MAC TERMINAL COMMANDS

⇨ SHORTCUTS

Key/Command	Description
Ctrl + A	Go to the beginning of the line you are currently typing on
Ctrl + E	Go to the end of the line you are currently typing on
Ctrl + L / Ctrl + K	Clears the screen
Ctrl + U	Cut everything backwards to the beginning of line
Ctrl + K	Cut everything forward to end of linne
Ctrl + Y	Paste whatever was cut by last cut command
Ctrl + C	Kill whatever you are running
Ctrl + D	Exit the shell
Ctrl + (underscore)	Undo the last command
Esc + T	Swap the last two words before the cursor

⇨ CORE COMMMANDS

Key/Command	Description
cd [folder]	change directory
cd ~	home directory
cd /	root of drive
cd -	previous directory
ls	short listing
ls -l	long listing
ls -a	listing incl. hidden files
ls -lh	long listing with human
ls -R	entire content of folder recursively
sudo [command]	run command
open [file]	open a file
top	displays active processes press q to quit
nano[file]	opens the file using nano editor
vim [file]	opens the file using the vim editor
clear	clears the screen
reset	resets the terminal display

⇨ CHAINING COMMANDS

Key/Command	Description
[command-a]; [command-b]	Run command A and then B, regardless of success of A
[command-a] && [command-b]	Run command B if A succedded
[command-a] || [command-b]	Run command B if A failed
[command-a] &	Run command A in background

[command-a] | [command-b]	Run command A and then pass the result to command B.

⇨ COMMAND HISTORY

Key/Command	Description
Ctrl + r	Interactively search through previously typed commands
!!	Execute the last command typed

⇨ FILE MANAGEMENT

Key/Command	Description
touch [file]	Create a new file
pwd	Full path to working directory
.	Current folder
..	Parent/enclosing directory
rm [file]	remove a file
rm -i [file]	remove with confirmation
rm – r[dir]	remove a directory and contents
rm -f [file]	force removal without confirmation
cp [file] [newfile]	copy file to file
cp [file] [dir]	copy file to directory
mv [file] [new filename]	move/rename
pbcopy <[file]	copies files contents to clipboard
pbpaste > [file]	

⇨ DIRECTORY MANAGEMENT

Key/Command	Description
mkdir [dir]	Create a new directory
rmdir [dir]	remove directory
rm -R [dir]	remove directory and contents
[command] > [file]	push output to file
[command] < [file]	tell command to read content from a file

⇨ HELP

Key/Command	Description
[command] -h	offers help
info [command]	offers help
man [command]	show the help manual
whatis [command]	gives a one-line description of

GIT COMMANDS

⇨ GLOSSARY

Keywords	Description
git	Open – source distributed version – control system, used to store code
GitHub, GitLab	Platform for hosting and collaborating on Git repositorites
staging	proposed files/directories that you'd like to commit
commit	saving all staged files/directories to your local repository
branch	An independent line of development, so you can develop features isolated from each other.
clone	local version of a repository that all team members to keep change in sync with
remote	common repository that all team members to keep that changes in sync with
fork	copy of a repository owned by a different user
pull request	A method of submitting contributions to a repository
HEAD	represents your current working directory
untracked	new files that git doesn't yet track
modified	changed
staged	file is ready to be committed
unmodified	unchanged

⇨ CONFIGURATION

Key/Command	Description
git config --global user.name [name]	Set author name to be used for all commits
git config –global user.email [email]	Set author email to be used for all commits

⇨ CORE COMMANDS

Key/Command	Description
git init [directory]	Creates a new local repository
git clone [repo]	Creates local copy of remote repository
git add [directory]	Stages specific [directory]
git add [file]	Stages specific [file]
git add -A	Stages all changed files
git add .	Stages new and changed files
git add -u	Stages changed and deleted files
git commit -m "[message]"	commit everything that is staged
git status	shows status of changed as untracked, modified or staged

⇨ SYNCHRONIZATION OF CHANGES

Key/Command	Description

git fetch	Downloads all history from the remote branches
git merge	Merges remote branch into current local branch
git pull	downloads all history from the remote branch and merges into the current location
git push	pushes all the commits from the current local branch to its remote equivalent

⇨ **UNDO CHANGES**

Key/Command	**Description**
git checkout – [file]	replace file with contents from HEAD
git revert [commit]	create new commit that undoes changes made in [commit], then apply it to the current branch
git reset [file]	remove [file] from staging area
git reset --hard	remove all local changes in working directory

⇨ **BRANCHES**

Key/Command	**Description**
git branch [branch]	Create a new branch
git checkout [branch]	Switch to that branch
git checkout [branch] -b	Create and checkout new branch
git merge [branch]	merge [branch] into current branch
git branch -d [branch]	deletes the [branch]
git push origin [branch]	push [branch] to remote
git branch	lists local branches
git branch -r	list remote branches
git branch -a	list local

⇨ **Working with Git:**

init -> used to create a new repo
git init
git remote add origin <- link ->
git remote -v (to verify remote)
git branch (to check branch)
git branch -M main (to rename branch)
git push -u origin main

git fetch origin main
git pull origin main

⇨ **Branch Commands:**

git branch (to check branch)
git branch -M main (to rename branch)

git checkout <- branch.name -> (to navigate)
git checkout -b <- new branch name -> (to create new branch)
git branch -d <- branch name -> (to delete branch)

⇨ **Merging Code**

Way – 1
git diff <-branch name-> (to compare commits, branches files and more)
git merge <-branch name-> (to merge 2 branches)

Way – 2
Create a PR

⇨ **Pull Command**

git pull origin main

⇨ **Resolving Merge Conflicts**

An event that takes place when Git is unstable to automatically resolve differences in code between two commits.

⇨ **Undoing Changes**

Case 1: staged changes
git reset <-file name->
git reset
Case 2: committed changes (for one commit)
git reset HEAD-1
Case 3: committed changes (for many commits)
git reset <-commit hash->
git reset –hard <-commit hash->

⇨ **Updates were rejected because the tip of your current branch is behind**

i. git fetch origin
ii. git reset –hard origin/main
iii. git push -f origin main
iv. clear

⇨ If there exists any problem with pushing then use: git push -f origin main
⇨ fatal: refusing to merge unrelated histories -> git pull origin branchname --allow-unrelated-histories