Git and GitHub

What is Git?

Git is a distributed version control system used to manage and track changes in software development projects. It is a free and open-source tool that was created by Linus Torvalds in 2005. Git is designed to handle everything from small to very large projects with speed and efficiency. It allows developers to work on the same codebase without interfering with each other's work. Git's popularity has increased over the years, and it is now the most widely used version control system in the world.

Here are some common Git commands:

- `git init`: Initializes a new Git repository
- `git clone`: Copies a repository onto your local machine
- `git add`: Adds changes to the staging area
- `git commit`: Saves changes to the local repository
- `git push`: Uploads changes to a remote repository
- `git pull`: Downloads changes from a remote repository
- `git branch`: Lists all branches in the repository

- `git checkout`: Switches to a different branch or commit
- `git merge`: Merges changes from one branch into another
- `git status`: Shows the current status of the repository
- `git log`: Shows the commit history
- `git diff`: Shows the difference between two versions of a file

The `git clone` command is used to copy a repository onto your local machine. This creates a local copy of the remote repository, including all branches and commit history. You can then make changes to the code locally and upload them to the remote repository using the `git push` command.

The `git commit -m"file.txt is added"` command is used to save changes to the local repository with a commit message "file.txt is added". The `-m` flag is used to add a message to the commit, which is a brief description of the changes made in the commit. This message helps other developers understand the changes made in the commit.

The `cat` command is a command-line utility that is used to concatenate and display files. It can be used to display the contents of a file, create a new file, or append the contents of one or more files to an existing file. For example, the command `cat file.txt` will display the contents of the `file.txt` file in the terminal.

To open a folder in the terminal, you can use the `cd` command followed by the path to the folder. For example, if you want to open a folder named "my_folder" on your desktop, you would use the command `cd ~/Desktop/my_folder`. This will change the current working directory to "my_folder" and allow you to run commands in that folder.

To commit a file, you can use the `git commit` command followed by the file name. For example, `git commit file.txt` will save changes made to the `file.txt` file to the local repository. You can also add a commit message using the `-m` flag, like this: `git commit -m "added file.txt"`.

In Git, a fork is a copy of a repository that allows you to freely experiment with changes without affecting the original repository. Forking a repository creates a new copy on your GitHub account that you can modify and push changes to. This is useful when you want to contribute to an open-source project or work on your own version of an existing repository. Once you have made changes to the fork, you can create a pull request to propose changes to the original repository. The owner of the original repository can then review your changes and decide whether to merge them into the main codebase.

To clone a forked repository to your local machine, you can use the `git clone` command followed by the URL of the forked repository. Here are the steps to do this:

1. On GitHub, navigate to the forked repository that you want to clone.
2. Click the "Code" button and copy the URL of the repository.
3. Open a terminal and use the `cd` command to navigate to the directory where you want to clone the repository.
4. Use the command `git clone` followed by the URL of the repository. For example, `git clone <https://github.com/your-username/repo-name.git` >.
5. Press Enter and Git will clone the repository to your local machine.

Once the repository is cloned, you can make changes to the code and push them back to the remote repository using the `git push` command.

The `git remote add upstream url` command is used to add a new remote repository to a local Git repository. The `upstream` repository is commonly used when working with forks of open-source projects. When you fork a repository, you create a copy of the repository on your own account. You can then make changes to the code and push them to your forked repository. However, if the original repository is also being updated, you may want to pull those changes into your forked repository. To do this, you can add the original repository as a remote repository using the `git remote add` command. The `upstream` keyword is commonly used to refer to the original repository. The `url` parameter is the URL of the repository you want to add as a remote. Once the upstream repository is added, you can use the `git fetch upstream` command to download changes

from the upstream repository, and the `git merge upstream/master` command to merge those changes into your local repository. This allows you to keep your forked repository up-to-date with the original repository.

In Git, a commit is a snapshot of the repository at a specific point in time. When you make changes to the code, you can use the `git commit` command to save those changes to the local repository. Each commit has a unique identifier and a commit message, which is a brief description of the changes made in that commit. Commits are used to track the history of changes made to the code over time, and they can be used to revert changes or compare different versions of the code.

To create a new branch in Git, you can use the `git branch` command followed by the name of the new branch. For example, to create a new branch called "my-feature-branch", you would use the command `git branch my-feature-branch`. This will create a new branch based on the current branch you are on. To switch to the new branch, you can use the `git checkout` command followed by the name of the new branch. For example, `git checkout my-feature-branch` will switch to the "my-feature-branch" branch. Alternatively, you can use the `git checkout -b` command followed by the name of the new branch to create and switch to the new branch in one step. For example, `git checkout -b my-feature-branch` will create a new branch called "my-feature-branch" and switch to it.

In Git, a pull request is a mechanism for proposing changes to a repository. When you make changes to a forked repository, you can submit a pull request to the original repository to ask the owner to merge your changes into the main codebase. Pull requests are useful when you want to contribute to an open-source project or collaborate with other developers on a shared codebase. The pull request allows the owner of the original repository to review your changes and provide feedback before deciding whether to merge them. Once the pull request is accepted and merged, your changes will become part of the main codebase.

The `vi file.txt` command is used to open the `file.txt` file in the vi text editor. Vi is a command-line text editor that is commonly used in Unix and Linux systems. Once the

file is open in vi, you can edit the contents of the file and save your changes using various vi commands. For example, to insert text into the file, you can press the `i` key to enter insert mode, type the text you want to insert, and then press the `Esc` key to exit insert mode. To save your changes and exit vi, you can type `:wq` and then press Enter. This will write your changes to the file and quit vi.

The `git push origin` command is used to upload changes to a remote Git repository. The `origin` keyword specifies the remote repository that changes should be pushed to. For example, if you have a repository on GitHub, you would use `git push origin` to upload changes to that repository. When you run this command, Git will upload all changes that have been committed to the local repository since the last push. If there are conflicts between the local and remote repositories, Git will prompt you to resolve them before continuing with the push.

The `git log` command is used to display the commit history of a Git repository. It shows a list of all commits, with the most recent commits listed first. Each commit is listed with its unique identifier, author, date, and commit message. The `git log` command is useful for tracking the history of changes made to the codebase over time, and for identifying who made specific changes and when they were made.

The `git stash` command is used to save changes that are not yet ready to be committed to a separate "stash" area. This is useful when you need to switch to a different branch or work on a different task, but you don't want to commit your changes yet. The `git stash` command will save your changes and revert your working directory to the last committed state. You can then switch to a different branch or work on a different task. When you're ready to continue working on the changes you stashed, you can use the `git stash apply` command to restore the changes to your working directory.

The `git checkout main` command is used to switch to the `main` branch in a Git repository. This command changes the current branch to `main`, allowing you to work on that branch or view its commit history. When you run `git checkout main`, any changes you made on the previous branch will be removed and replaced with the contents of the `main` branch.

This command is useful when you want to work on a different branch or view the latest changes in the `main` branch.

The `git fetch --all --prune` command is used to download all changes from all remote repositories and remove any references to remote branches that no longer exist. The `--all` flag tells Git to fetch changes from all remote repositories, and the `--prune` flag tells Git to remove any references to remote branches that have been deleted. This command is useful when you want to update your local repository with changes from all remote repositories and ensure that your local repository is up-to-date with the latest changes. Once you have fetched changes from all remote repositories, you can use the `git merge` command to merge those changes into your local repository.

The `git pull upstream main` command is used to download changes from the `main` branch of the upstream repository and merge them into the local repository. The `upstream` keyword refers to the original repository that was forked, and `main` is the name of the branch from which changes will be pulled. This command is useful when you want to update your forked repository with changes from the original repository. Once the changes have been pulled, you can push them to your forked repository using the `git push` co

The `git push origin main` command is used to upload changes to the `main` branch of the remote repository specified by `origin`. The `origin` keyword specifies the remote repository, and `main` is the name of the branch that changes will be pushed to. When you run this command, Git will upload all changes that have been committed to the local `main` branch since the last push. If there are conflicts between the local and remote repositories, Git will prompt you to resolve them before continuing with the push.

In case of merging conflicts, you will need to manually resolve the conflicts before you can merge the changes. Git will automatically attempt to merge changes from two different branches, but if there are conflicting changes to the same file or lines of code, Git will not be able to automatically resolve the conflict. To resolve the conflict, you will need to manually edit the file to remove the conflicting changes and then commit the

changes. Once the conflicts are resolved, you can use the `git add` and `git commit` commands to save the changes and complete the merge.