Cheatography

Git commands Cheat Sheet

by itsellej via cheatography.com/55812/cs/14830/

SETTING UP A REPOSITORY

Git init

git init

Creates a new repository in a directory

Git clone

git clone [url] [new directory name]

Clone a repo into a new directory

git clone [url]

Clone a repo into the current directory

SAVING CHANGES

Git add

git add [file name]

Add files to staging area

git add .

Add all changed files to staging area

git add '*[file type]'

Example "git add *.txt" to add only text files to the staging area

git add [directory]

Stages changes of files in a directory

https://www.atlassian.com/git/tutorials/saving-changes#git-add

Git reset

git reset HEAD [file name]

Resets file in working directory to be the same as the HEAD (last) commit

git reset [commit ID]

Resets files in working directory to be the same as the commit specified

Git commit

git commit

Opens atom, so you can add a commit message on top line. Remember to save

git commit -m ["commit message"]

Add commit message using the command line

git commit -a -m ["commit message"]

Commits changed tracked files

* Style guide for writing commit messages: http://udacity.github.io/git-styleguide/
Keep commits small. Make one commit per logical change.
Messages written in present tense.

https://www.atlassian.com/git/tutorials/saving-changes#git-commit

Git diff

git diff

Display changes to files in working directory (not staged)

git diff --staged

Display changes to staged files

**git diff [commit id 1] [commit id 2]

Compare two commits

git diff HEAD

Display changes between staged and unstaged file changes

Compare changes between files

UNDOING CHANGES

git clean

git clean -n

Dry run. Does not delete files, but shows which files would be deleted

git clean -f

Initiates the actual deletion of untracked files

git clean -d

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

git clean (cont)

Remove any untracked directories. Use in combination with previous commands above

- Command works on **untracked files** (not added to staging area yet)
- Hard filesystem deletion
- Works on files, not directories

https://www.atlassian.com/git/tutorials/undoing-changes/git-clean

git revert

git commit HEAD

Reverses most recent commit

git commit [commit ID]

Reverses changes made associated with a specific commit ID

git commit [commit ID] --no-edit

Will not open the editor. Default command will open editor

- Inverts changes made from the previous commit
- History of commits is not lost
- Good for shared repos

https://www.atlassian.com/git/tutorials/undoing-changes/git-revert

REWRITING HISTORY

git commit --amend

git commit --amend m [new commit message]*

Edit the commit message on last commit

git commit --amend --no-edit

Adding forgotten staged files to recent commit with no commit message

git commit --amend

Take most recent commit and add new staged changes to it

- Run when nothing is staged*
- Amended commits are new commits. Previous commit will no longer be available
- Don't use on public commits which other devs have based their work on

https://www.atlassian.com/git/tutorials/rewriting-history

COLLABORATING AND SYNCING - GITHUB

Git remote

git remote

Check if you have any remote repositories. *Exception* - if you have cloned a repo, command will return original repo as a remote repo

git remote -v

Displays the full path to the remote repo

git remote add origin [github url]

Add a remote repo. Origin = name of remote repo. Can add alternative name instead of origin

git remote [url] [branch name]

Point remote branch to correct url

git remote rm [remote repo name]

Remove connection to remote repo specified

git remote rename [remote repo name] [new name]

Rename a remote repo

When you have multiple branches, you can:
- **merge all branches** into your local repo, and push to remote repo, or;
- **push individual branches** from local to remote repo

https://www.atlassian.com/git/tutorials/syncing#git-remote

Git fetch

git fetch [remote repo name]

Retrieve all branches from remote repo

git fetch [remote repo name] [branch]

Retrieve all commits on remote's (origin) master branch*. Use when both local and remote have changes the other does not have

git fetch --dry-run

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Git fetch (cont)

See changes to the remote repo before pulling into local repo

- Use to see what everybody else has been working on
- Fetched content is represented as a remote branch. Does not affect local repo
- Follow with git merge origin/master to merge remote repo changes to local repo
- Then push new merge commit back to the remote repo
- git push origin master

https://www.atlassian.com/git/tutorials/syncing#git-fetch

Git pull

git pull [remote repo]

Pull changes from remote repo to your local repo. Fast forward merge. Alternative is **git fetch**

git pull [remote repo]/[branch name]

Pull changes from remote repo branch to your local repo

git pull --rebase [remote repo]*

Pull and merge remote into local

- To be used if remote repo may have changes in the form of merged commits
- Git pull command = git fetch and git merge
- using rebase ensures a linear history by preventing unnecessary merge commits
- can use following command to ensure git pull uses rebase automatically, instead of merge:
git config --global branch.autosetuprebase always

https://www.atlassian.com/git/tutorials/syncing#git-pull

git push

git push [remote repo] [branch name]

Push commits from local repo to remote repo. *Example: git push origin master*

git push [remote repo] --all

Push commits from all local branches to remote repo

git push [remote repo] --tags*

Sends all of your local tags to the remote repository

- Tags are not automatically pushed with other git push commands

https://www.atlassian.com/git/tutorials/syncing#git-push

INSPECTING A REPOSITORY

Git shortlog & git log

git shortlog

Alphabetical list of names and commit messages made by each person

git shortlog -s -n

Displays the number of commits made next to each person's name

git log

Shows all commits made. Full history

git log — stat

Displays names of files changed during the commits

git log --graph

Visual representation of branches, including commits

git log --graph --oneline

Condensed visual representation of branches, including commits

git log -n [number]

Displays specified number of commits only

git log -p [commit id]

Displays changes made to the file(s)

git log -patch [commit id]

Displays changes made to the file(s)

git log -p -w

Ignores whitespace changes

git log -p [file/directory]

Displays change history of file or directory

git log --author=[name]

Filter by author name. Show only their commits

git log --author="full name"

Filter by author's full name. Show only their commits

git log --author="[person 1]\|[person 2]"

Show commits by either person 1 or person 2

git log --grep="Search term"

Show commits which contain the search term only in the commit message

git log --after="[date]"

Display commits made after a certain date

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Cheatography

Git commands Cheat Sheet

by itsellej via cheatography.com/55812/cs/14830/

Git shortlog & git log (cont)

git log --before="[date]"

Display commits made before a certain date

git log --after="[date]" --before="[date]"

Display commits made after **but** before a certain date

git log -- [file name 1] [file name 2]

Display history related to file or files

git log --branches= *

View commits across all branches

Displays list of commits made.
- **Down arrow** scrolls through commit history.
- **Press q** to exit.
- date format = yy-m-d

https://www.atlassian.com/git/tutorials/git-log

Git status

git status

List which files are staged, unstaged, and untracked.

Git show

git show

Display changes made in the last commit

git show [commit id]

Display changes made in a specific commit

git show HEAD

Show details of the commit HEAD is currently pointing at

USING BRANCHES

Git branch

git branch

List of branches in repository

git branch [new branch name]

Creates a new branch

git branch [new branch name] [commit id]

Creates a new branch and points it to the commit specified

git branch -d [branch name]

Deletes a branch. Use -D to force delete

git branch -m [new name]

Rename an existing branch

git branch -a

List all remote branches

https://www.atlassian.com/git/tutorials/using-branches

Git checkout

git checkout [branch name]

Switch to working on another branch

git checkout -b [new branch name]

Create a new branch and switch to it

git checkout [commit id]

Viewing how files were when the commit was created

git checkout HEAD [filename]

Use with unstaged changes. Restore file in working directory to how it is at the last commit

https://www.atlassian.com/git/tutorials/using-branches/git-checkout

Git merge

git merge [branch name]

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Git merge (cont)

[Branch name] is name of branch that will be merged into receiving branch (where HEAD is currently pointing to

- Integrate independent lines of development, created by git branch, and integrate them into a single branch
- use git status to ensure HEAD is pointing to merge receiving branch
- use git fetch to ensure all branches are up to date with remote changes

https://www.atlassian.com/git/tutorials/using-branches/git-merge

OTHER

Git tag

git tag

Displays all current tags

git tag -a [new tag name]

Create a new tag at current commit

git tag -a [new tag name] [7 digits of commit id]

Create a new tag at a previous commit

git tag -d [tag name]

Delete a tag

- Purpose: to point out particular commits / make them stand out
- Example: label with a version number
- Tag stays locked to a commit

git rebase

git rebase -i HEAD~[num]

Merge a number [num] of commits*. Creates a new commit id

*HEAD points to the current location

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com