

GIT AND COMMAND LINE CHEAT SHEETS

Git Documentation





GIT CHEAT SHEET

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



CREATE Clone an existing repository <code>\$git clone <url></code> Create a new local repository <code>\$git init</code>	BRANCHES & TAGS List all existing branches <code>\$git branch -a</code> Switch HEAD branch <code>\$git checkout <branch></code> Create a new branch based on your current HEAD <code>\$git branch <new-branch></code> Create a new tracking branch based on a remote branch <code>\$git checkout -b <new-branch></code> Delete a local branch <code>\$git branch -d <branch></code> Mark the current commit with a tag <code>\$git tag <tag-name></code>	MERGE & REBASE Merge <branch> into your current HEAD <code>\$git merge <branch></code> Rebase your current HEAD onto <branch> <i>Don't rebase published commits!</i> <code>\$git rebase <branch></code> Abort a rebase <code>\$git rebase --abort</code> Continue a rebase after resolving conflicts <code>\$git rebase --continue</code> Use your configured merge tool to solve conflicts <code>\$git merge tool</code> Use your editor to manually solve conflicts and (after resolving) mark file as resolved <code>\$git add <resolved-file></code> <code>\$git rm <resolved-file></code>
LOCAL CHANGES Changed files in your working directory <code>\$git status</code> Changes to tracked files <code>\$git diff</code> Add all current changes to the next commit <code>\$git add .</code> Add some changes in <file> to the next commit <code>\$git add -p <file></code> Commit all local changes in tracked files <code>\$git commit -a</code> Commit previously staged changes <code>\$git commit</code> Change the last commit <i>Don't commit published commits!</i> <code>\$git commit --amend</code>	UPDATE & PUBLISH List all currently configured remotes <code>\$git remote -v</code> Show information about a remote <code>\$git remote show <remote></code> Add new remote repository, named <remote> <code>\$git remote add <remote> <url></code> Download all changes from <remote>, but don't integrate into HEAD <code>\$git fetch <remote></code> Download changes and directly merge/integrate into HEAD <code>\$git pull <remote> <branch></code> Publish local changes on a remote <code>\$git push <remote> <branch></code> Delete a branch on the remote <code>\$git branch -d <remote/branch></code> Publish your tags <code>\$git push --tags</code>	UNDO Discard all local changes in your working directory <code>\$git reset --hard HEAD</code> Discard local changes in a specific file <code>\$git checkout HEAD <file></code> Revert a commit (by producing a new commit with contrary changes) <code>\$git revert <commit></code> Reset your HEAD pointer to a previous commit ...and discard all changes since then <code>\$git reset --hard <commit></code> ...and preserve all changes as unstaged changes <code>\$git reset <commit></code> ...and preserve uncommitted local changes <code>\$git reset --keep <commit></code>

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

TOWER
The best Git Client for Mac & Windows



COMMAND LINE CHEAT SHEET

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



DIRECTORIES <code>\$pwd</code> Display path of current working directory <code>\$cd <directory></code> Change directory to <directory> <code>\$cd ..</code> Navigate to parent directory <code>\$ls</code> List directory contents <code>\$ls -la</code> List detailed directory contents, including hidden files <code>\$mkdir <directory></code> Create new directory named <directory>	FILES <code>\$rm <file></code> Delete <file> <code>\$rm -r <directory></code> Delete <directory> <code>\$rm -f <file></code> Force-delete <file> (add -r to force-delete a directory) <code>\$mv <file-old> <file-new></code> Rename <file-old> to <file-new> <code>\$mv <file> <directory></code> Move <file> to <directory> (possibly overwriting an existing file) <code>\$cp <file> <directory></code> Copy <file> to <directory> (possibly overwriting an existing file) <code>\$cp -r <directory1> <directory2></code> Copy <directory1> and its contents to <directory2> (possibly overwriting files in an existing directory) <code>\$touch <files></code> Update file access & modification time (and create <file> if it doesn't exist)	SEARCH <code>\$find <dir> -name "<file>"</code> Find all files named <file> inside <dir> (use wildcards [?] to search for parts of filenames, e.g. "file.*") <code>\$grep "<text>" <file></code> Output all occurrences of <text> inside <files> (add -r for recursive) <code>\$grep -r "<text>" <dir></code> Search for all files containing <text> inside <dir>
OUTPUT <code>\$cat <file></code> Output the contents of <file> <code>\$less <file></code> Output the contents of <file> using the less command (which supports pagination etc.) <code>\$head <file></code> Output the first 10 lines of <file> <code>\$<cmd> > <file></code> Direct the output of <cmd> into <file> <code>\$<cmd> >> <file></code> Append the output of <cmd> to <file> <code>\$<cmd1> <cmd2></code> Direct the output of <cmd1> to <cmd2> <code>\$clear</code> Clear the command line window	PERMISSIONS <code>\$chmod <rb> <file></code> Change permissions of <file> to <rb> <code>\$chmod <R 600> <directory></code> Change permissions of <directory> (and its contents) to 600 <code>\$chown <user>:<group> <file></code> Change ownership of <file> to <user> and <group> (add <R> to include a directory's contents)	NETWORK <code>\$ping <host></code> Ping <host> and display status <code>\$whois <domains></code> Output whois information for <domains> <code>\$curl -O <url/outfile></code> Download <file> (via HTTP[S] or FTP) <code>\$ssh <username>@<host></code> Establish an SSH connection to <host> with user <username> <code>\$scp <file> <user>@<host>:<remote/path></code> Copy <file> to a remote <host>
PROCESSES <code>\$ps ax</code> Output currently running processes <code>\$top</code> Display live information about currently running processes <code>\$kill <pid></code> 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