
CHEAT SHEETS GIT

# 50+ Git commands cheat sheet | PDF Download

2 months ago 13 min read

Table of Contents

## Why our Git commands cheat sheet?

Git commands cheat sheet PDF file that serves as a quick reference book for all Git commands with examples to work with Git. Git branches, repositories, changes, and more.

## What is a Distributed Version Control System?

A distributed version control system (DVCS) is a system that keeps track of the changes you make to files in your project.
This change history is stored locally on your computer and allows you to easily return to a previous version of your project if something goes wrong.
It is easy to move these changes to remote version control systems like GitHub or BitBucket.

# Git installation

You must first install Git on your computer before you can use it. It's usually a good idea to upgrade to the new version.

Git is available for Windows, macOS, and Linux OS. Get git for your operating system or select OS-specific installation here.

Linux installation: follow step by step git installation guide.

Mac OS: Download git on macOS

Windows: Download the guide of git on windows.

# Git configuration

## How to check your Git configuration using the command line?

This command returns all the config info including user name and email.

```
git config -l
```

You will get something like this

```
user.email=enail@gmail.com
user.name=username
http.schannelcheckrevoke=false
```

OR you will see the details like this

```
// Example output config

core.symlinks=false
core.autocrlf=true
```

```
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
help.format=html
http.sslcainfo=/ssl/certs/ca-bundle.crt
diff.astextplain.textconv=astextplain
rebase.autosquash=true
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
credential.helper=manager
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
user.email=enail@gmail.com
user.name=username
http.schannelcheckrevoke=false
```

## How to set up or change your Git username?

You can change or set a new username for git configuration. Use this command to set up your user name.

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

## How to set up your Git user email?

Like user name, you can change or set up a new email for your git configuration. Use the following command. Replace the dummy email with your email.

```
git config --global user.email "dummy_email@gmail.com"
```

### How do you cache your Git login credentials?

The caching of login credentials makes it easy to log in without typing your credentials each time while working with Git repositories.
The following command will cache your login information.

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

## Basic Git Commands, working with the repository.

The first step is to create a new Git repository in your project's root directory and initialize the git.

```
git init
```

## Add new files

GitHub recommends every repository include a README, LICENSE, and .gitignore. files in their repositories. You can add a README file with this command.

```
git add README.md
```

Git tracks all the changes you make. However, you need to add these files to commit. Git doesn't add files (newly created or modified files ) to the commit automatically.

You can check the current status of changes using this command.

```
git status
```

Git will show you something like this

```
$ git status
On branch master
Your branch is up to date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        abc.txt
nothing added to commit but untracked files present (use "git add" to tra
```

In the example above you can see the file named '**abc.txt**' is untracked. An untracked file is simply a file that does not exist in the Git index. To add this file the git provides a simple command.

```
git add fileNameHere
```

In our case ***" fileNameHere " is ' abc.txt '***.

If you have added/created many files then it is time-consuming to run the above command for each untracked file. This problem is solved by this command.

```
git add .
```

The dot or period in the above command stands for **"all"**. This command will add

all the untracked files in your project to the **staging area**.

## Commit changes

Once you made changes or added files you can commit them with a small commit message.

```
git commit -m "first commit"
```

The ***-m*** flag means 'message'. If you omit this flag then you will see an editor(normally vim) create the message. simply type the message in the editor and save it.

It is possible to add all the files and commit them at once. Using the **-am** option, you can **add** and **create** a commit message in one command.

In git, the **-a** flag is used to connect all of the files to your commit, after which you must run another command to write your commit message.

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

You can view the commit history using this command.

```
git log -p
```

```
#view commit history

git log -p
commit ab81c1e1b0146e03b448d08aeb8ca4f19c831dbf (HEAD -> master)
Author: niat786 <mail@gmail.com>
```

```
Date:   Sat Apr 10 15:17:26 2021 +0500
    your commit message here
diff --git a/abc.txt b/abc.txt
deleted file mode 100644
index e69de29..0000000
commit 6eac3d206e71470f38a35b5938f93306349241f3
Author: niat786 <niat786@gmail.com>
Date:   Sat Apr 10 15:11:16 2021 +0500
    ABC file added
diff --git a/abc.txt b/abc.txt
new file mode 100644
index 0000000..e69de29
```

## How do you work with branches in git?

There is only one branch called the main branch available in Git. However, you can create as many as you want. Multiple branches don't conflict with each other. Add one logical task to a single branch. for example a branch for bug fixes and another branch for a new feature in your project, etc. At last, you can merge them into one. we will go through step by step with branches in git.
Initially here is how you can create a new branch.

```
git branch newBranchName
```

Here is how you can switch to this newly created branch. Use **checkout** keyword.

```
git checkout newBranchName
```

If you want to list out all the branches then use this command.

```
git branch
```

```
## git branch, Example
```

```
## git commands list with examples

## this will print all the branches

git branch
  * master
  firstBranch
  secondBranch
```

**The * character** that precedes the **master branch** denotes the branch that you are actually working on.

## How to delete a branch in Git?

Sometimes you may need to delete a branch. After the merge with the main or master branch, other branches become irrelevant. To delete a branch simply run this command.

```
git branch -d branchNameHere
```

## How to replace a branch in Git?

There is a shortcut to creating, switching, and deleting the previous branch. simply use a shortcut to replace a branch in git. The following command will replace the current branch.

```
git branch -M main
```

# Working with the remote repository.

It is better to host code on remote version control systems. It is because you will get a fast, secure, and reliable platform to track changes anywhere. They are cross-platform compatible. Which makes it easy to move to the deployment of your project.

There are some major version control systems like **GitHub** and **Bitbucket** etc.

## How to add a remote repository in Git?

It is easy to push your code to the remote. You need a remote repository. It could be GitHub or any other version control system. we will assume GitHub here.
The following command adds a remote repository to your local repository.

```
git add remote nameOfTheRemote yourRepoURL
```

For example, we will add a repository like this. In our case name of the remote is the origin.

```
git remote add origin https://github.com/testaccount/testrepo.git
```

# Find out the remote repo URL

If you have cloned a project from GitHub then you can find the remote URL or repo URL with the help of this command.

```
git remote -v
```

The output will be something like this.

```
## Example in Git commands cheat sheet
## git commands list with examples
git remote -v
origin  https://github.com/test/test.git (fetch)
origin  https://github.com/test/test.git (push)
##
```

## Get more information about your remote.

```
git remote show origin
```

The origin is the name of our remote. You can use your own name of remote here. The above command will give you all the information about the remote.

The output will be similar to this.

```
## Examples in Git commands cheat sheet
## git commands list with examples

git remote show origin
* remote origin
  Fetch URL: https://github.com/test/test.git
  Push  URL: https://github.com/test/test.git
  HEAD branch: master
  Remote branch:
    master tracked
  Local branch configured for 'git pull':
    master merges with remote master
  Local ref configured for 'git push':
    master pushes to master (fast-forwardable)
```

### How do you push changes to a remote repository in Git?

When you've completed all of your work and are ready to save it to a remote repository, use the command below to push all of your changes:

```
git push
```

If you are pushing for the first time then it is better to mention the name of the remote and branch as well.

```
git push origin master
```

OR you can use this command

```
git push -u origin main
```

The -u flag adds an **upstream** or tracking reference to any branch that is up to date or successfully moved, which is used by **argument-less git-pull** and other commands.

As a result, after pushing your local branch with the -u option, it will be automatically connected to the remote branch, and you will be able to use git pull without any arguments.

```
It's the same as --set-upstream
```

## Working with existing Git repository.

What if you have an existing project on Git and you want to add or reduce some features?

There are some useful fundamental commands available to make your work simple and fast.

## How to copy remote repo to local machine?

Sometimes it needs to download a pre-built open source project from a version control system. For example, **spring, Laravel, Django**, and other frameworks are available on GitHub rather than creating them from scratch.
If you want to download a copy of a project already available on any version control system then use the following command.

```
git clone remoteRepoURL
```

Find out the remote URL with This command. This command is explained above.

```
git remote -v
```

## How to pull the repo from the remote?

If your team members are working on a repository, you can use the command below to get the most recent changes made to the remote repository.

```
git pull
```

### What is the difference between git pull and git clone?

Their working pattern looks similar but they are different.
**git clone:** This command is used to get a copy of a project from a remote. It downloads a full copy and  It's usually only used once for a given repository unless you want to have multiple copies of a project.
**git pull:** This command is used to update changes. It usually updates the local copy of a project. It downloads the changes from remote to local.

## Working with files

### What is git ignore?

It simply means ignoring specified files and folders. There is a .gitignore file that specifies which files or folders to ignore in a project.
Create a text file and call it .gitignore to create a local .gitignore file.
the .gitignore (should be the same as the dot ( . ) at the beginning of .gitignore the file name). Then make any necessary changes to this file.
Each new line should specify a new file or folder name that Git should ignore.

Use patterns or simple files and folder names to make an entry to `.gitignore` file. Remember the following patterns while making entries.

- * stands for a wildcard match.
- The / character is used to ignore the path in the `.gitignore` file.
- To add comments to a `.gitignore` use #.

Here is an example from the Laravel git ignore file.

```
### git commands list with examples/node_modules
/public/hot
/public/storage
/storage/*.key
/vendor
.env
.env.backup
.phpunit.result.cache
```

```
Homestead.json
Homestead.yaml
npm-debug.log
yarn-error.log
```

### How the .gitignore file is useful?

It is useful because we don't need to move all the dependencies and dev dependencies to the remote. It will be a waste of memory. for example, the **node_modules** may contain many packages. All the information about **node_modules** is stored in a **package.json** file. so we can regenerate it with a simple NPM (node package manager) command. ( `npm install` )

## Working with snapshots and the Git staging area

A snapshot is a representation of the state of something (for example, a folder) at a particular point in time. In simple words, a repository's snapshot is similar to a video's screenshot.

Between the working directory and the repository, the Git index serves as a staging area. It's used to put together a group of changes that you want to commit all at once.

The working directory, staging area, and repository are the three locations in Git where file changes can be made. Read more about this in detail here.

we will focus only on commands only.

```
git status
```

As we mentioned about this command. it is used to see modified files in your working directory. Another command in this category is also mentioned above. which is:

```
git add fileNameHere
```

## How reset command work in git?

It undoing the most recent commit while maintaining the changes in staging.

How to **unstage** a file while retaining the changes in the working directory? The reset command is useful for undoing the most recent commits.

```
git reset fileNameHere
```

For example here is a sample output. you can see the working mechanism of the reset command.

```
### git commands list with examples
step 1.
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        a.txt
........................
step 2.
$ git add a.txt
.......................
step 3.
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   a.txt
........................
```

```
step 4. (this will undo step 2)
$ git reset a.txt
.....................
step 5.
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        a.txt
```

## What is git diff and how does it work?

Git diff is a command used to view changes between the staged version of a file and the not staged version of the file. in simple words, it is a comparison of what has changed but has not been staged.
the syntax of the command is as follows.

```
git diff
```

The output will be similar to something like this. Just consider we are changing a file named **a.txt**

The plus sign(+) at the starting of the line means the current line is added.

the minus sign(-) at the starting means, current line deleted

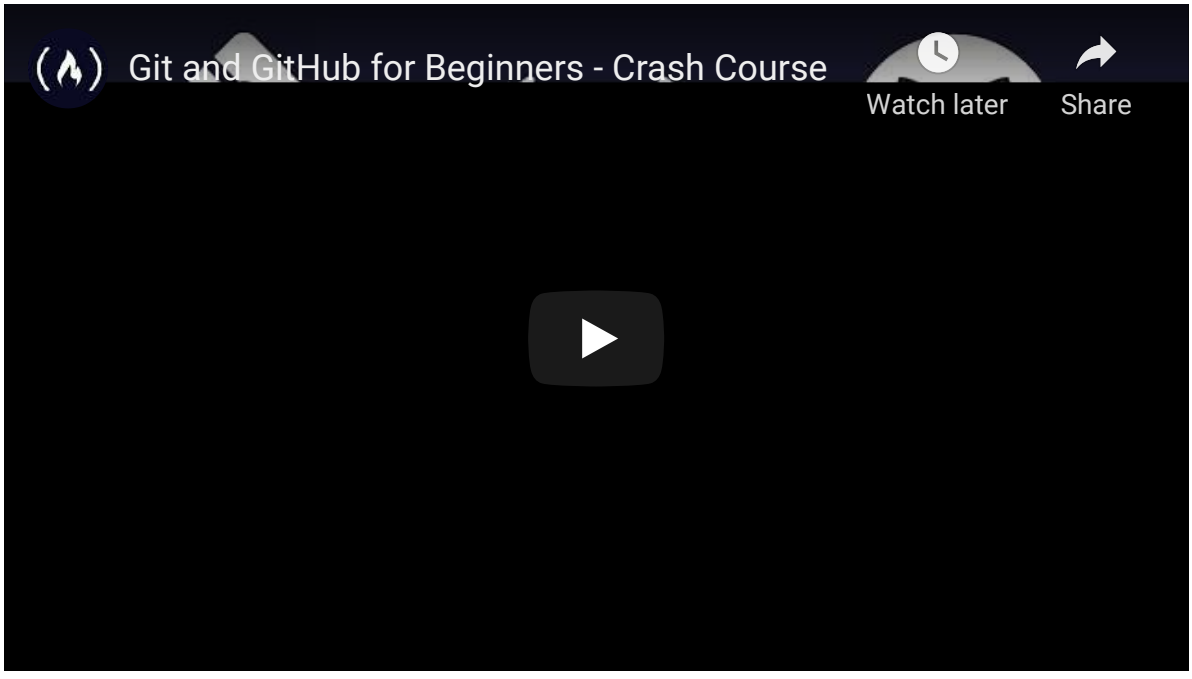
```
### git commands list with examples

$ git diff
diff --git a/a.txt b/a.txt
index 1deb9e0..0f51dd5 100644
--- a/a.txt
+++ b/a.txt
@@ -1 +1,2 @@
-new file A added
\ No newline at end of file
+new file A added
+a new line added to the file
```

```
\ No newline at end of file
```

The `--staged` flag is used to view the changes in the staged file that is not committed yet. for example, consider the same example for the file mentioned above.

```
### git commands list with examples
$ git diff --staged
diff --git a/a.txt b/a.txt
new file mode 100644
index 0000000..1deb9e0
--- /dev/null
+++ b/a.txt
@@ -0,0 +1 @@
+new file A added
\ No newline at end of file
```

## Git commands list with examples | Youtube Video



# Download Git commands cheat sheet PDF file

The given PDF file ( available for download ) and its content do not belong to us. Here you will get a simple two-page Git commands **cheat sheet**. The git commands list with examples may not be available as explained here. It covers all the basic commands necessary for your daily work. If you want to get a free copy of the cheat sheet then download it here.

DOWNLOAD PDF FILE

all git commands | Git | Git commands Cheatsheet | Git commands PDF download | Git commands with examples | GitHub Commands

## Posts

**Laravel 9 Blade Template cheat sheet | PDF Download**

4 min read

**Best PHP frameworks for web development**

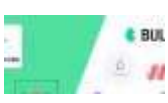
5 min read

**Create Glass Effect in CSS, Bootstrap, or Tailwind CSS**

2 min read

**Guide to CSS pseudo-elements with examples | cheatsheet download**

3 min read



**5 CSS Frameworks for Developers and**