

# Git Cheat Sheet – 50 Git Commands You Should Know

---

## What is a Distributed Version Control System?

---

A distributed version control system is a system that helps you keep track of changes you've made to files in your project.

This change history lives on your local machine and lets you revert to a previous version of your project with ease in case something goes wrong.

Git makes collaboration easy. Everyone on the team can keep a full backup of the repositories they're working on on their local machine. Then, thanks to an external server like BitBucket, GitHub or GitLab, they can safely store the repository in a single place.

This way, different members of the team can copy it locally and everyone has a clear overview of all changes made by the whole team.

Git has many different commands you can use. And I've found that these fifty are the ones I use the most often (and are therefore the most helpful to remember).

So I have written them down and thought it'd be nice to share them with the community. I hope you find them useful – Enjoy.

## How to check your Git configuration:

---

The command below returns a list of information about your git configuration including user name and email:

```
git config -l
```

## How to setup your Git username:

---

With the command below you can configure your user name:

```
git config --global user.name "Fabio"
```

## How to setup your Git user email:

---

This command lets you setup the user email address you'll use in your commits.

```
git config --global user.email "signups@fabiopacifici.com"
```

## How to cache your login credentials in Git:

---

You can store login credentials in the cache so you don't have to type them in each time. Just use this command:

```
git config --global credential.helper cache
```

## How to initialize a Git repo:

---

Everything starts from here. The first step is to initialize a new Git repo locally in your project root. You can do so with the command below:

```
git init
```

## How to add a file to the staging area in Git:

---

The command below will add a file to the staging area. Just replace `filename_here` with the name of the file you want to add to the staging area.

```
git add filename_here
```

## How to add all files in the staging area in Git

---

If you want to add all files in your project to the staging area, you can use a wildcard `.` and every file will be added for you.

```
git add .
```

## How to add only certain files to the staging area in Git

---

With the asterisk in the command below, you can add all files starting with 'fil' in the staging area.

```
git add fil*
```

## How to check a repository's status in Git:

---

This command will show the status of the current repository including staged, unstaged, and untracked files.

```
git status
```

## How to commit changes in the editor in Git:

---

This command will open a text editor in the terminal where you can write a full commit message.

A commit message is made up of a short summary of changes, an empty line, and a full description of the changes after it.

```
git commit
```

## How to commit changes with a message in Git:

---

You can add a commit message without opening the editor. This command lets you only specify a short summary for your commit message.

```
git commit -m "your commit message here"
```

## How to commit changes (and skip the staging area) in Git:

---

You can add and commit tracked files with a single command by using the -a and -m options.

```
git commit -a -m "your commit message here"
```

## How to see your commit history in Git:

---

This command shows the commit history for the current repository:

```
git log
```

## How to see your commit history including changes in Git:

---

This command shows the commit's history including all files and their changes:

```
git log -p
```

## How to see a specific commit in Git:

---

This command shows a specific commit.

Replace commit-id with the id of the commit that you find in the commit log after the word commit.

```
git show commit-id
```

## How to see log stats in Git:

---

This command will cause the Git log to show some statistics about the changes in each commit, including line(s) changed and file names.

```
git log --stat
```

## How to see changes made before committing them using "diff" in Git:

---

You can pass a file as a parameter to only see changes on a specific file.

`git diff` shows only unstaged changes by default.

We can call diff with the `--staged` flag to see any staged changes.

```
git diff
git diff all_checks.py
git diff --staged
```

## How to see changes using "git add -p":

---

This command opens a prompt and asks if you want to stage changes or not, and includes other options.

```
git add -p
```

## How to remove tracked files from the current working tree in Git:

---

This command expects a commit message to explain why the file was deleted.

```
git rm filename
```

## How to rename files in Git:

---

This command stages the changes, then it expects a commit message.

```
git mv oldfile newfile
```

## How to ignore files in Git:

---

Create a `.gitignore` file and commit it.

## How to revert unstaged changes in Git:

---

```
git checkout filename
```

## How to revert staged changes in Git:

---

You can use the `-p` option flag to specify the changes you want to reset.

```
git reset HEAD filename
git reset HEAD -p
```

## How to amend the most recent commit in Git:

---

`git commit --amend` allows you to modify and add changes to the most recent commit.

```
git commit --amend
```

!!Note!!: fixing up a local commit with amend is great and you can push it to a shared repository after you've fixed it. But you should avoid amending commits that have already been made public.

## How to rollback the last commit in Git:

---

`git revert` will create a new commit that is the opposite of everything in the given commit. We can revert the latest commit by using the head alias like this:

```
git revert HEAD
```

## How to rollback an old commit in Git:

---

You can revert an old commit using its commit id. This opens the editor so you can add a commit message.

```
git revert comit_id_here
```

## How to create a new branch in Git:

---

By default, you have one branch, the main branch. With this command, you can create a new branch. Git won't switch to it automatically – you will need to do it manually with the next command.

```
git branch branch_name
```

## How to switch to a newly created branch in Git:

---

When you want to use a different or a newly created branch you can use this command:

```
git checkout branch_name
```

## How to list branches in Git:

---

You can view all created branches using the `git branch` command. It will show a list of all branches and mark the current branch with an asterisk and highlight it in green.

```
git branch
```

## How to create a branch in Git and switch to it immediately:

---

In a single command, you can create and switch to a new branch right away.

```
git checkout -b branch_name
```

## How to delete a branch in Git:

---

When you are done working with a branch and have merged it, you can delete it using the command below:

```
git branch -d branch_name
```

## How to merge two branches in Git:

---

To merge the history of the branch you are currently in with the `branch_name`, you will need to use the command below:

```
git merge branch_name
```

## How to show the commit log as a graph in Git:

---

We can use `--graph` to get the commit log to show as a graph. Also, `--oneline` will limit commit messages to a single line.

```
git log --graph --oneline
```

## How to show the commit log as a graph of all branches in Git:

---

Does the same as the command above, but for all branches.

```
git log --graph --oneline --all
```

## How to abort a conflicting merge in Git:

---

If you want to throw a merge away and start over, you can run the following command:

```
git merge --abort
```

## How to add a remote repository in Git

---

This command adds a remote repository to your local repository (just replace `https://repo_here` with your remote repo URL).

```
git add remote https://repo_here
```

## How to see remote URLs in Git:

---

You can see all remote repositories for your local repository with this command:

```
git remote -v
```

## How to get more info about a remote repo in Git:

---

Just replace `origin` with the name of the remote obtained by running the `git remote -v` command.

```
git remote show origin
```

## How to push changes to a remote repo in Git:

---

When all your work is ready to be saved on a remote repository, you can push all changes using the command below:

```
git push
```

## How to pull changes from a remote repo in Git:

---

If other team members are working on your repository, you can retrieve the latest changes made to the remote repository with the command below:

```
git pull
```

## How to check remote branches that Git is tracking:

---

This command shows the name of all remote branches that Git is tracking for the current repository:

```
git branch -r
```

## How to fetch remote repo changes in Git:

---

This command will download the changes from a remote repo but will not perform a merge on your local branch (as `git pull` does that instead).

```
git fetch
```

Commit after commit, Git builds up a log. You can find out the remote repository log by using this command:

```
git log origin/main
```

## How to merge a remote repo with your local repo in Git:

---

If the remote repository has changes you want to merge with your local, then this command will do that for you:

```
git merge origin/main
```

## How to get the contents of remote branches in Git without automatically merging:

---

This lets you update the remote without merging any content into the local branches. You can call `git merge` or `git checkout` to do the merge.

```
git remote update
```

## How to push a new branch to a remote repo in Git:

---

If you want to push a branch to a remote repository you can use the command below. Just remember to add `-u` to create the branch upstream:

```
git push -u origin branch_name
```

## How to remove a remote branch in Git:

---

If you no longer need a remote branch you can remove it using the command below:

```
git push --delete origin branch_name_here
```

## How to use Git rebase:

---

You can transfer completed work from one branch to another using `git rebase`.

```
git rebase branch_name_here
```

Git Rebase can get really messy if you don't do it properly. Before using this command I suggest that you re-read the official documentation [here](#)

## How to run rebase interactively in Git:

---

You can run `git rebase` interactively using the `-i` flag.

It will open the editor and present a set of commands you can use.

```
git rebase -i master
# p, pick = use commit
# r, reword = use commit, but edit the commit message
# e, edit = use commit, but stop for amending
# s, squash = use commit, but meld into previous commit
# f, fixup = like "squash", but discard this commit's log message
# x, exec = run command (the rest of the line) using shell
# d, drop = remove commit
```



## How to force a push request in Git:

---

This command will force a push request. This is usually fine for pull request branches because nobody else should have cloned them.

But this isn't something that you want to do with public repos.

```
git push -f
```

---