in a new container.

For example:

$ docker run --name mycontainer -it ubuntu:16.04 /bin/bash

This command runs a new container based on the Ubuntu 16.04 image and starts a shell session in the container.

Docker start: Start one or more stopped containers.

For example:

$ docker start mycontainer

This command starts the container named "mycontainer".

Docker stop: Stop one or more running containers.

For example:

$ docker stop mycontainer

This command stops the container named "mycontainer".

Docker rm: Remove one or more containers.

For example:

$ docker rm mycontainer

This command removes the container named "mycontainer".

Docker ps: List containers.

For example: $ docker ps

This command lists all running containers.

Docker images: List images.

For example: $ docker images

This command lists all images stored locally on the host.

Docker pull: Pull an image or a repository from a registry.

For example: $ docker pull ubuntu:16.04

This command pulls the Ubuntu 16.04 image from the Docker Hub registry.

Docker push: Push an image or a repository to a registry.

For example:

$ docker push myimage

This command pushes the image named "myimage" to the Docker Hub registry.