

Git Cheat Sheet

Daniel Park

March 26, 2022

Contents

| | | |
|----------|--------------------------|----------|
| 1 | Preliminaries | 2 |
| 2 | Terminal commands | 2 |
| 3 | Git commands | 3 |
| 4 | Github commands | 5 |

1 Preliminaries

| Step | Description |
|--|---|
| Installing VS Code | Open up VSCode. Go to ‘Shell Command: Install ‘code command ...’ to set up the environment variable. This enables the command <code>code</code> . |
| <code>command -flag <dir></code> | Every terminal and git command follows the given format. All file and folder names <code><dir></code> must be in quotations. |
| Configure VS Code | Run <code>git config --global core.editor "code --wait</code> in git bash. This will configure VS Code as the default git editor over VIM. |

2 Terminal commands

| Command | Description |
|--|--|
| <code>ls</code> | Lists contents of current directory |
| <code>ls <folder></code> | Lists contents of <code><folder></code> |
| <code>ls -a</code> | Lists all directories and files, including hidden ones |
| <code>start .</code> <code>open .</code> | Opens up the file explorer in the current directory The <code>start</code> command on Mac |
| <code>pwd</code> | Shows the current directory location |
| <code>cd <folder></code> <code>cd</code> <code>cd ..</code> | Changes your current directory to <code><folder></code> Go back to home directory. Note that <code>~</code> denotes home directory Move backwards to the parent folder of current directory |
| <code>clear</code> | Clears the terminal |
| <code>code .</code> | Opens the current directory in VSCode |
| <code>touch <file></code> <code>touch <file1> <file2></code> <code>mkdir <folder></code> | Creates a file called <code><file></code> in the current directory Creates files called <code><file1></code> and <code><file2></code> in the current directory Creates a folder called <code><folder></code> |
| <code>rm <file></code> <code>rm -rf <folder></code> | Deletes the file called <code><file></code> Deletes the folder <code><folder></code> |

3 Git commands

Git structure:

Working directory $\xrightarrow{\text{git add}}$ Staging area $\xrightarrow{\text{git commit}}$ Local repo $\xrightarrow{\text{git push}}$ Remote repo

Git GUI: [Git Kraken](#). Use the GUI to copy commit hashes.

Resources:

- Full documentation: git-scm.com/docs
- Book on git: git-scm.com/book

Git ignore template: [git ignore io](#)

- Use touch `‘.gitignore’` to create the file.
- Set up the `.gitignore` file first before committing files. Otherwise you must remove the files to be ignored from the cache.

| Command | Description |
|---|--|
| <code>git init</code> | Initializes the repository in the current directory. Creates a hidden folder called <code>‘.git’</code> . |
| <code>git status</code> | Shows the status of the current git repository. |
| <code>git add <file></code> <code>git add .</code> | Adds <code><file></code> to the staging area. Adds all changed files to the staging area. |
| <code>git commit</code> <code>git commit -m <msg></code> <code>git commit --amend</code> | Uploads all staged files to the git repository. Creates a commit. Creates a commit with the message <code><msg></code> . Updates the previous commit with all currently staged changes. |
| <code>git log</code> <code>git log --oneline</code> | Shows the history of all commits made in the repo. Condenses the log into abbreviated hashes and the header line of each commit message. |
| <code>git branch</code> <code>git branch <branch></code> | Lists all branches. Active branch has an asterisk (*). Creates a new branch <code><branch></code> with the parent commit being where HEAD is on. Multiple branches share the same parent commit. |
| <code>git switch <branch></code> <code>git checkout <branch></code> <code>git switch -c <branch></code> <code>git checkout <branch></code> | Switches the active branch to <code><branch></code> . All unstaged changes are lost. A generalized command of <code>git switch</code> and <code>git restore</code> . Creates a new branch and switches you over to it. Does the same thing as <code>git switch -c <branch></code> . |
| <code>git branch -d <branch></code> | Deletes the branch <code><branch></code> . Must be fully merged before doing so. |

| | |
|---|---|
| <code>git branch -D ⟨branch⟩</code> | Deletes the branch <code>⟨branch⟩</code> unconditionally. |
| <code>git branch -m ⟨branch⟩</code> | Renames the current branch name to <code>⟨branch⟩</code> . |
| <code>git merge ⟨branch⟩</code> | Merges <code>⟨branch⟩</code> to the active branch. If there is a merge conflict, it must be resolved manually. After, you must stage and commit these resolved changes. |
| <code>git diff</code> <code>git diff HEAD</code> <code>git diff ⟨branch1⟩ ⟨branch2⟩</code> <code>git diff ⟨commit1⟩ ⟨commit2⟩</code> | Shows all unstaged changes since the last commit. Lists all changes in the working tree since your last commit. Includes both staged and unstaged changes. Lists all changes between the tips of <code>⟨branch1⟩</code> and <code>⟨branch2⟩</code> . Lists all changes between the commit hashes <code>⟨commit1⟩</code> and <code>⟨commit2⟩</code> . |
| <code>git stash (save)</code> <code>git stash pop</code> <code>git stash clear</code> | Saves all uncommitted changes to the stash. Removes the most recently stashed items and reapplies them to your working directory. Clears all stashes. |
| <code>git checkout HEAD ⟨file⟩</code> <code>git checkout HEAD ⟨k⟩</code> <code>git restore ⟨file⟩</code> <code>git restore --source ⟨commit⟩ ⟨file⟩</code> <code>git restore --staged ⟨file⟩</code> | Discard any changes made in that file, reverting it back to HEAD. Detach HEAD <i>k</i> commits back. To re-attach HEAD, use the <code>git switch</code> command. Alternative command to <code>git checkout</code> . Reverts <code>⟨file⟩</code> to its previous state at the commit <code>⟨commit⟩</code> . Unstages the file <code>⟨file⟩</code> . |
| <code>git reset ⟨commit⟩</code> <code>git reset --hard ⟨commit⟩</code> <code>git revert</code> | Resets the repo back to a previous commit. All changes persist as unstaged changes. Implements git reset, and all changes are removed from your working directory. Undo the changes as a new commit. Does not delete prior commits made. |

4 Github commands

| Command | Description |
|---|--|
| <code>git clone <url></code> | Retrieves all files associated with the github repository in <url> and will copy them to your local machine. Creates a new folder with the cloned repo in it, and assigns the default remote name 'origin'. |
| <code>git remote -v</code> <code>git remote add <name> <url></code> <code>git remote rename <old-name> <new-name></code> <code>git remote remove <name></code> | <p>Lists all remote names.</p> <p>Creates a new remote called <name> linked to the github repo <url>.</p> <p>Renames the remote <old-name> to <new-name>.</p> <p>Removes the remote <name>.</p> |
| <code>git push <remote> <branch></code> <code>git push -u <remote> <branch></code> <code>git push <remote> <local-b>:<remote-b></code> | <p>Pushes work in branch up to the remote branch.</p> <p>Sets the upstream of the local branch so that it tracks the corresponding remote branch with the same name. This allows us to set up the shortcut command <code>git push</code>.</p> <p>Pushes the local branch <local-b> up to the remote branch <remote-b>.</p> |
| <code>git branch -r</code> <code>git fetch <remote> <remote-b></code> | <p>View all remote branches tracked by the local repository.</p> <p>Takes remote changes from the remote branch <remote-b> and creates a new branch called <remote>/<remote-b> on your local repo.</p> |