Note: *This is not a comprehensive reference by any stretch of the imagination!*

Before we start

```
git config --global user.name "My User Name"
git config --global user.email user@example.com
```

Set up your username and email to be used in commits. (Leave out the `--global` if you just want to set them for a particular repo.)

Create a repo

`git init reponame`	Make a new repo
`git init --bare reponame`	Make a new bare repo
`git init .`	Make a new repo in the current directory

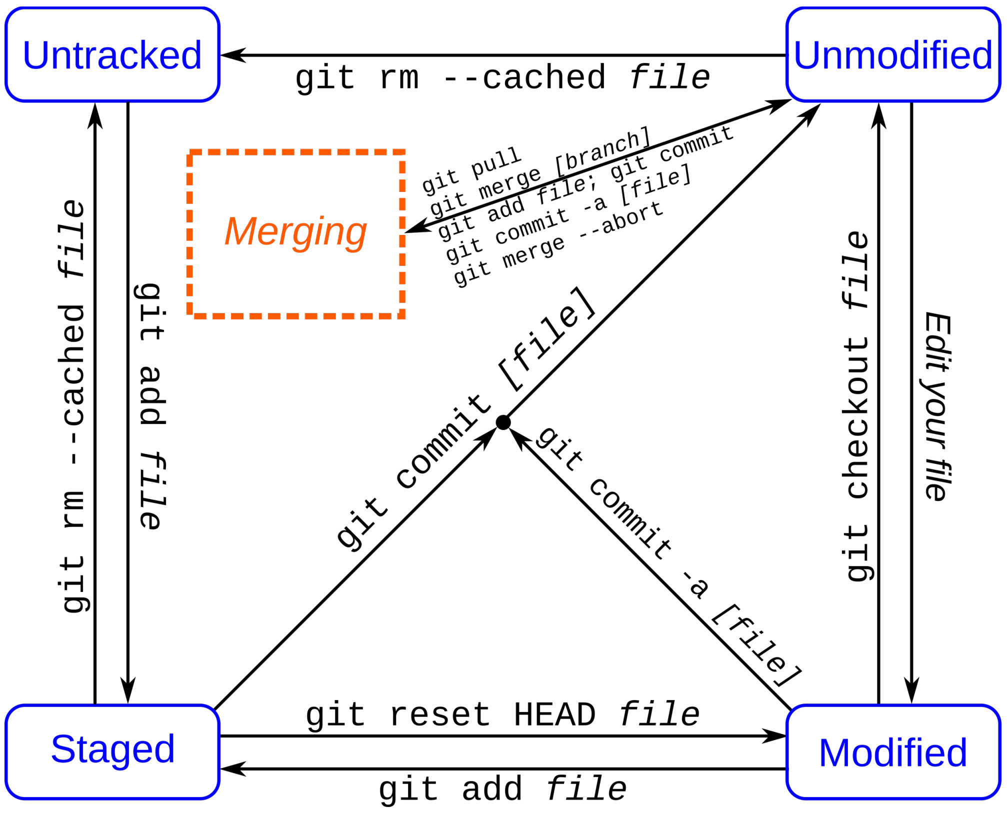
Destroy a repo

Remove the `.git` subdirectory to return the tree to a non-repo.

Or just remove the whole directory tree.

File States

`git add filename`	Move file from untracked or modified to staged
`git commit filename`	Move file from staged to unmodified, making a commit
`git commit -a filename`	Shortcut: same as `git add` followed by `git commit`
`git checkout filename`	Restore a modified file to its last unmodified state
`git rm --cached filename`	Stop tracking a file
`git reset HEAD filename`	Move a file from staged to modified
`git status`	Show file and repo states



File states

Untracked	Git knows nothing about this file and is ignoring it
Unmodified	The file is in the repo, unchanged from the last commit
Modified	The file is in the repo, changed from the last commit
Staged	The modified file has been copied to staged and is now ready to commit

Diff

`git diff`	Show all differences between your working tree and the last commit on this branch
`git diff` *`filename`*	Show the difference between a modified file and its unmodified state
`git diff` *`branchname`*	Show the differences between this branch and another

`git difftool`	Same as git diff, but launches an external diff tool
`git config --global diff.tool` *difftool*	Configure git to use a specific diff tool

Branches

`git branch`	Show local branches
`git branch` *branchname*	Create a new branch at `HEAD`
`git checkout` *branchname*	Checkout a branch
`git checkout -b` *branchname*	Create and checkout a branch
`git branch -d` *branchname*	Delete a fully-merged branch
`git branch -D` *branchname*	Delete a branch unconditionally
`git branch -r`	Show remote-tracking branches

Merging

`git merge` *branchname*	Merge a branch into this branch
`git merge --abort`	Reset and pretend this merge never started
`git merge --squash` *branchname*	Merge, squashing all *branchname* commits into one
`git add` *filename*	Move a file to staged during a merge
`git commit`	Make a commit and finish the merge
`git commit -a` *[filename]*	Auto stage then commit, finishing the merge
`git pull`	Might trigger a merge; see below.

Checking out Commits

`git checkout` *hash*	Checkout a specific hash (specify at least the first 4 digits)
`git checkout` *branchname*	Checkout a branch
`git checkout` *tagname*	Checkout a tag
`git checkout HEAD^`	Checkout the commit before head
`git checkout HEAD^^`	Checkout the commit before the commit before head, etc.

`git checkout HEAD~2`	Checkout the commit 2 commits ago (same as `HEAD^^`)
`git checkout HEAD~3`	Checkout the commit 3 commits ago (same as `HEAD^^^`)
`git checkout` *`branch file`*	Checkout a *file* from a *branch* (overwrites your file)
`git checkout -p` *`branch file`*	Interactively patch *file* from another *branch*

Remotes

`git clone` *`remoteURI local`*	Clone a remote repo to a local repo
`git push`	Push all local commits to the remote
`git pull`	Pull all remote commits to the local. Shorthand for `git fetch` followed by `git merge`.
`git push -u` *`remotename localbranch`*	Create a remote-tracking branch
`git push` *`remotename`* `--delete` *`branchname`*	Delete a remote-tracking branch from the remote
`git fetch`	Retrieve commits, branches, etc. from remote. Rare.
`git remote -v`	Show the remotes information for the repo
`git remote add` *`remotename URI`*	Add a new remote

Misc

`man gitglossary`	Bring up a glossary (Unix/Linux/MacOS)
`man git` *`topic`*	Bring up `man` page for topic, e.g. `man git commit`
`git archive --format=zip -o` *`filename`*`.zip HEAD`	Create a ZIP archive of the files at `HEAD`
`gitk`	Tcl/Tk-based front-end to viewing commit history
`git status`	Show file and repo states

Glossary

Bare repo	Contains repo info, but no directly accessible files

Branch	A named reference to a commit that can move with `HEAD`. Branches refer to single commits, not to many commits.
Checkout	Bring the working tree up to date with a particular commit
Commit	A snapshot of the state of your project
Commit-ish	Anything that refers to a particular commit
Detached head	`HEAD` points to a commit with no branch or tag
Fast-forward merge	Easy merge when one branch is a direct ancestor of another; the branch labels are merely brought up to the same commit. No new commits are created.
Fork	[GitHub] To make a clone of a repo that is owned by a different GitHub user
GitHub	A web-based front-end to git that enables easier collaboration between users and projects. Git is not GitHub. GitHub uses git.
GitLab	A competitor to GitHub
Hash	A unique numeric reference to a commit (or other object)
`HEAD`	Another name for the "current branch"
`main`	Name of the main branch
`master`	Deprecated name of the main branch
Merge	Bring the contents of one branch into this branch. After all conflicts are resolved, a new commit is created and both branches point to this new commit.
`origin`	The default name of the upstream repo (that you cloned from)
Parent	The commit(s) before another
Pull request	[GitHub] A request to another user to merge changes from your fork
Ref	A reference to a commit, e.g. HEAD,
Remote-tracking branch	Branch on a remote that's kept in sync with a local branch
Tag	A named commit; like a branch, except fixed

Topic Branch	A local branch used for adding a specific feature
`upstream`	[GitHub] The name of the remote this one was forked from
Working tree	Locally checked out files including modifications

Resources

Git website	https://git-scm.com/
Git `man` pages (reference)	https://git-scm.com/docs
Pro Git book	https://git-scm.com/book/en/v2
Git tutorial	https://git-scm.com/docs/gittutorial
Git usage examples	https://git-scm.com/docs/giteveryday
Example Git workflows	https://git-scm.com/docs/gitworkflows
Glossary	https://git-scm.com/docs/gitglossary
How to Undo (almost) Anything in git	https://github.com/blog/2019-how-to-undo-almost-anything-with-git

Things to learn next

`git rebase`	Collapse a number of commits into a single commit
`git stash`	Temporarily save your working tree off to the side
`git cherry-pick`	Bring individual commits from one branch into another
`git reflog`	Show or modify references history
`git reset`	Reset the `HEAD` to a specified state
`git bisect`	Quickly determine which commit broke something
`git blame`	Determine who made a particular change in a file
`.gitignore`	A file containing a list of files git should completely ignore
`.gitconfig` `.git/config`	Global and repo-specific configuration files
`tig`	A third-party text-based tool for examining the repo