GIT AND COMMAND LINE CHEAT SHEETS

- **Git Documentation**

fournova

GIT CHEAT SHEET

presented by Tower - the best Git client for Mac and Windows

CREATE

Clone an existing repository

$ git clone ssh://user@domain.com/repo.git

Create a new local repository

$ git init

LOCAL CHANGES

Changed files in your working directory

$ git status

Changes to tracked files

$ git diff

Add all current changes to the next commit

$ git add .

Add some changes in <file> to the next commit

$ git add -p <file>

Commit all local changes in tracked files

$ git commit -a

Commit previously staged changes

$ git commit

Change the last commit

Don't amend published commits!

$ git commit --amend

COMMIT HISTORY

Show all commits, starting with newest

$ git log

Show changes over time for a specific file

$ git log -p <file>

Who changed what and when in <file>

$ git blame <file>

BRANCHES & TAGS

List all existing branches

$ git branch -av

Switch HEAD branch

$ git checkout <branch>

Create a new branch based on your current HEAD

$ git branch <new-branch>

Create a new tracking branch based on a remote branch

$ git checkout --track <remote/branch>

Delete a local branch

$ git branch -d <branch>

Mark the current commit with a tag

$ git tag <tag-name>

UPDATE & PUBLISH

List all currently configured remotes

$ git remote -v

Show information about a remote

$ git remote show <remote>

Add new remote repository, named <remote>

$ git remote add <shortname> <url>

Download all changes from <remote>, but don't integrate into HEAD

$ git fetch <remote>

Download changes and directly merge/integrate into HEAD

$ git pull <remote> <branch>

Publish local changes on a remote

$ git push <remote> <branch>

Delete a branch on the remote

$ git branch -dr <remote/branch>

Publish your tags

$ git push --tags

MERGE & REBASE

Merge <branch> into your current HEAD

$ git merge <branch>

Rebase your current HEAD onto <branch>

Don't rebase published commits!

$ git rebase <branch>

Abort a rebase

$ git rebase --abort

Continue a rebase after resolving conflicts

$ git rebase --continue

Use your configured merge tool to solve conflicts

$ git mergetool

Use your editor to manually solve conflicts and (after resolving) mark file as resolved

$ git add <resolved-file>

$ git rm <resolved-file>

UNDO

Discard all local changes in your working directory

$ git reset --hard HEAD

Discard local changes in a specific file

$ git checkout HEAD <file>

Revert a commit (by producing a new commit with contrary changes)

$ git revert <commit>

Reset your HEAD pointer to a previous commit ...and discard all changes since then

$ git reset --hard <commit>

...and preserve all changes as unstaged changes

$ git reset <commit>

...and preserve uncommitted local changes

$ git reset --keep <commit>

30-day free trial available at www.git-tower.com

TOWER

The best Git Client for Mac & Windows

fournova

COMMAND LINE CHEAT SHEET

presented by Tower - the best Git client for Mac and Windows

DIRECTORIES

$ pwd

Display path of current working directory

$ cd <directory>

Change directory to <directory>

$ cd ..

Navigate to parent directory

$ ls

List directory contents

$ ls -la

List detailed directory contents, including hidden files

$ mkdir <directory>

Create new directory named <directory>

OUTPUT

$ cat <file>

Output the contents of <file>

$ less <file>

Output the contents of <file> using the less command (which supports pagination etc.)

$ head <file>

Output the first 10 lines of <file>

$ <cmd> > <file>

Direct the output of <cmd> into <file>

$ <cmd> >> <file>

Append the output of <cmd> to <file>

$ <cmd1> | <cmd2>

Direct the output of <cmd1> to <cmd2>

$ clear

Clear the command line window

FILES

$ rm <file>

Delete <file>

$ rm -r <directory>

Delete <directory>

$ rm -f <file>

Force-delete <file> (add -r to force-delete a directory)

$ mv <file-old> <file-new>

Rename <file-old> to <file-new>

$ mv <file> <directory>

Move <file> to <directory> (possibly overwriting an existing file)

$ cp <file> <directory>

Copy <file> to <directory> (possibly overwriting an existing file)

$ cp -r <directory1> <directory2>

Copy <directory1> and its contents to <directory2> (possibly overwriting files in an existing directory)

$ touch <file>

Update file access & modification time (and create <file> if it doesn't exist)

PERMISSIONS

$ chmod 755 <file>

Change permissions of <file> to 755

$ chmod -R 600 <directory>

Change permissions of <directory> (and its contents) to 600

$ chown <user>:<group> <file>

Change ownership of <file> to <user> and <group> (add -R to include a directory's contents)

SEARCH

$ find <dir> -name "<file>"

Find all files named <file> inside <dir> (use wildcards [*] to search for parts of filenames, e.g. "file.*")

$ grep "<text>" <file>

Output all occurrences of <text> inside <file> (add -i for case-insensitivity)

$ grep -rl "<text>" <dir>

Search for all files containing <text> inside <dir>

NETWORK

$ ping <host>

Ping <host> and display status

$ whois <domain>

Output whois information for <domain>

$ curl -O <url>/<file>

Download <file> (via HTTP(S) or FTP)

$ ssh <username>@<host>

Establish an SSH connection to <host> with user <username>

$ scp <file> <user>@<host>:/remote/path

Copy <file> to a remote <host>

PROCESSES

$ ps ax

Output currently running processes

$ top

Display live information about currently running processes

$ kill <pid>

Quit process with ID <pid>

30-day free trial available at www.git-tower.com

TOWER

The best Git Client for Mac & Windows

HOMEWORK - PART 1

- Make a new folder called “wd3_week1_homework”, and add a copy of the circus starter to it. Create a git repository with that project and do the following:
  - Commit the default circus starter project with the message “Initial Commit”
  - Delete the contents of README.md and replace it with “<your name> - WD3 - Git Homework”
  - Use markdown syntax to make your name bold in the readme and the “WD3 - Git Homework” part cursive
  - Delete the body of the HTML file and replace it with basic markup that includes your name, your quarter, and a list of three things that this pandemic has taught you
  - Commit this to the master branch with the message "Second commit"
- Next, demonstrate the use of branches:
  - Make a new branch called “styling"
  - Add some color and font settings to make your HTML page look nicer. Like you would want an “About Me” page on your portfolio to look
  - Commit your changes to the “styling” branch as you make them. Make sure your commit messages describe what you've done. I want to see at least two commits
  - Merge those changes back into the main branch