GIT QUICK REFERENCE CARD

Most frequently used GIT commands – Version v1.1 May 2014

Main commands

`add` ..... Add file contents to the index

- `-A [:/]` adds all file changes including deletions to the staging area [for the whole tree]
- `-f` allow adding otherwise ignored files
- `-p` choose hunks of patch between the index and the work tree and add them to the index

`am` ..... Apply a series of patches from a mailbox

`archive` ..... Create an archive of files from a named tree

`bisect` ..... Find by binary search the change that introduced a bug

⇒ `search for existence of` *file* `in last commits: bisect run test -f` *file.txt*

`branch` ..... List, create, or delete branches

- `-a` lists all branches including the remote branches
- `-d` delete `...branch`
- `-m` rename branch: ... `old_name new_name`
- `-v` lists more information about the branches

`bundle` ..... Move objects and refs by archive

`checkout` ..... checkout a branch or paths to the working tree (prepend `--` before path to avoid ambiguousity).

- `-b` create a branch `new_name` and switch to it instantly

⇒ `restore working tree and reset index: checkout HEAD - -` *dir*

`cherry-pick` . can be used to change order of commits

`citool` ..... Graphical alternative tocommit

`clean` ..... Remove untracked files from the working tree

- `-f` force (required!)
- `-x` include hidden files
- `-d` directories as well

`clone` ..... Clone a repository into a new directory

`commit` ..... Record changes to the repository

- `-a` adding changes of files known by the Git repository to the commit
- `-m` Set comment for commitment – correct it with prepending `--amend` to argument

`describe` ..... Show the most recent tag that is reachable from a commit

`diff` ..... Show changes between commits, commit and working tree, etc.

- `--cached` compare stage area with last commit

`fetch` ..... Download objects and refs from another repository

`format-patch` Prepare patches for e-mail submission

`gc` ..... Cleanup unnecessary files and optimize the local repository

`grep` ..... Print lines matching a pattern

`gui` ..... A portable graphical interface to Git

`init` ..... Create an empty git repository or reinitialize an existing one

`log` ..... Show commit logs (prepend `--` before path to avoid ambiguousity)

- `-p` show diffs of each commit
- `--stat` precise output with some stats
- `--abbrev-commit` shortend (but unique) SHA-1 values for the commit objects
- `--oneline` show logs as shortened oneliner
- `--reverse` reverse log-history output
- `--follow` entire history including renames
- `--graph` show visual timeline

⇒ `show all files that ever existed in HEAD: git log --pretty=format: --name-status | cut -f2- | sort -u`

`merge` ..... Join two or more development histories together

- `-s` specify your merge strategy (see `man git-merge`)
- `-X` apply option to strategy (eg. *ours*, *theirs*)
- `--ff` only update the branch pointer, without creating a merge commit (default)

`mv` ..... Move or rename a file, a directory, or a symlink

`notes` ..... Add/inspect commit notes

`pull` ..... Fetch from and merge with another repository or a local branch

`push` ..... Update remote refs along with associated objects

⇒ `delete branch in remote repo called testbranch: push origin :`testbranch

`rebase` ..... Forward-port local commits to the updated upstream head

- `-i` rebase interactively
- `--autosquash` When the commit log message begins with "squash! ..." (or "fixup! ..."), automatically modify the todo list of `rebase -i`

`reset` ..... Reset current HEAD to the specified state (Remove a *file* from staging area)

- `--soft` move HEAD pointer only
- `--mixed` resets staging area to new HEAD (default!)
- `--hard` resets staging area and working tree to new HEAD

`revert` ..... Revert an existing commit

`rm` ..... Remove files from the working tree and from the index

- `-r` remove recursive – for removing additionally from repo, add `--cached *`

`shortlog` ..... Summarize git log output

`show` ..... Show various types of objects

⇒ `allows you to see and retrieve files from branches, commits and tags: show [reference]:[filename] > out.txt`

`stash` ..... Stash the changes in a dirty working directory away

`status` ..... Show the working tree status

`submodule` ... Initialize, update, add or inspect submodules

`--recurse-submodules`
pull changes from main repo + submodules

`tag` ..... Create, list, delete or verify a tag object signed with GPG

`gitk` ..... The git repository browser

Manipulators

`config` ..... Get and set repository or global options

`--list` show all items in config

`--get-regexp`
show given pattern in my configuration, e.g. `'alias'`

`--global` write to global `~/.gitconfig` file

`--system` write to system-wide configuration

`--local` to repository's configuration `.git/config`

⇒ `set alias for staged file prompt: config --global alias.staged 'diff --cached'`

`fast-export` . Git data exporter

`fast-import` . Backend for fast Git data importers

`filter-branch`
Rewrite branches

`mergetool` ... Run merge conflict resolution tools to resolve merge conflicts

`pack-refs` ... Pack heads and tags for efficient repository access

`prune` ..... Prune all unreachable objects from the object database

`reflog` ..... Manage reflog information

`relink` ..... Hardlink common objects in local repositories

`remote` ..... Manage set of tracked repositories

`-v` show details about the remotes

⇒ `remove all branches which are not in the remote anymore: remote update --prune`

`repack` ..... Pack unpacked objects in a repository

`replace` ..... Create, list, delete refs to replace objects

Interrogators

`annotate` ..... Annotate file lines with commit information

`blame` ..... Show what revision and author last modified each line of a file

`cherry` ..... Find commits not merged upstream

`count-objects`
Count unpacked number of objects and their disk consumption

`difftool` ..... Show changes using common diff tools

`fsck` ..... Verifies the connectivity and validity of the objects in the database

`get-tar-commit-id`
Extract commit ID from an archive created usingarchive

`help` ..... Display help information about git

`instaweb` ..... Instantly browse your working repository in gitweb

`merge-tree` .. Show three-way merge without touching index

`rerere` ..... Reuse recorded resolution of conflicted merges

`rev-parse` ... Pick out and massage parameters

`show-branch` . Show branches and their commits

`verify-tag` .. Check the GPG signature of tags

`whatchanged` . Show logs with difference each commit introduces

Interacting with Others

`archimport` .. Import an Arch repository into git

`cvsexportcommit`
Export a single commit to a CVS checkout

`cvsimport` ... Salvage your data out of another SCM people love to hate

`cvsserver` ... A CVS server emulator for git

`imap-send` ... Send a collection of patches from stdin to an IMAP folder

`quiltimport` . Applies a quilt patchset onto the current branch

`request-pull` Generates a summary of pending changes

`send-email` .. Send a collection of patches as emails

`svn` ..... Bidirectional operation between a Subversion repository and git

Manipulators (plumbing)

`apply` ..... Apply a patch to files and/or to the index

`checkout-index`
Copy files from the index to the working tree

`commit-tree` . Create a new commit object

`hash-object` . Compute object ID and optionally creates a blob from a file

`index-pack` .. Build pack index file for an existing packed archive

`merge-file` .. Run a three-way file merge

`merge-index` . Run a merge for files needing merging

`mktag` ..... Creates a tag object

`mktree` ..... Build a tree-object from ls-tree formatted text

`pack-objects` Create a packed archive of objects

`prune-packed` Remove extra objects that are already in pack files

`read-tree` ... Reads tree information into the index

`symbolic-ref` Read and modify symbolic refs

`unpack-objects`
Unpack objects from a packed archive

`update-index` Register file contents in the working tree to the index

`update-ref` .. Update the object name stored in a ref safely

`write-tree` .. Create a tree object from the current index

Interrogators (plumbing)

`cat-file` ..... Provide content or type and size information for repository objects

`diff-files` .. Compares files in the working tree and the index

`diff-index` .. Compares content and mode of blobs between the index and repository

`diff-tree` ... show files which have been changed by last commit

`--name-only` show only names of files

`-r` recursively

`for-each-ref` Output information on each ref

`ls-files` ..... Show information about files in the index and the working tree

`ls-remote` ... List references in a remote repository

`ls-tree` ..... List the contents of a tree object

⇒ `list all files for specific branch: ls-tree -r master --name-only`

`merge-base` .. Find as good common ancestors as possible for a merge

`name-rev` ..... Find symbolic names for given revs

`pack-redundant` Find redundant pack files

`rev-list` ..... Lists commit objects in reverse chronological order

`show-index` .. Show packed archive index

`show-ref` ..... List references in a local repository

`unpack-file` . Creates a temporary file with a blobs contents

`var` ..... Show a git logical variable

`verify-pack` . Validate packed git archive files

Synching repositories

`daemon` ..... A really simple server for git repositories

`fetch-pack` .. Receive missing objects from another repository

`http-backend` Server side implementation of Git over HTTP

`send-pack` ... Push objects over git protocol to another repository

`update-server-info` Update auxiliary info file to help dumb servers

Helper commands

`http-fetch` .. Download from a remote git repository via HTTP

`http-push` ... Push objects over HTTP/DAV to another repository

`parse-remote` Routines to help parsing remote repository access parameters

`receive-pack` Receive what is pushed into the repository

`shell` ..... Restricted login shell foronly SSH access

`upload-archive` Send archive back toarchive

`upload-pack` . Send objects packed back tofetch-pack

Internal helper commands

`check-attr` .. Display gitattributes information

`check-ref-format` Ensures that a reference name is well formed

`fmt-merge-msg` Produce a merge commit message

`mailinfo` ..... Extracts patch and authorship from a single e-mail message

`mailsplit` ... Simple UNIX mbox splitter program

`merge-one-file` The standard helper program to use withmerge-index

`patch-id` ..... Compute unique ID for a patch

`sh-setup` ..... Common git shell script setup code

`stripspace` .. Filter out empty lines

emzap79@gmail.com

This TeXfile is based on Gabriel B. Burcas © git-qrc.tex and has then been modified to my own use.