

Git Cheat Sheet

Thom Parkin, Author

Important Terminology/Concepts

- Working Directory (Working Tree)
 - Edited code files
- Index (Staging Area)
 - Files/changes to be part of an upcoming commit
- History
 - Committed project repository
- Branches
 - Points to a commit and assoc. history (lightweight)

.gitignore

```
src/.secrets/ # entire directory requires trailing slash
*.src # without fwd slash ignore applies to related files
/*.*idea # root path required to be specific
!*.src # use bang(!) to negate an ignore pattern
```

\$ man gitignore for details and examples

HOW TO GIT HELP

\$ man git or \$ git help <command>

Configuration

Configuration is divided into *System*, *Global* and *Local*. Global configuration applies to all projects under git control on the system. Local config will affect only the local project on which it is set (overriding Global or System). The essential configuration is username and email, like so:

```
$ git config --global user.name "Codee Wrighter"
$ git config --global user.email "codeninja@sp.com"
```

\$ git config --list displays current config data

Remove a File From the Entire Commit History

—**tree-filter** option runs the specified command after each checkout and then recommits the results. Here's how you would remove all occurrences of 'pswds.txt' in the history:

```
$ git filter-branch --tree-filter 'rm -f pswds.txt' HEAD
```

Staging Changes and Making Commits

Changes that git will track must be moved to the index (staged)

```
git add filename.txt more-filenames.log directory/plus-filename entire_directory/.
```

git add -u Update the index for files already being tracked

\$ git add -A Mass ADD of all files in the working tree

Git Cheat Sheet

Thom Parkin, Author

Staging Changes and Making Commits (Continued)

Changes that git will track must be moved to the index (staged)

```
git add filename.txt more-filenames.log directory/plus-filename entire_directory/.
```

`git add -u` Update the index for files already being tracked

`$ git add -A` Mass ADD of all files in the working tree

`$ git add -n` Dry-run to show what will be added to the index

`$ git diff --staged` **--staged** is an alias for **--cached**

`$ git diff 3b45ff90 785fe4` `git diff HEAD~3` Diff between 2 commits/current HEAD and 3 commits

`$ git status`

`$ git reset filename`

`$ git checkout filename`

`$ git rm` and `git rm --cached`

Branching and Merging

`$ git checkout -b new_branchname` Creates a branch

`$ git push origin new-branch`

`$ git branch` Shows all branches

`$ git reset filename`

`$ git branch -r` Displays only remote branches

`$ git checkout filename`

`$ git checkout branchname` if remote `git checkout origin/branch-name`

`git checkout -b localname origin/branchname` Set local tracking branch

`$ git fetch -p` View changes (pull request) before merging them

Git Cheat Sheet

Thom Parkin, Author

Working With the Index

`git add -i` Select only parts of changed files (chunks)

`$ git commit --amend` is useful after `$ git add forgotten-file` but also to update the commit message

Only use `$ git commit --amend` with nothing in the index

`$ git stash list` Will list the current stash

`$ git stash save "memorable name"` Will save the entire state of the index

`$ git stash pop "memorable name"` Restores state and deletes entry in stash (can be on another branch)

`$ git stash clear` Will eliminate all stashed commits

Changing History

`$ git revert bd9345` Removes commit from the history

`$ git reset --hard` Resets uncommitted changes (in the index)

Rebase rewrites history (by changing lineage of commits)

`$ git checkout master` then `$ git rebase feature` will apply **feature** ON **master**

`$ git rebase -i HEAD~3` Interactively (selectively) rebase from 3 commits ago

`$ git push --force` Deletes previous commits in favor of this one

*Tip: on shared repositories it is better to use **revert** than push a new commit!*

Git Cheat Sheet

Thom Parkin, Author

Clone Remote (With Tracking)

```
$ git clone <repository url>
```

```
$ git clone --depth=1 <repository url>
```

Check out only the latest commit (not the entire history)

Update Tracking Branch(es)

```
$ git remote add upstream <repository url> then $ git pull --all
```

How to Fix Merge Conflicts

- Edit the affected files (removing the merge markers “<<<<” and “====” as appropriate)
 - Add the changed file(s) to the index once more and then create a new commit
-

Other Useful Commands

```
$ git reflog
```

 Can be a savior because it is a record of every action performed on the repository

```
$ git blame main.c
```

 Shows a history of every change to this file

```
$ git blame -L 10,12 main.c
```

 Limits it to lines 10 through 12 (inclusive)

“Magically” Locate the Commit Where Things Went Wrong

```
$ git bisect start then $ git bisect good 23df99 then $ git bisect bad HEAD
```

About the Author

Thom has been writing software since the days when phones had wires. A self-proclaimed pariahacker and huge board gamer living in Florida, Thom is an advisor in the Sitepoint Forums, always learning and anxious to share and teach others.