**Version Control**

**GIT**

GIT is the open source distributed version control system that facilitates GITHUB activities on your laptop or desktop.

**git fetch vs git pull**

Git pull updates your current local working branch with all new commits from the corresponding remote branch on GitHub. git pull is a combination of git fetch and git merge.

**GitHub**: a platform for hosting and collaborating on Git repositories

**commit**: a Git object, a snapshot of your entire repository compressed into a SHA

**branch**: a lightweight movable pointer to a commit

**clone**: a local version of a repository, including all commits and branches

**remote**: a common repository on GitHub that all team member use to exchange their changes

**fork**: a copy of a repository on GitHub owned by a different user

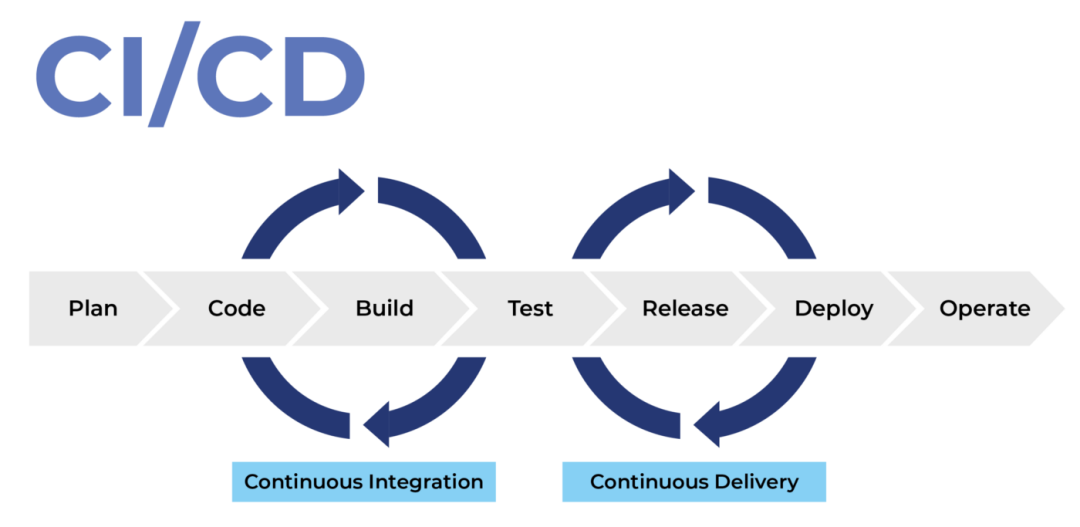
**pull request**: a place to compare and discuss the differences introduced on a branch with reviews, comments, integrated tests, and more

**HEAD**: representing your current working directory, the HEAD pointer can be moved to different branches, tags, or commits when using git checkout

**CVS vs DVCS :**

A Centralized Version Control system (CVS) contains a server and a client. but in DVCS, each client has own version and changes history (local server).

**WORKFLOW :**

Workflows are essential to ensure code is managed correctly and reduce mistakes from happening.

## Continuous Integration

## Continuous Integration, or CI, is used to automate the integration of code changes from multiple developers into a single main stream. CI is often used to automatically compile the project and run tests on every code change to ensure that the build remains stable and prevent regressions in functionality.

## Continuous Delivery

## Continuous Delivery is an extension of Continuous Integration. Once the changes have been merged into the main stream, a Continuous Delivery system automatically packages the application and prepares it for deployment.This helps avoid human error when packaging the application.

## Continuous Deployment

## Continuous Deployment is an extension of Continuous Delivery. The goal of Continuous Deployment is to deploy and release software to customers frequently and safely. The strategy commonly involves automatically deploying to a test (also known as staging) environment first to validate the deployment package and software changes. Once validated, it can automatically deploy to the live (also known as production) environment for customers.