****GIT AND GITHUB NOTES:-****

****Git:****

****What is Git?:** Git** is a distributed version control system designed for tracking changes in source code during software development. It allows multiple developers to collaborate on projects by managing revisions, facilitating teamwork, and providing mechanisms for branching, merging, and tracking changes over time. Git is widely used in software development for its efficiency, flexibility, and robust version control capabilities.

****Key Concepts:****

****Repository (Repo):** A directory or storage space where your project's files are kept, along with the history of changes.**

****Commit:** A snapshot of your project at a point in time. Each commit has a unique identifier.**

****Branch:** A parallel version of your repository, allowing you to work on different features independently.**

****Merge:** Combining changes from different branches into the main branch.**

****Clone:** Creating a copy of a repository on your local machine.**

****Push/Pull:** Sending changes from your local repository to a remote repository (push) or fetching changes from a remote repository to your local repository (pull).**

****Basic Workflow:****

**Initialize a Git repository (git init).**

**Add files to the staging area (git add).**

**Commit changes to the local repository (git commit).**

**Push changes to a remote repository (git push).**

**Pull changes from a remote repository (git pull).**

****GITHUB:****

****What is GitHub?:** GitHub is a web-based platform that provides hosting for Git repositories and collaboration tools for software development projects.**

****Key Features:****

****Repository Hosting:** Allows users to store and manage Git repositories online.**

****Collaboration:** Facilitates teamwork by providing features such as pull requests, issues, and project boards.**

****Documentation:** Supports project documentation through README files and wikis.**

****Integration:** Integrates with various tools and services, including CI/CD pipelines and project management tools.**

****Common Use Cases:****

****Open Source Projects:** Many open-source projects host their code on GitHub, allowing for community contributions and collaboration.**

****Team Collaboration:** Teams use GitHub to work together on projects, manage tasks, and review code changes.**

****Portfolio:** Developers often use GitHub to showcase their projects and contributions to potential employers.**

****Workflow with GitHub:****

**Create a repository on GitHub.**

**Clone the repository to your local machine (git clone).**

**Make changes, stage them, commit them locally, and push them to GitHub.**

**Collaborate with others by creating pull requests, reviewing code, and merging changes.**

****Working directory****

****Staging area****

****Local repository****

****Remote repository****

****Working Directory****: This is where you do all your work: creating, editing, and organizing files. It's essentially your project folder on your local machine.

****Staging Area****: Also known as the index, this is where you prepare your changes before committing them to the repository. You use commands like **git add** to move changes from the working directory to the staging area.

****Local Repository****: This is where Git stores all the snapshots of your project's history on your local machine. When you commit changes, Git stores them in the local repository.

****Remote Repository****: This is a copy of your repository stored on a server, usually hosted by a service like GitHub. It serves as a central location for collaboration and backup. When working with others, you push your local changes to the remote repository and pull their changes into your local repository.

****Working dir to staging area (add files)****

****staging area to local rep(commit)****

****Local rep to Remote rep (push)****

****Working Directory to Staging Area (Add Files)****: You use the **git add** command to move changes from the working directory to the staging area. This prepares them for the next commit.

****Staging Area to Local Repository (Commit)****: Once your changes are staged, you use the **git commit** command to permanently save them to the local repository. Each commit represents a snapshot of your project at a certain point in time.

****Local Repository to Remote Repository (Push)****: After committing your changes locally, you use the **git push** command to upload them to the remote repository. This allows others to see your changes and collaborate on the project.

****we can send one by one file to staging area from working directory****

****from staging to local we will commit all the files whatever we needed****

****GIT COMMANDS :-****

****git****

****git init****

****git status****

****git add (file name)****

****git commit -m " (info msg) "****

****git push (after adding origin of repository)****

****git clone****

****git remote add origin https://github.com/sainavatej/git-tuts.git****

****git push -u origin main****

****git checkout(to create new branch)****

****git clone****

****cd( dir or change directory)****

****git****:

* + This is the base command for executing various Git operations. It serves as the entry point for interacting with Git and its subcommands.

****git init****:

* + Initializes a new Git repository in the current directory. This command creates a **.git** directory, which contains all the necessary files for version control.

****git status****:

* + Shows the current status of the repository, including any changes to files (tracked or untracked), files staged for commit, and the branch you're currently on. It helps you understand what's happening in your repository.

****git add (file name)****:

* + Adds changes in the specified file(s) to the staging area. This command prepares the changes to be included in the next commit.

****git commit -m " (info msg) "****:

* + Commits the changes staged in the staging area to the local repository, along with a descriptive message. This creates a snapshot of the changes and records them in the repository's history.

****git push****:

* + Uploads local repository commits to a remote repository. It's typically used after adding a remote repository with **git remote add** and specifying the branch to push to.

****git clone****:

* + Creates a local copy of a remote repository. This command clones the entire repository, including all branches and commit history, onto your local machine.

****git remote add origin**** [https://github.com/sainavatej/git-tuts.git](https://github.com/sainavatej/git-tuts.git" \t "https://chat.openai.com/c/_new):

* + Adds a remote repository with the specified name (**origin**) and URL. This command establishes a connection between your local repository and the remote repository hosted on GitHub (or another platform).

****git push -u origin main****:

* + Pushes the local branch (typically **main** or **master**) to the remote repository (**origin**) and sets it as the upstream branch. Subsequent pushes can be done with just **git push** after this initial setup.

****git checkout (to create new branch)****:

* + Creates a new branch or switches to an existing branch. If you specify a new branch name, Git creates it and makes it the current branch.

****git fetch**** :-

The **git fetch** command is used to retrieve updates from a remote repository without merging them into your local branches. It downloads new data from the remote repository, including branches, tags, and commits, but does not automatically merge any changes into your working branch.

****cd (dir or change directory)****:

* + Changes the current directory in the command line interface. This command is not specific to Git but is commonly used in conjunction with Git to navigate to different directories where repositories are located.

****GITHUB (Repository creation)- Read me file,git ignore file****

****Branches---****

****Main/Master (main code will present which we deploy)****

****PULL REQUEST(MERGE) from our local branch to main or master (reviewed by team lead and then merged)****