Git quick reference for beginners

Kevin Markham
http://www.dataschool.io/

The guide below is organized by task, with an emphasis on basic tasks and common command line arguments. It begins with the workflow for cloning, updating, and syncing with a remote repo because that's a common way to get started with Git and GitHub.

Note that this is only a reference guide, and will not teach you Git. It does not explain the difference between staged and committed, what to do with a .gitignore file, or when to create a branch. But if you are already familiar with those concepts, this guide will hopefully refresh your memory and help you to discover other commands you might need.

Cloning a remote repo (that you created or forked on GitHub)

- git clone ⟨your-repo-url⟩

  copies your remote repo to your local machine (in a subdirectory with the repo's name), and automatically creates an "origin" handle
- git remote add upstream ⟨forked-repo-url⟩

  adds an "upstream" handle for the repo you forked
- git remote -v

  shows the handles for your remotes
- git remote show ⟨handlename⟩

  inspect a remote in detail

Tracking, committing, and pushing your changes

- git add ⟨name⟩

if untracked, start tracking a file or directory; if tracked and modified, stage it for committing

- git reset HEAD ⟨name⟩

  unstage a changed file

- git commit -m "message"

  commits everything that has been staged with a message

  - -a -m "message"

    automatically stages any modified files, then commits

  - --amend -m "new message"

    fixes the message from the last commit

- git push origin master

  pushes your commits to the master branch of the origin

Syncing your local repo with the upstream repo

- git fetch upstream

fetch the upstream and store its master branch in "upstream/master"

- git merge upstream/master

  merge that branch into the working branch

Viewing the status of your files

- git status

  check which files have been modified and/or staged since the last commit

- git diff

  shows the diff for files that are modified but not staged

  - --staged

    shows the diff for files that are staged but not committed

Viewing the commit history

- git log: shows the detailed commit history

  - -1

    only shows the last 1 commit

  * -p

    shows the line diff for each commit

  * -p --word-diff

    shows the word diff for each commit

  * --stat

    shows stats instead of diff details

  * --name-status

    shows a simpler version of stat

  * --oneline

    just shows commit comments

- gitk

  open a visual commit browser

Managing branches

- git branch

  shows a list of local branches

  * ⟨branchname⟩

    create a new branch with that name

  * -d ⟨branchname⟩

    delete a branch

  * -v

    show the last commit on each local branch

  * -a

    show local and remote branches

  * -va

    show the last commit on each local and remote branch

  * --merged

    list which branches are already merged into the working branch (safe to delete)

  * --no-merged

    list which branches are not merged into the working branch

- git checkout ⟨branchname⟩

  switch the HEAD pointer to a different branch

  * -b ⟨branchname⟩

    create a new branch and switch to it

Removing, deleting, and reverting files

- git rm ⟨name⟩

  deletes that file from the disk, then stages its deletion

  - --cached ⟨name⟩

    stops tracking a file, then stages its deletion (but does not delete it from the disk)

- git mv ⟨oldname⟩ ⟨newname⟩

  renames the file on disk, then stages the deletion of the old name and addition of the new name

- git checkout -- ⟨name⟩

  revert a modified file on disk back to the last committed version

Other basic commands

- git init

  initialize Git in an existing directory

- git config --list

  shows your Git configuration

- touch .gitignore

  create an empty .gitignore file