



Configuration

\$ git config user.name <username>	Defines the username for the user of the <i>current repository</i> (local config) as a key-value pair.
\$ git config user.email <email>	Defines the email for the user of the <i>current repository</i> (local config) as a key-value pair.
\$ git config pull.rebase <false true>	Sets the default behavior of pull operations - to merge or rebase for the current repository.
\$ git config --global <any-command>	The `--global` flag changes the scope of the configuration from the local `config` file (current repository) to all repositories on your machine.

Basics

\$ git init	Initializes a Git repository in the current directory.
\$ git clone <url>	Clones a remote Git repository into the current directory, under a directory with the name of the remote one.
\$ git clone <URL> name	Clones a remote Git repository into the current directory, under a directory called `name`.
\$ git add <file_name>	Adds an updated file with the given filename, from the working tree to the staging area.
\$ git add <fileglob>	Adds the matching updated files from the working tree to the staging area.
\$ git add .	Adds all the updated files from the working tree to the staging area.
\$ git status	Prints status of modified files in your working tree - marking them as staged, unstaged and tracked respectively.
\$ git commit -m "<message>"	Creates a commit with the given message for the changes in the staging area.

Branching

\$ git branch <branch_name>	Creates a new branch, with the specified branch name. Cannot create two branches with the same name.
\$ git branch -m <branch_name> <new_name>	Renames branch.
\$ git branch -M <branch_name> <new_name>	Force rename branch. If duplicate names exist, the branch you're renaming will override the old one.
\$ git checkout <branch>	Checks out (sets working tree) to the specified branch.
\$ git checkout -b <branch>	Creates a new branch, and checks out to it. Convenience command that combines the two before this.
\$ git merge <branch>	Merges the specified branch into the one you're currently checked out on.
\$ git rebase <branch>	Reapplies the commits of the branch you're currently checked out on, on top of the branch you specify in the method.
\$ git log	Logs all commits in the currently checked out branch, with commit object info.
\$ git log -n	Logs the last `n` commits in the currently checked out branch, with commit object info.
\$ git log --oneline	Logs all commits in the currently checked out branch, with each commit in a single line containing only hash and message.
\$ git log --graph	Visualizes the logged commits in the CLI.



Remote Branches

\$ git fetch <remote> <branch>	Updates the relevant local `origin/branch` with the content of the given remote repository's branch.
\$ git fetch <remote> <remote_branch>:<local_branch>	Updates specified local `origin/branch` with the content of the specified remote repository's branch.
\$ git pull <remote> <branch>	Performs a fetch of the given branch from the given remote repository, then merges the result in the relevant branch of the local repository.
\$ git pull <remote> <remote_branch>:<local_branch>	Pull from the specified remote branch into the specified local branch.
\$ git pull -u <remote> <branch>	Sets up upstream between the local and remote branch.
\$ git pull	Pulls from the branch, and into the branch you've linked via the `-u` flag.
\$ git push <remote> <branch>	Sends the commits of the relevant local branch to the given branch of the given repository.
\$ git push <remote> <local_branch>:<remote_branch>	Explicitly set the branch you push from and to.
\$ git push -u <remote> <branch>	Same as regular pushing, but sets the remote branch as the upstream of the current one.
\$ git push	Same as the previous one, but uses upstream of the current branch.

Advanced

\$ git stash push	Saves the indexed changes into the stash (a dedicated local space for in-progress work).
\$ git stash	Saves unstaged changes in the `stash` for temporary storage.
\$ git stash pop	Reapplies previously saved change from the stash and removed it from it.
\$ git stash apply	Reapplies previously saved change from the stash, and keeps it in the stash.
\$ git stash apply pop stash@{0}	Applies/pops the given stash from the list of stashed changed. Latest is `0`.
\$ git stash list	Lists all saved stashes.
\$ git tag <tag>	Gives a label to `HEAD`.
\$ git tag -d <tag>	Deletes tag.
\$ git tag -a <tag>	Create Tag object, not just label.
\$ git push <remote> <tag>	Push tag to remote repository.
\$ git checkout <tag>	Checkout to given tag. Usually, in detached HEAD mode.
\$ git diff <file>	Check difference between staged and unstaged file states.
\$ git diff <commit1> <commit2>	Check difference between commits.
\$ git diff <branch1> <branch2>	Check difference between branches.
\$ git add -i	Interactive staging mode.



Reverting and Changing History

\$ git cherry-pick <reference>	Creates a copy of the given commit, under a brand new hash.
\$ git rebase -i <reference>	Reapplies the current branch's commits onto the given reference, but allows for specific actions on each commit.
\$ git reset <reference>	Resets a branch to the given commit, making changes uncommitted. Defaults to `reset --mixed`.
\$ git reset --hard <reference>	Resets a branch to given commit, discarding changes.
\$ git reset --soft <reference>	Resets a branch to given commit, keeping changes staged.
\$ git revert <reference>	Creates an anti-commit of the given commit, making a new commit in the history with the opposite changes.
\$ git revert --continue	Continue reverting after solving revert conflict.
\$ git revert --abort	Abort reverting.
\$ git commit --amend -m "new message"	Edit latest commit.

Common Errors

- Pushing before pulling while there are new changes on the remote repository.

Solution: Pull changes before pushing our own.

- Merging or rebasing while there are changes in the staging area.

Solution: Either commit or stash changes before merging, rebasing or pulling.

- Committing wrong files, or wrong message.

Solution: ``$ git commit -amend -m "new message"`, after `$ git add file` after adding relevant files.`

- Made a typo in the branch name.

Solution: ``$ git branch -m branch-name new-name``.

- Unstage files or directories from index.

Solution: ``$ git reset HEAD``.