Git Cheat Sheet

© Erik E. Lorenz, March 27, 2014

Help

`git help`	show commonly used commands
`git help -a`	show all commands
`git` *command* `--help`	show help for a specific command

Internal Files and Directories

`~/.gitconfig`	global git configuration. See `git config --global`
`./.git/`	internal git data
`./.git/config`	local git configuration
`.gitignore`	list of patterns and relative paths to ignore in *repo*

Terms

term	description	examples
repo	project dir. Contains `.git/`	~/code/project
commit	set of changes relative to previous commit	c03aa34b...
branch	reference to a commit. Automatically updated by `git commit`	master devel remotes/origin/master
file	regular file within *repo*	README src/main.cpp
dir	regular subdirectory of *repo*	src/
tag	fixed reference to a commit	v1.0
remote	alias for a remote url	origin github
url	address of a remote git repository. Can be read-only (http) or read-write	http://github.com/usr/proj gitolite@sim:project /mnt/tmp/project.git

`HEAD`	reference to current *commit*
`HEAD^`	reference to last *commit*
unstaged file	a file which hasn't been added to the next commit
staged file	a file which has been added to the next commit
tracked file	a file which has been previously committed
untracked file	a file which hasn't been tracked during the last commit
new file	a previously untracked file
ignored file	a file which won't be tracked or staged. see `.gitignore`

Cheat Sheet Color Coding

`cmd`	Usually not harmful. Changes local references only.
`cmd`	Rewrites history, permanently deletes data or annoys others

Global Setup

`git config --global user.name "`*Your Name*`"`	user settings
`git config --global user.email "`*your@mail.com*`"`	
`git config --global color.ui true`	terminal colors

Bonus: add the `git graph` alias:

```
git config --global alias.graph "log --graph --all
--format=format:'%C(bold blue)%h%C(reset) - %C(bold
green)(%ar)%C(reset) %C(white)%s%C(reset) %C(bold white)--
%an%C(reset)%C(bold yellow)%d%C(reset)' --abbrev-commit
--date=relative"
```

Repository Initialization

`git init` *dir*	create *dir* as a git repository
`git init`	convert the current directory to a git repository
`git clone` *url*	clone a remote repository into a new directory
`--bare`	without working copy
`--depth` *num*	shallow clone: omits older commits

Workflow Examples

`cd` *dir*
`git init`
`git add .`
`git commit -m 'initial commit'`
	creation of a new repository

`git checkout -b feature-`*asd*
(file editing)
`git add .`
`git commit -m 'feature` *asd* `implemented'`
	usual workflow when writing new features or fixing bugs

`git checkout master`
`git merge --no-ff feature-`*asd*
`git branch -d feature-`*asd*
`git push origin master`
	updating the master branch after a feature has been successfully implemented.

Managing Local Changes

`git status`	show staged, unstaged and new files
`git status` *dir*|*file*	show status of a subset of files
`git diff`	show all changed lines since last commit
`git diff` *file*	show `diff` for *file* only
`git diff` *file commit*	show changes since *commit*
`git add` *dir*|*file*	stage *dir* or *file* for the next commit
`git add -u`	stage all tracked files
`git add --all`	stage all existing files, including new files
`git add -A`	alias for `git add --all`
`git stage`	alias for `git add`
`git rm` *file*	delete *file* and stage its deletion
`git mv` *file... target*	move and stage one or more files
`git commit`	bundle staged changes int a new *commit*
`-m '`*commit message*`'`	pass the commit message in-line
`--amend`	add changes to last commit

Managing Local Branches

`git branch`	show current branch
`git branch -a`	show all branches
`git checkout` *branch*	reset files to *branch*
`git branch` *branch*	create a new branch at the current commit
`git checkout -b` *branch*	create a new branch and check it out
`git merge` *branch*	merge changes from another branch
`--no-ff`	force a new commit (cleaner merge history)
`--ff-only`	don't merge if both branches were changed
`git branch -d` *branch*	delete a merged local branch
`git branch -D` *branch*	force deletion of a local branch

Stashes

`git stash save`	save all changes in a stash and reset to last commit
`git stash list`	list all stashes
`git stash show`	show which files were changed
`git stash pop`	remove and apply stash (e.g. to a different branch)

Remote Repositories

`git clone` *url*	creates the remote `origin` during `init`
`git remote -v`	show all remotes and addresses
`git remote add` *remote url*	create a new remote
`git remote rm` *remote*	delete a remote
`git fetch` *remote*	fetch all branches without merging
`git fetch --all`	fetch from all remotes without merging
`git pull` *remote branch*	`fetch` and `merge` a remote branch
`--ff-only`	Only merge if there's no file conflict
`git push` *remote branch*	upload local commits to *remote*
`git push` *remote* `:`*branch*	delete *branch* on *remote*
`git remote prune` *remote*	delete invalid refs to remote branches

Upstreams

`git push -u` *remote branch*	set upstream and push to *remote branch*
`git pull`	pull current branch from upstream
`git push`	push current branch to upstream
`git push --all`	push all branches to their upstream
`git branch --set-upstream-to=`*remote/branch*	
	set upstream of the current branch

Managing Tags

`git tag -a` *tag commit*|*branch*	create a tag at *commit* or *branch*
`-m '`*tag message*`'`	pass the tag message in-line
`git tag -d` *tag*	delete *tag* locally
`git push --tags`	push all tags
`git push` *remote* `:refs/tags/`*tag*	delete a remote tag

Searching for Files

`git grep '`*pattern*`'`	search all tracked files for *pattern*
`git ls-files`	list tracked files
`-d`	list deleted files
`-m`	list modified files
`-u`	on merge conflict, list unmerged files
`--other --exclude-standard`	list untracked files
`git clean -ndx`	list untracked and ignored files
`git clean -ndX`	list ignored files

Working Directory Cleanup

`git clean -f`	delete untracked files
`git clean -fd`	delete untracked files and directories
`git clean -fdX`	delete ignored files only
`git clean -fdx`	delete untracked and ignored files
`git archive --prefix=`*project/* `-o` *project*`.tar.gz` *branch*	
	create a tar.gz archive of all tracked files

Commit History

`git log`	show local commit history
`git graph`	pseudo-graphical commit history. See **Setup**
`git ls-tree -r HEAD`	show modification commit for each file
`git show` *commit*	`diff` of the given *commit* to its parent
`git blame` *file*	show the modification commit for each line

Undo Bad Things ™

`git reset HEAD` *file*	undo `git add` of a single file
`git reset --hard`	undo everything since last commit
`git checkout --` *file*	undo *file* changes since last commit
`git rebase`	undo `git branch` by moving com-
`--onto` *branch commit*	mits on top of the old branch
`git push` *remote commit*`:`*branch*	revert remote branch to *commit*
`rm -rfv` *repo*`/.git*`	undo `git init`

Accidentally committed a password file? Delete any copy or reference of it on the current branch:

```
git filter-branch --force --index-filter 'git rm --cached
--ignore-unmatch file ' --prune-empty --tag-name-filter cat -- --all
```

Further Reading

A successful Git branching model
http://nvie.com/posts/a-successful-git-branching-model/
AlBlue's Git Tip of the Week
http://alblue.bandlem.com/Tag/gtotw/
git Documentation
http://git-scm.com/doc