Git documentation

Git Basics

- **ls:** Lists files and directories in your current working directory.
- **ls -a:** Lists all files and directories, including hidden ones (starting with a dot).

Initializing a Git Repository

- **git init:** Creates a new Git repository in the current directory. This is the first step to start version controlling your project.

Configuring Git

- **git config --global user.name "sudusikchi":** Sets your global Git username, which will be used in commit messages.
- **git config --global user.email "sudershan2222@gmail.com":** Sets your global Git email address, also used in commit messages.

Staging and Committing Changes

- **git add .:** Adds all modified and newly created files in the current directory and its subdirectories to the staging area. The staging area is where you prepare changes before committing them to your Git repository.
- **git status:** Shows the status of your working directory and staging area. It indicates which files have been modified, staged, or untracked (not yet added to Git).
- **git commit -m "Your commit message":** Creates a snapshot of the staged changes and stores it in the Git repository history, along with your commit message. This message should describe the changes you made.

Remote Repositories and Collaboration

- **git remote add origin https://github.com/sudusikchi/MyProject.git:** Defines a remote repository named "origin" that points to your project on GitHub. This allows you to push and pull changes between your local repository and the remote one.

Pushing and Pulling Changes

- **git push origin master:** Pushes your local master branch to the remote origin repository (GitHub in this case). This shares your commits with others who have access to the remote repository.
- **git pull origin master:** Pulls changes from the remote master branch on GitHub and merges them into your local master branch. This keeps your local repository up-to-date with the remote one.

Diff Tool Commands

- **git diff:** Shows the difference between your working directory and the staging area, or between the staging area and the latest commit (HEAD). It highlights changes made to lines of code.
- **git difftool:** Opens a visual diff tool to compare versions of files, making it easier to see the changes. You may need to install a separate diff tool for this to work.

Staged Diff

- **git diff --staged:** Shows the difference between the index (staging area) and the last commit. This is useful to see what specific changes you've staged for the next commit.
- **git difftool --staged:** Opens a visual diff tool to compare the staged changes with the last commit.

Uncommitted Diff

- **git diff HEAD:** Shows the difference between your working directory and the last commit (HEAD). This allows you to see all the changes you've made since the last commit, including staged and unstaged ones.
- **git difftool HEAD:** Opens a visual diff tool to compare your working directory with the last commit.

Viewing Commit History

- **git log:** Lists all commits made in the repository, showing commit SHA-1 hashes, author information, date, and commit message. You can use various options with git log to filter or format the output.
- **git show <commit_hash>:** Shows the details of a specific commit, including the commit message, author information, date, and a diff of the changes made in that commit.

Shortened Status

- **git status s:** Provides a concise summary of the status of your working directory and staging area, similar to the full git status command but with less detail.

Incorrect Commands

- **git add.:** This command is missing a space after add. The correct syntax is git add <file>, where <file> is the filename you want to add to the staging area.
- **git commint -m "#":** There's a typo in commint. The correct command is git commit -m "Your commit message". Also, remember to replace "#" with a meaningful commit message describing your changes.

Ignoring Files (Creating a .gitignore File)

1. Create a new file named .gitignore in your project's root directory.
2. In the .gitignore file, list all file patterns