**GIT COMMANDS**

Contents:

1. About Git
2. About Git Hub
3. Git Tools
4. Git commands:

* Config
* Init
* Add
* Commit
* Clone
* Fork
* Checkout
* Rm
* Log
* Status
* Pull
* Push
* Merge
* About Git:

**Git is an open-source distributed version control system** which is available for everyone at zero cost.

It is designed to handle minor to major projects with speed and efficiency. It is developed to co-ordinate the work among programmers.

The version control allows you to track and work together with your team member at the same workspace.

* **About Git Hub:**

**GitHub is a Git repository hosting service**. It is a web-based service. GitHub facilitates with all of the features of distributed version control and source code management (SCM) functionality of Git.

It also supports some of its characteristics in a single software tool.

* **Git Tools:**

To explore the robust functionality of Git, we need some tools. Git comes with some of its tools like Git Bash, Git GUI to provide the interface between machine and user. It supports inbuilt as well as third-party tools.

Git comes with built-in GUI tools like **git bash**, **git-gui**, and **gitk** for committing and browsing. It also supports several third-party tools for users looking for platform-specific experience.

**Git bash:**

Git Bash is an application for the Windows environment. It is used as Git command line for windows.

Git Bash provides an emulation layer for a Git command-line experience. Bash is an abbreviation of **Bourne Again Shell**. Git package installer contains Bash, bash utilities, and Git on a Windows operating system.

Git Bash also includes the full set of Git core commands like **git clone, git commit, git checkout, git push**, and more.

Git-gui:

Git GUI is a powerful alternative to Git BASH. It offers a graphical version of the Git command line function, as well as comprehensive visual diff tools.

We can access it by simply right click on a folder or location in windows explorer. Also, we can access it through the command line by typing below command.

$ git gui

Gitk:

Gitk is a graphical history viewer tool. It's a robust GUI shell over **git log** and **git grep**. This tool is used to find something that happened in the past or visualize your project's history.

Gitk can invoke from the command-line. Just change directory into a Git repository.

* **Git Commands:**

1. **Git Config:**

This command configures the user. This command is the first and necessary command on the git command line. This command sets the author name and email address to be used with our commits.

Syntax: $ git config --global user.name “author”

$ git config --global user.email “[authors@gmail.com](mailto:authors@gmail.com)”

1. Git Init:

The git init command is used to create a new blank repository. It is used to make an existing project as a Git project.

Creating a Repository for a Blank project: To create a blank repository, open command line on your desired directory and run the init command as follows:

Syntax: $ git init

1. Git Add:

The git add command is used to add file contents to the [Index (Staging Area)](https://www.javatpoint.com/git-index).This command updates the current content of the working tree to the staging area. It also prepares the staged content for the next commit.

The git add command can be run many times before making a commit. These all add operations can be put under one commit.

Syntax: $ git add <File name>

1. Git Commit:

It is used to record the changes in the repository. It is the next command after the [git add](https://www.javatpoint.com/git-add). Every commit contains the index data and the commit message.

A commit command is used to fetch updates from the staging area to the repository.

The staging and committing are co-related to each other. Staging allows us to continue in making changes to the repository, and when we want to share these changes to the version control system, committing allows us to record these changes.

Syntax using -m: $ git commit -m "Commit message."

Syntax using -a: $ git commit -a

1. Git Clone:

In Git, cloning is the act of making a copy of any target repository. The target repository can be remote or local. You can clone your repository from the remote repository to create a local copy on your system.

Syntax: $ git clone **<repository** URL**>**

1. Git Fork:

A fork is a rough copy of a repository. Forking a repository allows you to freely test and debug with changes without affecting the original project. One of the excessive use of forking is to propose changes for bug fixing.

* Fork the repository.
* Make the fix.
* Forward a pull request to the project owner.

Note: Forking is not a Git function; it is a feature of Git service like GitHub.

1. Git Checkout:

In Git, the term checkout is used for the act of switching between different versions of a target entity. The **git checkout** command is used to switch between branches in a repository. Be careful with your staged files and commits when switching between branches.

The git checkout command operates upon three different entities which are files, commits, and branches. Sometimes this command can be dangerous because there is no undo option available on this command.

Operations on checkout: branch. Checkout branch

a) git branch: $ git branch (Now, you have the list of available branches).

b) git checkout: $ git checkout <branch name> (switching to newbranch).

c) Create and switch to newbranch: $ git checkout -b <branchname>

1. Git rm:

In Git, the term rm stands for remove. It is used to remove individual files or a collection of files.

Syntax: $ git rm <file Name>

1. Git log:

The advantage of a version control system is that it records changes. These records allow us to retrieve the data like commits, figuring out bugs, updates. But, all of this history will be useless if we cannot navigate it. At this point, we need the git log command.

Syntax: $ git log

1. Git Status:

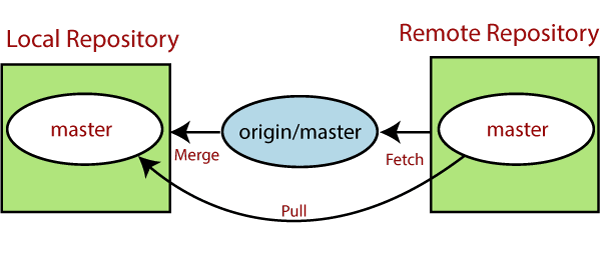
The git status command is used to display the state of the repository and staging area. It allows us to see the tracked, untracked files changes. This command will not show any commit records or information.

Syntax: $ git status

1. Git pull:

The term pull is used to receive data from GitHub. It fetches and merges changes from the remote server to your working directory. The **git pull command** is used to pull a repository.

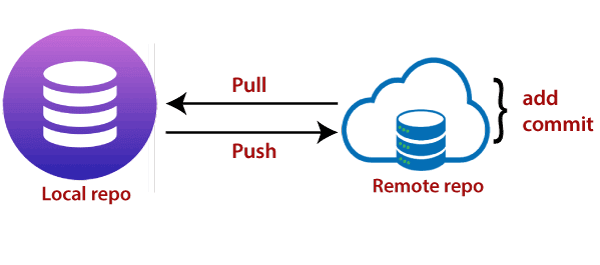
Syntax: $ git pull **<option>** [**<repository** URL**><refspec>**...]



1. Git push:

The push term refers to upload local repository content to a remote repository. Pushing is an act of transfer commits from your local repository to a remote repository. Pushing is capable of overwriting changes; caution should be taken when pushing.

Syntax: $ git push <option> [<Remote URL><branch name><refspec>



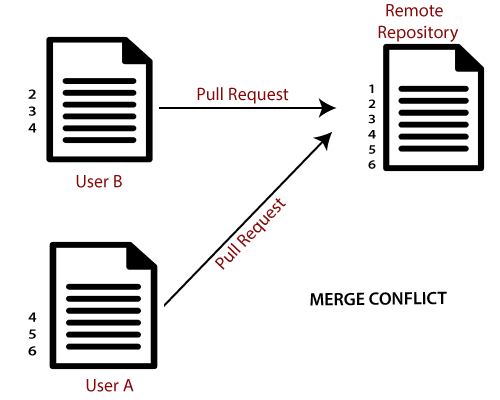
1. Git Merge:

The git merge command facilitates you to take the data created by git branch and integrate them into a single branch.

Syntax: $ git merge <Branchname>

Merge conflict:

When two branches are trying to merge, and both are edited at the same time and in the same file, Git won't be able to identify which version is to take for changes. Such a situation is called merge conflict.



**Git Commands:**

**Git Commands:**