GIT COMMANDS

Git is a distributed version control system used for tracking changes in source code during software development. It was created by Linus Torvalds in 2005 for managing the Linux kernel source code, and has since become widely used in the software development industry.

Git works by creating a repository (a database that tracks changes to files over time) and allowing users to make changes to the files in that repository. Users can make changes to the files and commit those changes to the repository, which creates a snapshot of the file at that point in time. This allows users to track changes over time, collaborate with others, and revert to previous versions of the files if necessary.

Some of the main benefits of using Git are:

1. Version control: Git allows developers to track changes to code over time, making it easy to roll back to previous versions if needed.
2. Collaboration: Git makes it easy for multiple developers to work on the same codebase, by allowing them to merge changes made by different people.
3. Branching and merging: Git makes it easy to create and switch between different branches of code, allowing developers to work on multiple features or bug fixes simultaneously.

Git commands are used to manage the Git repository and make changes to the files in that repository. Here are some commonly used Git commands:

1. **git init**:
Initializes a new Git repository.

Example:

```
$ git init my-project
```

This command initializes a new Git repository in the **my-project** directory.

2. **git clone**:
Creates a copy of an existing Git repository.

Example:

```
$ git clone https://github.com/username/repo.git
```

This command creates a local copy of the **repo** repository hosted on GitHub.

3. **git add**:
Adds changes to the staging area.

Example:

```
$ git add file.txt
```

This command adds the changes made to **file.txt** to the staging area.

4. **git commit**:
Records changes to the repository.

Example:

```
$ git commit -m "Add new feature"
```

This command records the changes made to the repository with a commit message "Add new feature".

5. **git push**:
Uploads local changes to a remote repository.

Example:

```
$ git push origin master
```

This command uploads the local changes to the **master** branch of the **origin** remote repository.

6. **git pull**:
Downloads changes from a remote repository.

Example:

```
$ git pull origin master
```

This command downloads the changes made to the **master** branch of the **origin** remote repository.

7. **git branch**:
Lists all available branches in the repository.

Example:

```
$ git branch
```

This command lists all available branches in the repository.

8. **git checkout**:
Switches between branches in the repository.

Example:

```
$ git checkout feature-branch
```

This command switches to the **feature-branch** branch in the repository.

9. **git merge**:
Combines changes from one branch into another.

Example:

```
$ git merge feature-branch
```

This command combines the changes made in the **feature-branch** branch into the current branch.

10. **git status**:
Displays the current status of the repository and any changes that have been made.

Example:

```
$ git status
```

This command displays the current status of the repository and any changes that have been made.

11. git log:
Shows a list of all the commits made in the repository.

Example:

```
$ git log
```

This command shows a list of all the commits made in the repository.

These are just a few examples of the many Git commands available. It's important to understand the purpose and usage of each command in order to manage and collaborate on code effectively.