**GIT and GITHUB**

**What is GIT and GITHUB?**

**Full Form of GIT:** Global Information Tracking System (GIT). GIT is also known as Version Control System. Purpose of GIT is to store, to tracking, to sharing. In **Azile Methodology** and **Waterfall Methodology** and we use tracking code separately through Git hub. GIT acts as medium where the development environment can create a file, that file would be transferred to storage. GITHUB is a platform used to maintain a code, store a code, write a code, share a code, tracking code. We can send any files to GIT HUB. Content files /folder will send. Empty files would not be sent.

**GITHUB:** GitHub, on the other hand, is a platform built around Git. It provides hosting for Git repositories, along with a web-based interface and additional features to facilitate collaboration. GitHub offers tools for code review, issue tracking, project management, continuous integration, and more. It allows developers to store their Git repositories remotely on GitHub's servers, making it easy to share code with others, collaborate on projects, and contribute to open-source projects. GitHub also provides social features like following other users, starring repositories, and forking projects.

In summary, Git is the version control system itself, while GitHub is a platform built around Git that provides hosting for repositories and additional collaboration features.

**What is Deployment?**

**Deployment:** We can create a code that code is run through device is called Deployment(layman). Deployment is transferring the current application code to a server which is properly tested.

“Sub folders” are called **Branches**

“Main Folders” are called **Repository**

**What is Key features of GIT?**

**Git** is a distributed version control system widely used for tracking changes in source code during software development. It offers numerous features that facilitate collaborative development, code management and project tracking. Here are some key features of Git:

**Disturbed Development**: It allows multiple developers to same work Concurrently (laymen terms) Git allows multiple developers to work on same project concurrently. Each developer has their own copy of repository. This distributed model enables developers to work offline and later synchronise their changes with the main repository.

**Branching & Merging:** Git provides robust branching and merging capabilities, allowing developers to create separate branches to work on features or fixes independently. Branches can be merged back into the main codebase easily, facilitating collaboration and parallel development workflows. collaborative coding.

**Commit Tracking:** Git tracks changes to files through commits, which capture snapshots of the repository at specific points in time. Each commit includes metadata such as the author, timestamp, and a unique identifier, enabling developers to trace the history of changes and understand why specific modifications were made.

**Staging Area (Index):** Git introduces a staging area (also known as the index), which acts as a middle ground between the working directory and the repository. Developers can selectively stage changes before committing them, allowing for more granular control over which modifications are included in each commit.

**Fast Performance**: Git is designed to be fast and efficient, even with large repositories and extensive histories. It employs various optimizations, such as delta compression and binary-diff storage, to minimize storage and transfer overhead, enabling quick operations even on complex projects.

**Security**: Git ensures data integrity and security through cryptographic hashing. Each object in the Git repository is identified by a unique SHA-1 hash, which guarantees the integrity of the data. Additionally, Git supports various authentication mechanisms, including SSH and HTTPS, to control access to repositories.

**Collaboration Tools**: Git integrates with various collaboration tools and platforms, such as GitHub, GitLab, and Bitbucket, which provide additional features like issue tracking, pull requests, code reviews, and project management. These platforms enhance team productivity and streamline the development workflow.

**Flexible Workflows**: Git accommodates a wide range of workflows, from centralized to decentralized, depending on the needs of the development team. Whether following a feature branch workflow, Git flow, or a customized approach, Git provides the flexibility to adapt to different project requirements and team dynamics.

**Open Source and Community Support**: Git is an open-source project with a large and active community of developers. This community contributes to the ongoing development and improvement of Git, as well as providing support, documentation, and a wealth of third-party tools and extensions.

**Customizability and Extensibility**: Git is highly customizable and extensible, allowing users to configure various aspects of its behavior and integrate with other tools seamlessly. Developers can create custom scripts, hooks, and plugins to automate tasks, enforce workflows, and extend Git's functionality according to their specific needs.