CREATE A TUTORIAL FOR GIT AND GITHUBS WITH THE COMMANDS

INTRODUCTION

What is Git?

Git is a free, open-source version control system that helps developers track and manage changes to code. It's the most widely used version control system in the world.

What is Github ?

GitHub is a web-based platform that hosts Git repositories, enabling developers to collaborate on code, track changes, and manage projects, while also facilitating open-source software development.

Purpose of Git :

Git's primary purpose is to act as a distributed version control system (VCS), allowing developers to efficiently track changes in their code, collaborate on projects, and revert to previous versions if needed.

Here's a more detailed explanation:

- **Version Control:**

  Git keeps a history of all changes made to files, allowing developers to see how a project has evolved over time and revert to specific versions.

- **Collaboration:**

  Git enables multiple developers to work on the same project simultaneously, merging their changes efficiently and resolving conflicts.

- **Distributed Nature:**

  Unlike centralized systems, Git allows each developer to have a complete copy of the project's history locally, enabling offline work and faster collaboration.

- **Branching and Merging:**

  Git's branching model allows developers to create isolated environments for new features or bug fixes, which can then be merged back into the main codebase.

- **Efficiency:**

  Git is designed to be fast and efficient, even with large projects, making it a popular choice for software development teams.

- **Open Source:**

  Git is a free and open-source software, meaning it's available for anyone to use and contribute.

What is Git command line ?

Basic Git Commands

1. **Initialize a Git repository**
   Initializes a new Git repository in your current directory.

   ```
   bash
   Copy
   git init
   ```

2. **Clone a repository**
   Clone a remote repository to your local machine.

   ```
   bash
   Copy
   git clone <repository-url>
   ```

3. **Check repository status**
   Shows the status of changes (e.g., modified, staged, untracked).

   ```
   bash
   Copy
   git status
   ```

4. **Add files to staging area**
   Stage changes to a specific file (or all files).

   ```
   bash
   Copy
   git add <file-name>
   git add .
   ```

5. **Commit changes**
   Commit the staged changes with a descriptive message.

   ```
   bash
   Copy
   git commit -m "Commit message"
   ```

6. **View commit history**
   View the commit history of the repository.

   ```
   bash
   Copy
   git log
   ```

7. **Push changes to remote repository**
   Push your commits to a remote repository (e.g., GitHub).

   ```
   bash
   Copy
   ```

```
git push origin <branch-name>
```

8. **Pull changes from remote repository**
   Fetch and merge changes from the remote repository into your local repository.

```
bash
Copy
git pull origin <branch-name>
```

9. **Create a new branch**
   Create a new branch in your repository.

```
bash
Copy
git branch <branch-name>
```

10. **Switch branches**
    Switch to a different branch.

```
bash
Copy
git checkout <branch-name>
```

11. **Merge branches**
    Merge changes from one branch into another.

```
bash
Copy
git merge <branch-name>
```

12. **Delete a branch**
    Delete a local branch that you no longer need.

```
bash
Copy
git branch -d <branch-name>
```

13. **Show changes made in files**
    See the differences between your working directory and the last commit.

```
bash
Copy
git diff
```

14. **Remove a file**
    Delete a file from the repository and stage it for commit.

```
bash
Copy
git rm <file-name>
```

15. **Undo changes**
    Revert a file back to the last committed state.

```
bash
Copy
git checkout -- <file-name>
```

16. **Stash changes**
    Temporarily save changes that you don't want to commit right away.

```
bash
Copy
git stash
```

17. **Apply stashed changes**
    Retrieve stashed changes.

```
bash
Copy
git stash apply
```

18. **View remotes**
    Show a list of remote repositories linked to your local repo.

```
bash
Copy
git remote -v
```

19. **Add a remote**
    Add a new remote repository.

```
bash
Copy
git remote add <remote-name> <repository-url>
```