

Git Cheat Sheet

© Erik E. Lorenz, April 9, 2014

Help

git help	show commonly used commands
git help -a	show all commands
git <i>command</i> --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 <i>repo</i>

Terms

term	description	examples
<i>repo</i>	project dir. Contains .git/	~/code/project
<i>commit</i>	set of changes relative to its parent commit(s)	c03aa34b...
<i>branch</i>	series of commits. References the latest commit. Updated on git commit	master devel remotes/origin/master
<i>file</i>	regular file within <i>repo</i>	README src/main.cpp
<i>dir</i>	regular subdirectory of <i>repo</i>	src/
<i>tag</i>	fixed reference to a commit	v1.0
<i>remote</i>	alias for a remote url	origin github
<i>url</i>	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 <i>commit</i>
HEAD~	reference to last <i>commit</i>
index	where changes are stored prior to a 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	Most frequent commands
cmd	Usually not harmful: Changes local references only
cmd	Rewrites history, permanently deletes data or annoys others

Global Setup

git config --global user.name " <i>Your Name</i> "	user settings
git config --global user.email " <i>your@mail.com</i> "	
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 <i>dir</i>	create <i>dir</i> as a git repository
git init	convert the current directory to a git repository
git clone <i>url</i>	clone a remote repository into a new directory
--bare	without working copy
--depth <i>num</i>	shallow clone: omits older commits

Workflow Examples

cd <i>dir</i> git init git add . git commit -m 'initial commit'	setup of a new repository
(file editing) git add <i>file</i> git commit -m 'feature <i>asd</i> implemented'	usual workflow
git checkout master git merge --no-ff feature- <i>asd</i> git branch -d feature- <i>asd</i> git push origin master	update of the master branch with changes from a feature branch before deleting it

Managing Local Changes

git status	show changed files since last commit
git status <i>dir file</i>	show status of a subset of files
git diff	show all changed lines since last commit
git diff <i>file</i>	show diff for <i>file</i> only
git diff <i>file commit</i>	show changes since <i>commit</i>
git add <i>dir file</i>	stage <i>dir</i> or <i>file</i> 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 add -i	interactive staging
git stage	alias for git add
git rm <i>file</i>	delete <i>file</i> and stage its deletion
git mv <i>file... target</i>	move and stage one or more files
git commit	bundle staged changes into a new <i>commit</i>
-m ' <i>commit message</i> '	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 <i>branch</i>	move to <i>branch</i>
git branch <i>branch</i>	create a new branch at the current commit
git checkout -b <i>branch</i>	create a new branch and move to it
git merge <i>branch</i>	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 mergetool	starts a program to help with merge conflicts
git branch -d <i>branch</i>	delete a merged local branch
git branch -D <i>branch</i>	force deletion of a local branch

Stashes

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

Remote Repositories

git clone <i>url</i>	automatically creates the remote origin
git remote -v	show all remotes and addresses
git remote add <i>remote url</i>	add a new remote
git remote rm <i>remote</i>	remove a remote
git fetch <i>remote</i>	download all <i>remote</i> branches w/o merge
git fetch --all	fetch from all remotes
git pull <i>remote branch</i>	fetch and merge a remote branch
--ff-only	Only merge if there's no file conflict
git push <i>remote branch</i>	upload local commits to <i>remote</i>
git push <i>remote :branch</i>	delete <i>branch</i> on <i>remote</i>
git remote prune <i>remote</i>	delete invalid refs to remote branches

Upstreams

git push -u <i>remote branch</i>	set upstream and push to <i>remote branch</i>
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= <i>remote/branch</i>	set upstream of the current branch

Managing Tags

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

Searching for Files

git grep ' <i>pattern</i> '	search all tracked files for <i>pattern</i>
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= <i>project/</i> -o <i>project.tar.gz branch</i>	create a tar.gz archive of all tracked files

Commit History

git log	show local commit history
--oneline	one line per commit
--decorate	show tags and branch names
--merges	show merge commits only
git graph	graphical commit history. See Global Setup
git ls-tree -r HEAD	show modification commit for each file
git show <i>commit</i>	diff of the given <i>commit</i> to its parent
git blame <i>file</i>	show the modification commit for each line

Undo Bad Things TM

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

Delete any copy or reference of a file on the current branch:

git filter-branch --force --index-filter 'git rm --cached --ignore-unmatch <i>file</i> ' --prune-empty --tag-name-filter cat -- --all

Further Reading

Git Documentation

<http://git-scm.com/doc>

A successful Git branching model

<http://nvie.com/posts/a-successful-git-branching-model/>

AIBlue's Git Tip of the Week

<http://alblue.bandlem.com/Tag/gtotw/>