CONFIGURE TOOLING

`git config --global user.name "[name]"`	Set the name you want attached to your commit transactions
`git config --global user.email "[email address]"`	Set the email you want attached to your commit transactions

CREATE REPOSITORIES

`git init`	Creates a new local repository with the specified name
`git clone [repo url]`	Downloads a project and its entire version history

MAKE CHANGES

`git status`	List changed files in your working directory
`git diff`	List changes to tracked files
`git diff --staged`	List file differences between staging and the last file version
`git add .`	Add *all* current changes to the next commit
`git add -p [file-name]`	Add *some* changes in [file-name] to the next commit
`git commit -m " [commit msg] "`	Commit local changes and add msg
`git commit -a`	Commit all local changes in tracked files
`git commit --amend`	Change the last commit*
`git reset <file>`	Unstage the file, but preserve its contents

* Don't amend published commits!

GROUP CHANGES

`git branch`	List all local branches in the current repository
`git branch [branch-name]`	Create new [branch] based on your current HEAD
`git branch -d [branch-name]`	Delete specified branch
`git checkout [branch-name]`	Switch HEAD branch
`git merge [branch-name]`	Merge [branch-name] into your current HEAD
`git tag [tag-name]`	Mark the current commit with a tag
`cat .git/HEAD`	See what HEAD points too

You can think of the HEAD as the "current branch". When you switch branches with `git checkout`, the HEAD revision changes to point to the tip of the new branch.

By **gsalem**
cheatography.com/gsalem/

Not published yet.
Last updated 15th January, 2018.
Page 1 of 3.

SYNCHRONIZE

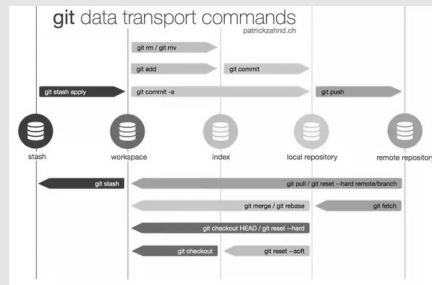|	
`git remote show [remote-name]`	Show information about a remote
`git remote -v`	List all currently configured remotes
`git branch -dr [remote]/[branch]`	Delete a branch on the remote
`git fetch [remote]`	Download all changes from [remote] but do'nt integrate into HEAD
`git pull [remote] [branch]`	Download changes and directly merge/integrates into HEAD
`git push [remote] [branch]`	Publish local changes on a remote
`git push --tags`	Publish your tags
`git merge [remote]/[branch]`	Combine the remote [branch] into your current HEAD
`git pull`	Download bookmark history and incorporates changes
`git add <resolved-file>`	Use your editor to manually solve conflicts and (after resolving) mark file as resolved
`git rm <resolved-file>`	Use your editor to manually solve conflicts and (after resolving) mark file as resolved

COMMIT HISTORY

`git log`	Show all commits, starting with newest
`git log -p [file-name]`	Show changes over time for a specific file
`git blame [file-name]`	Who changed what and when in [file-name]
`git show [commit-ID]`	Output metadata and content changes a specific commit

UNDO

`git reset --hard HEAD`	Discard all local changes in your working directory
`git checkout HEAD [file]`	Discard local changes in a specific file
`git revert [commit]`	Revert a commit (by producing a new commit with contrary changes)
`git reset --hard [commit]`	Discard all history and changes back to the specified commit
`git reset [commit]`	Revert a commit (by producing a new commit with contrary changes)*and preserve all changes as unstaged changes*
`git reset --keep [commit]`	Revert a commit (by producing a new commit with contrary changes)*and preserve uncommitted local changes*

By **gsalem**
cheatography.com/gsalem/

Not published yet.
Last updated 15th January, 2018.
Page 2 of 3.

WORKFLOW

workspace ⟷ working directory

index ⟷ staging area

local repository ⟷ HEAD

SOURCES

GitHub Git Cheat Sheet

Git Tower

Making Sense of Git – A Visual Perspective