Cheatography

Git & GitHub Cheat Sheet

by mike-zr via cheatography.com/123658/cs/23379/

Starting A Repo

git init

Initializes a Git repo in current directory.

git init <folder>

Initializes a Git repo in <folder> in current directory. Creates <folder> if it doesn't exist.

Staging And Committing

git add <file>

Adds <file> to staging area.

git add .

Adds all files in current directory to staging area.

git commit -m "<commit message>"

Commits files in staging area with a commit message.

git commit

Opens Git's default editor to write a commit message (can be multi-line). Save and exit to commit files in staging area with the commit message.
Ctrl+W=exit

git commit -am "<commit message>"

Adds modified files to staging area and commits them. (Omitting the -m flag will open default editor for commit message.)

git reset HEAD <file>

Removes <file> from staging area.

git checkout -- <file>

Staging And Committing (cont)

Undoes changes made to <file>, reverting back to last known state of <file> in the repo.

Renaming And Deleting Files

git mv <file> <new name>

Stages the renaming of <file>. The renaming takes place once committed.

git add -u

Stages deletions.

git add -A

Stages both changes/additions and deletions. In effect also stages renaming of files. (git add . does almost the same thing, read more HERE and HERE.)

git rm <file>

Stages the deletion of <file>. Committing will complete the deletion.

Using Git to rename and delete files (as opposed to using the operating system GUI) gives the benefit of Git automatically tracking the renaming and deleting. If you rename a file outside of Git, it will see the file as being deleted along with a new untracked file with the new name.

Information

git status

Returns the status of file changes, as well as new files.

git log

Returns all commits that are a part of the repo.

git show

Returns the last commit with a diff of the changes made. *(Press Q to get out of show command)*

git ls-files

Returns what files Git is tracking.

Differences

git diff

Shows the difference between what's recently changed in the working directory versus the HEAD position in the repo (which is usually the last commit on the branch).

git difftool

Opens the configured diff tool (e.g. P4Merge), and shows the difference between what's recently changed in the working directory versus the HEAD position in the repo (which is usually the last commit on the branch).

git diff <commit id A> <commit id B>

Shows the differences between two commits (one of the commit id's can be HEAD).

git difftool <commit id A> <commit id B>

Differences (cont)

Opens the configured diff tool (e.g. P4Merge) and shows the differences between two commits (one of the commit id's can be HEAD).
In P4Merge press Ctrl+Q to cycle to the next file involved in the diff.

Branching And Merging

git branch

Lists local branches and shows which one you're currently on.

git branch -a

Lists all branches, including remotes, and shows which one you're currently on.

git checkout <branch>

Switches to <branch>.

git checkout -b <branch>

Creates a new <branch> then switches to it.

git merge <branch>

Merges <branch> into the current branch. If there are conflicts, resolving them and committing will complete the merge.

git mergetool

While in a merging state, opens the configured merge tool (e.g. P4Merge).

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

Cheatography

Git & GitHub Cheat Sheet

by mike-zr via cheatography.com/123658/cs/23379/

Tags

git tag <tag name>

Creates a lightweight tag at the current commit.

git tag <tag name> <branch>

Creates a lightweight tag at the last commit on <branch>.

git tag -d <tag name>

Deletes the tag.

git tag -a <tag name> -m "<annotation>"

Creates an annotated tag at the current commit.

git tag -a <tag name> -m "<annotation>" <commit id>

Creates an annotated tag at the specified commit.

git tag --list

Lists tags.

git show <tag>

Shows the details of the commit associated with the <tag>, and the annotation if there is one.

git tag -f <tag> <commit id>

Updates <tag> to be associated with the specified commit. Omitting <commit id> will associate the tag with where HEAD is on the current branch (usually the last commit).

git push <remote ref> <tag>

Pushes the tag to the remote repo.

git push --force <remote ref> <tag>

Forces the push of the updated tag.

git push <remote ref> --tags

Tags (cont)

Pushes all the tags in the local repo to the remote repo.

git push <remote ref> :<tag>

Tells GitHub to delete the tag after the colon.

Remote Repos

git remote -v

Lists remote repos.

git remote add <remote ref> <remote url>

Adds reference to a remote repo. <remote ref> can be anything, but is conventionally called origin.

git remote set-url <remote ref> <remote url>

Updates the URL for the remote reference.

git remote show <remote ref>

Returns info about the remote repo.

git clone <remote url>

Clones the remote repo into the current directory inside a new folder that has the same name as the remote repo. (Git labels this remote as origin by default.)

git clone <remote url> <name>

Clones the remote repo into the current directory in a new folder named <name>. (Git labels this remote as origin by default.)

Fetching And Pulling

git fetch <remote url>

Git goes out to the remote repo and updates its local information about what's in the remote repo.

git fetch -p <remote url>

Git looks for any dead branches and removes those references. The -p stands for prune.

git pull <remote url> <local branch name>

Checks for changes in the remote repo, and pulls any changes to your local branch. It will open a commit window with a default commit message you can use. Will enter a merge state if there are conflicts.

git pull is actually a 2-in-1 command: fetch then merge. It first fetches the updates from the remote repo, then merges those changes into the local repo.

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com