



7 important **git** commands





git clone

`git clone` copies an existing Git repository into a new local directory.

The `git clone` command will **create a new local directory** for the repository, copy all the contents of the specified repository, create the **remote-tracked branches**, and check out an initial branch **locally**.



```
git clone <repository url>
```



Swipe →



git pull

The **git pull** command retrieves and downloads content from a remote repository and **updates the local repository** as soon as it has been downloaded.

In Git-based collaboration workflows, it is common to merge remote upstream changes into your local repository.



```
git pull
```



Swipe →



git checkout

The **git checkout** command navigates between two different branches in a Git repository.

checkout is used to view and make changes to different branches. You can check out a **past commit** in a repository to view how your project appeared in that state.



```
git checkout <branch name>  
git checkout -b <branch name>
```



Swipe →



git add

The **git add** command adds new or changed files in your working directory to the Git staging area. As you're working, you **change and save** a file or multiple files.

Then, **before you commit, you must git add.** This step allows you to choose what you are going to commit.



```
git add <file path>  
git add .
```



Swipe →



git commit

Used to **record the changes** in the repository. Every commit contains the **index data** and the **commit message** and every commit form a parent-child relationship.

When we add a file in Git, it will take place in the staging area. A commit command is used to **fetch updates** from the **staging area to the repository**.



```
git commit -m "<commit message>"
```





git push

A **git push** command, when executed, pushes the changes that the user has made on the **local machine** to the **remote repository**.

Once the users have cloned the remote repository and have made the necessary changes in their local device, they need to be **pushed to the remote repository** so that they are shared and used by other users.



```
git push  
git push --set-upstream  
<remote branch> <branch name>
```





git branch

Git branch provides you the ability to **create branches** and **perform risk-free development** in isolated programming space.

The "branch" command helps you create, delete, and list branches.

It's the go-to command when it comes to managing any aspect of your branches – no matter if in your **local repository** or on **your remote repository**.



```
git branch <new branch>  
git branch -d <branch name>
```



Swipe →