Cheatography

# Git cheat sheet Cheat Sheet

by Ariel Gonzalez (ArielJGS) via cheatography.com/159091/cs/33552/

## Initial commands

| | |
|---|---|
| git --version | Check yout git version |
| git config --global user.name "Your name" | Configure your name as default |
| git config --global user.email "Your email" | Configure your email as default |
| git config --global core.editor "code --wait" | Configure a default editor, here vscode is being used as default |
| git config --global -e | Open and edit your configuration settings |
| git config --global core.autocrlf "input/-true" | How git should handle end of lines |

## Viewing staged and unstaged changes

| | |
|---|---|
| git diff --staged | Staging area changes that are going to the next commit |

## Viewing history

| | |
|---|---|
| git log | Show history of your repository |
| git log --oneline | Short history output |
| git log --oneline --reverse | Show history from the first commit |
| git log --oneline --stat | Show all the files that have been changed in each commit |
| git log --stat | More details about each commit |
| git log --oneline --patch | Full changes details in each commit |

## Viewing history (cont)

| | |
|---|---|
| git log --oneline --author="Name" | Commits from a given author |
| git log --oneline --after\|--before="Date \| Relative date" | Commits from a given date |
| git log --oneline --grep="Sometext" | Commits that contain the specified text, case sensitive |
| git log --oneline -S"Sometext" | Find all a commits that have added or removed the specified text |
| git log --oneline -S"Sometext" --patch | Mixed with patch to show full details about the commit |
| git log --oneline hash..hash | Filter commits by a given range of commits |
| git log --oneline --file.txt | Find commits that have modified a given file |
| git log --oneline --stat -- file.txt | Short output for changes over a given file |
| git log --oneline --patch -- file.txt | Full changes over a given file |
| git log --pretty=format:"Your format" | Customize the way you see output, then use alias for ease of use |
| git config --global alias.youralias "log --pretty=format:'Your format' | Alias example |

## Stashing

| | |
|---|---|
| git stash push -m "Your message" | Save your changes without committing them if you need to switch to a different branch. Stashing = Saving something in a safe place |
| git stash push --all -m "Your message" | Stashing new untracked files |
| git stash list | Show a list of stashed files |
| git stash show stash@{0} \| 0 | Show specific stashed file by its index |
| git stash apply 0 | Apply this stash to our working directory |
| git stash drop 1 | Remove a specific stash |
| git stash clear | Remove all stashes |

## Viewing changes across commits

| | |
|---|---|
| git diff HEAD~2 HEAD | Show differences between a range of commits |
| git diff HEAD~2 HEAD file.txt | Same as above but for a single file |
| git diff HEAD~2 HEAD --name-only | Show list of modified files in a given range of commits |
| git diff HEAD~2 HEAD --name-status | Show list of files and the type of change for each one |

By **Ariel Gonzalez** (ArielJGS)
cheatography.com/arieljgs/

Not published yet.
Last updated 7th August, 2022.

## About merging

| | |
|---|---|
| Fast-forward | Fast forward merge can be performed when there is a direct linear path from the source branch to the target branch. In fast-forward merge, git simply moves the source branch pointer to the target branch pointer without creating an extra merge commit. |
| Three way merge | Fast-forward merge is not possible if the branches have diverged. Then we need a 3-way merge which uses a dedicated commit to merge two histories or you can say branches. This new commit is based on three different commits, the common ancestor of our branches which includes the before code and the tips of our branches which contains the after code. |
| Fast-forward controversy | Cons: Pollutes the history, linear history is preferred for some people |
| | Pros: True reflection of history, easier to undo a feature |

## No fast forward merge

| | |
|---|---|
| git merge --no-ff bugfix/login-form | Merges the specified branch without using fast forward merge |
| git config --global ff no | Disable fast forward in every repository |

## Managing merges

| | |
|---|---|
| git merge bugfix/signup-form | Merges a branch into master |
| git log --oneline --all --graph | Shows a graph for easier understanding of merges |
| git branch --merged | View list of branches that have been merged into master, it's safe to delete these branches |
| git branch -d bugfix/signup-form | Delete a branch |
| git branch --no-merged | View list of branches that have not been merged into master |
| git merge --abort | Abort a merge if you run into a conflict that you're not ready to fix |

## Managing your first repository

| | |
|---|---|
| git init | Initialize your repository |
| git status | Get status of your current changes |
| git status -s | Short status information |
| git add "./file-1\|file2/*.txt" | Add files to the staging area for review |

## Commiting changes

| | |
|---|---|
| git commit -m "Your message" | Commit changes from your staging area and add message |
| git commit | Add a longer message for bigger or more detailed descriptions |
| git commit -am "Your message" | Skip the staging area and commit changes directly |
| Tips | Commits shouldn't be too big or too short, also use present or past tense verbs but stick to only one and be clear with your messages |

## Restoring files

| | |
|---|---|
| git restore --staged file.txt | Unstage or restore a file in the staging area taking the content from the latest commit |
| git restore --source=hash file.txt \| HEAD~1 file.txt | Restore a file to an earlier version |
| git clean -fd | Discard local changes for new or modified files and directories |

By **Ariel Gonzalez** (ArielJGS)
cheatography.com/arieljgs/

Not published yet.
Last updated 7th August, 2022.

## Restoring files (cont)

| | |
|---|---|
| Note | The restore command takes a copy from the next environment, for example, the working directory takes a copy from the staging area and the staging area takes it from the latest commit |
| git checkout hash file.txt | Restore file from a given commit hash |

## Viewing a commit

| | |
|---|---|
| git show "hash \| HEAD~1" | Show what was changed in a given commit |
| git show "hash:files/file1.txt \| HEAD~1:files/file1.txt" | Show the content of a file in a given commit |
| git ls-tree "hash \| HEAD~1" | Show all files in a given commit |
| git show HEAD~1 --name-only | Show files that have been modified in a given commit |
| git show HEAD~1 --name-status | Show files + status: added, deleted, modifiedetc... |

## Blaming

| | |
|---|---|
| git blame file.txt | Show who modified a given file |
| git blame -e file.txt | With email |
| git blame -e -L 1,3 file.txt | With a range of specific lines |

## Working with branches

| | |
|---|---|
| git branch | Show a list of existing branches |
| git branch name | Create a new branch with a given name |
| git switch name | Switch to a different branch |
| git switch -C name | Create and switch to a branch |
| git branch -m name bugfix/signup-form | Change the name of a branch |
| git diff master..bugfix/signup-form | See differences between branches |
| git diff bugfix/signup-form | If you're already in master there is no need to specify it |
| git branch -d bugfix/signup-form | Delete a branch after it has served its purpose |
| git branch -D bugfix/signup-form | Force deletion if you want to discard any changes made in this branch |

## Removing files

| | |
|---|---|
| git ls-files | Show current files in your staging area |
| git rm "./file1\|file2/*.txt" | Remove files from the current directory and staging area at the same time |

## Renaming or Moving files

| | |
|---|---|
| git mv oldname.txt newname.txt | Move or rename files in the working directory and staging area at the same time |

## Ignore files

| | |
|---|---|
| Create .gitignore | Include any files that you want to ignore here, examples: logs/ \| main.log \| *.log |
| Note | This only ignores files or directories if they have not been committed in the repository before |
| git rm --cached -r logs/ | Remove directory that was already committed by accident to start ignoring it with .gitignore |

## Checking out a commit

| | |
|---|---|
| git checkout "hash" | Go back in time and check a previous commit, this will show the state of every file as it was at that point in time |
| git log --oneline --all | You will need to add the parameter --all to show every commit when you're checking an old commit |
| git checkout master | Go back to your latest commit |

## Finding contributors using shortlog

| | |
|---|---|
| git shortlog | Show people that have contributed to the project |
| git shortlog -n | Sorted by number of commits per author |
| git shortlog -n -s | Suppress the commit messages |
| git shortlog -n -s -e | Show email address |

## Finding contributors using shortlog (cont)

| | |
|---|---|
| git shortlog -n -s -e --before="" --after="" | Show contributors for a given date |

## Tagging

| | |
|---|---|
| git tag | Show a list of existing tags |
| git tag -n | With their messages |
| git tag v1.0 | Create a tag for the current latest commit |
| git tag v1.0 hash | Create a tag for a specific commit |
| git checkout v1.0 | Then you can reference a commit by its tag |

## Tagging (cont)

| | |
|---|---|
| Note | Git supports two types of tags: lightweight and annotated. A lightweight tag is very much like a branch that doesn't change—it's just a pointer to a specific commit. Annotated tags, however, are stored as full objects in the Git database. They're checksummed; contain the tagger name, email, and date; have a tagging message; and can be signed and verified with GNU Privacy Guard (GPG). It's generally recommended that you create annotated tags so you can have all this information; but if you want a temporary tag or for some reason don't want to keep the other information, lightweight tags are available too. |
| git tag -a v1.1 -m "Your message" | Create an annotated tag and provide a message to it |
| git show v1.1 | Show commit by its tag |
| git tag -d v1.1 | Delete a tag |