(https://www.educba.com/software-development/)

← (https://www.educba.com/git-repository-setup/)

→ (https://www.educba.com/git-checkout/)


# Introduction to GIT Commands

The following article provides an outline for GIT Commands. GIT is a very popular version

their branch and can push their code to merge into the master branch, thereby creating a complete code solution.



The goals as to why it has been among the very popular tools are due to its integrity, speed, and the that it supports all the non-linear workflows of a distributed manner. The GIT directory present on every computer is in itself a complete repository which is not the case with many client-server systems operating today. While using with GIT bash, there are some commands which should be known to you. In this post, we are going to discuss those commands.

your commitment to addressing.

```
git config –global user.email “[email address]”
```

```
hardik@hardik-PC MINGW64 ~
$ git config -global user.email "Hardik.kec@gmail.com"
```

- **git init:** It is used to start a new git repository. This is generally used at the beginning.

```
git init [repo name]
```

```
hardik@hardik-PC MINGW64 ~
$ git init C:\Users\hardik\Downloads
Initialized empty Git repository in C:/Users/hardik/UsershardikDownloads/.git/
```

- **git clone:** This command is used to clone or copy a repository from a URL. This URL generally is a bitbucket server, a stash or any other version control and source code management repository holding service.

```
git clone [URL]
```

```
hardik@hardik-PC MINGW64 ~
$ git clone https://github.com/agconti/kaggle-titanic.git
Cloning into 'kaggle-titanic'...
remote: Enumerating objects: 472, done.
remote: Total 472 (delta 0), reused 0 (delta 0), pack-reused 472
Receiving objects: 100% (472/472), 7.61 MiB | 613.00 KiB/s, done.
Resolving deltas: 100% (216/216), done.
```

```
git add (filename),
git add *
```

```
hardik@hardik-PC MINGW64 ~/Downloads
$ git add project_1
```

```
hardik@hardik-PC MINGW64 ~/Downloads
$ git add *
```

- **git commit –m:** It is used to snapshot or record a file in its version history permanently.

```
git commit –m [type in a message]
```

Giving a message text at the end of the commit command helps in identifying the details about the commit code.

```
hardik@hardik-PC MINGW64 ~/Downloads
$ git commit -m "First commit"
```

- **git commit –a:** This commit command is used to commit any such file which has been added as a result of the git add command. It is also responsible for committing any other files to which you have brought a change to since then.

```
git commit -a
```

```
hardik@hardik-PC MINGW64 ~/Downloads
$ git commit -a
```

- **git diff:** As the name suggests, this command is used to display all the differences between

(https://www.educba.com/software-development/)

```
$ git diff
```

- **git diff –staged:** It is used to display all the differences between staging area files and the latest version, which might be present.

```
git diff -staged
```

```
hardik@hardik-PC MINGW64 ~/Downloads
$ git diff -staged
```

- **git diff [first branch] [second branch]:** This is a very effective command as it is used to display the differences present between the two branches. Generally, a single developer will be working on his individual branch, which will then be combined into a master branch.

```
git diff [first branch] [second branch]
```

```
hardik@hardik-PC MINGW64 ~/Downloads
$ git diff branch1 branch2
```

- **git reset [file]:** This command, as the name suggests, is used to unstage a file. Even though it unstages the file, still the contents of the file have stayed intact.

```
git reset [file]
```

```
hardik@hardik-PC MINGW64 ~/Downloads
$ git reset style.css
```

## Intermediate GIT Commands

on the computer.

```
git rest [commit]
```

```
hardik@hardik-PC MINGW64 ~/Downloads
$ git reset 0055cf6486416cd68ef684bg84g
```

- **Git reset –hard [commit]:** This command is used to discard all the history and takes us to the last specified commit.

```
git reset –hard [commit]
```

```
hardik@hardik-PC MINGW64 ~/Downloads
$ git reset --hard 0055cf6486416cd68ef684bg84g
```

- **Git status:** This is one of the most frequently used as this is used to list down all the files which are ready to be committed.

```
git status
```

```
hardik@hardik-PC MINGW64 ~/Downloads
$ git status
```

- **Git rm:** As in the Unix, rm is used to remove; in the same way, rm is used to delete the from the present working directory and is also used to stage the deletion process.

- **Git log:** This is used for listing down the version history for the current working branch.

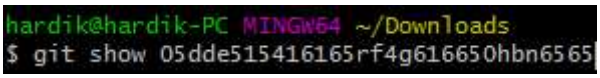
```
git log
```


- **git log –follow:** This is similar to that of git log with the additional difference that it lists the version history for a particular file, which often includes the renaming of the file also.

```
git log –follow [file]
```


- **git show:** This is used to display the metadata and all the content related changes of a particular commit.

```
git show [commit]
```




- **git tag:** This is used to give particular tags to the code commits.

```
git tag [commitID]
```

```
git branch
```

```
hardik@hardik-PC MINGW64 ~/Downloads
$ git branch
```

- **Git branch [branch-name]:** This is used to create a new branch.

```
Git branch [branch-name]
```

```
hardik@hardik-PC MINGW64 ~/Downloads
$ git branch branch_1
```

## Advanced Commands

The advanced commands are as follows:

- **Git branch –d [branch name]:** It is used to delete the current branch name specified.

```
git branch –d [branch name]
```

```
hardik@hardik-PC MINGW64 ~/Downloads
$ git branch -d branch_1
```

- **Git checkout:** It is helpful in switching from one branch to another.

```
git checkout [branch-name]
```

```
hardik@hardik-PC MINGW64 ~/Downloads
$ git checkout branch_2
```

(https://www.educba.com/software-development/)

- Many settings can be modified/altered as per your wish by making use of this command.
- You can turn particular command options off or on based on your choice and can also set the aliases.
- Another crucial point to mention is the visualization of the commit-graph, which comes in very handy when you are working on some project which consists of a lot of branching structures.

## Conclusion

The best way to memorize these commands is by making frequent use of them. Don't worry; if you don't have an official project, you can clone any repo from the stash and start working on GIT to get hands-on experience and a nice flavor.

## Recommended Articles

This has been a guide to GIT Commands. Here we have discussed concept, basic, intermediate as well as advanced GIT commands along with tips and tricks to use effectively. You may also look at the following article to learn more –

1. GitHub Actions (https://www.educba.com/github-actions/)
2. GitHub Commands List (https://www.educba.com/github-commands/)
3. Git Repository Setup (https://www.educba.com/git-repository-setup/)
4. Git Reset Hard (https://www.educba.com/git-reset-hard/)

(https://www.educba.com/software-development/)

development/courses/iot-course/?source=footer)

ASP.NET Course (https://www.educba.com/software-development/courses/asp-net-course/?source=footer)

VB.NET Course (https://www.educba.com/software-development/courses/vb-net-course/?source=footer)

PHP Course (https://www.educba.com/software-development/courses/php-course/?source=footer)