**Version Control**

**GIT**

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

**Revision History**

Revision History is a record of all changes within a project. It allows you to pinpoint who made the changes, when they were made and what was changed.

**Git fetch:** Downloads all history from the remote tracking branches

**Git merge:** Combines remote tracking branch into current local branch

**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

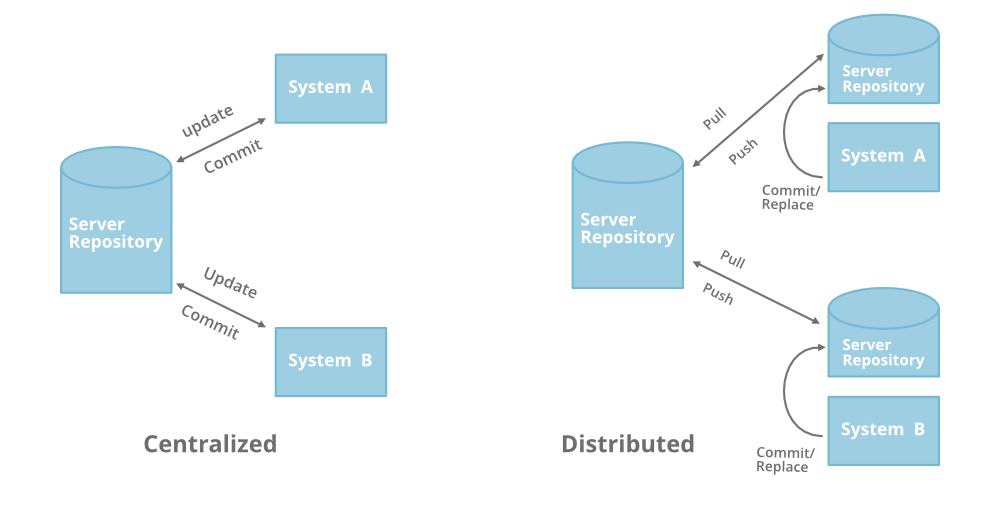
**remote**: a common repository on GitHub that all team member use to exchange their changes

**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

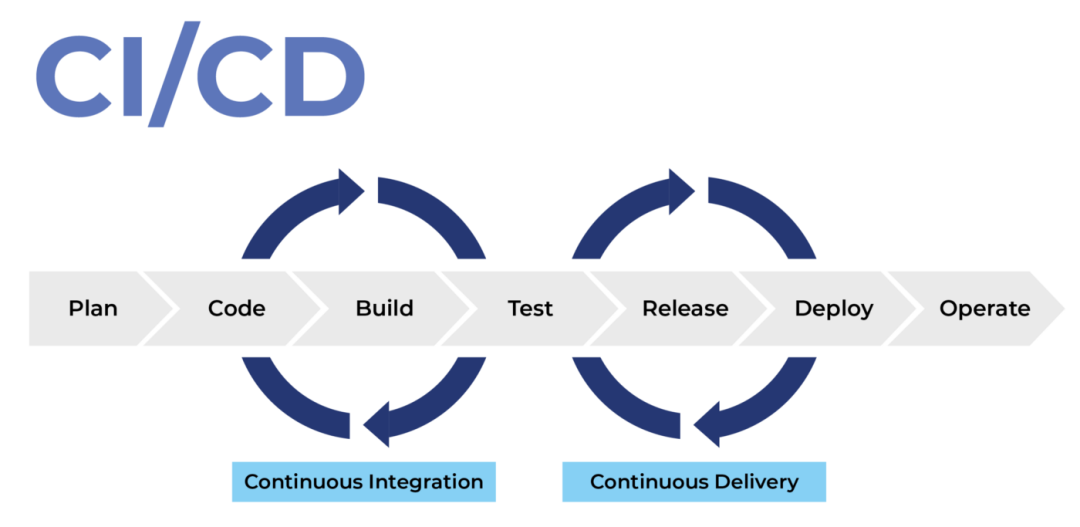
**CVS vs DVCS :**

A Centralized Version Control system (CVS) contains a server and a client (***Subversion(SVN) or Concurrent Version System(CVS)***). but in DVCS***( Git,*** [***Mercurial***](https://www.mercurial-scm.org/)), each client has own version and changes history (local server).

**STAGING**: This is the release candidate, and this environment is normally a mirror of the production environment. The staging area contains the "next" version of the application and is used for final stress testing and client/manager approvals before going live.

**PRODUCTION**: This is the currently released version of the application, accessible to the client/end users. This version preferably does not change except for during scheduled releases.

**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.Thishelps 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.

**Git Vs GitHub:**

**Git** is a version control system for tracking changes to projects. It's fast, reliable(fiable), open source and accessible syntax. **GitHub** is a cloud-based hosting service that lets you manage GIT repositories from user interface.

## Git WorkFlow

## 

## \* To remove file from staged : >> git restore --stage fileName.ext