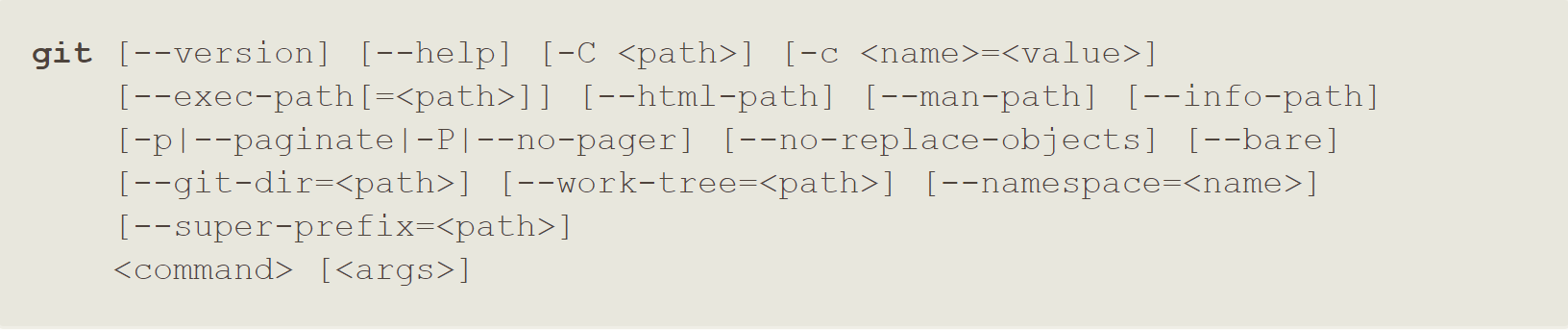
Git

Agenda:

1. Setup and Config
   1. git
   2. config
   3. help
2. Getting and Creating Projects
   1. init
   2. clone
3. Basic Snapshotting
   1. add
   2. status
   3. diff
   4. commit
   5. notes
   6. restore
   7. reset
   8. rm
   9. mv
4. Branching and Merging
   1. branch
   2. checkout
   3. switch
   4. merge
   5. mergetool
   6. log
   7. stash
   8. tag
   9. worktree
5. Sharing and Updating Projects
   1. fetch
   2. pull
   3. push
   4. remote
   5. submodule
6. Inspection and Comparison
   1. show
   2. log
   3. diff
   4. difftool
   5. range-diff
   6. shortlog
   7. describe
7. Gitignore
8. Patching
   1. apply
   2. cherry-pick
   3. diff
   4. rebase
   5. revert
9. Debugging
   1. bisect
   2. blame
   3. grep
10. Email
    1. am
    2. apply
    3. format-patch
    4. send-email
    5. request-pull
11. External Systems
    1. svn
    2. fast-import
12. Administration
    1. clean
    2. gc
    3. fsck
    4. reflog
    5. filter-branch
    6. instaweb
    7. archive
    8. bundle
13. Server Admin
    1. daemon
    2. update-server-info
14. Plumbing Commands
    1. cat-file
    2. check-ignore
    3. checkout-index
    4. commit-tree
    5. count-objects
    6. diff-index
    7. for-each-ref
    8. hash-object
    9. ls-files
    10. ls-tree
    11. merge-base
    12. read-tree
    13. rev-list
    14. rev-parse
    15. show-ref
    16. symbolic-ref
    17. update-index
    18. update-ref
    19. verify-pack
    20. write-tree
15. Setup and Config
    1. Git



**Prints the Git suite version that the git program came from.**

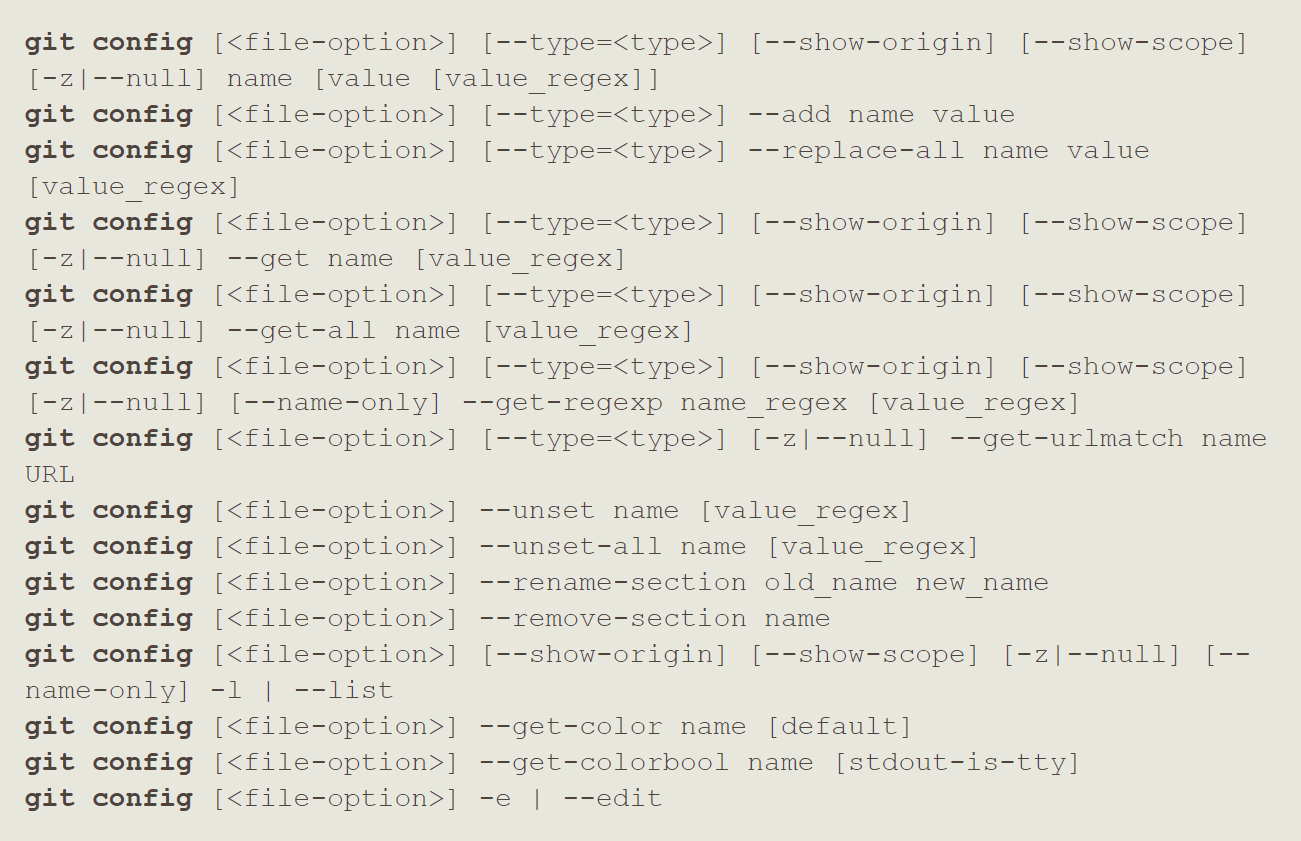
**$ git --version**

**Prints the synopsis and a list of the most commonly used commands.**

**$ git --help**

**$ git --help --all | -a**

* 1. Config



**Checking configuration settings:**

**$ git config --list**

**You can view all of your settings and where they are coming from using:**

**$ git config –-list --show-origin**

**Set your user name and email address globally:**

**$ git config --global user.name "<username>"**

**$ git config --global user.email <email>**

**Set your user name and email address for a specific project:**

**$ git config user.name "<username>"**

**$ git config user.email <email>**

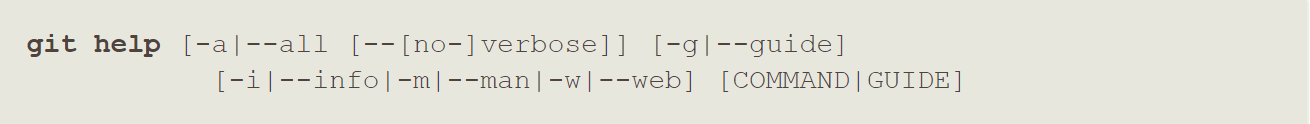
**Set up text editor for git:**

**$ git config --global core.editor <editor-name>**

**You can also check what Git thinks a specific key’s value is by typing:**

**$ git config <key>**

* 1. Help



**If you ever need help while using Git, there are three equivalent ways to get the comprehensive manual**

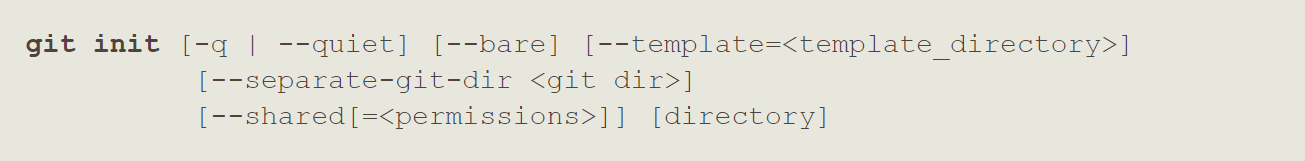
**page (manpage) help for any of the Git commands:**

**$ git help <verb>**

**$ git <verb> --help**

**$ man git-<verb>**

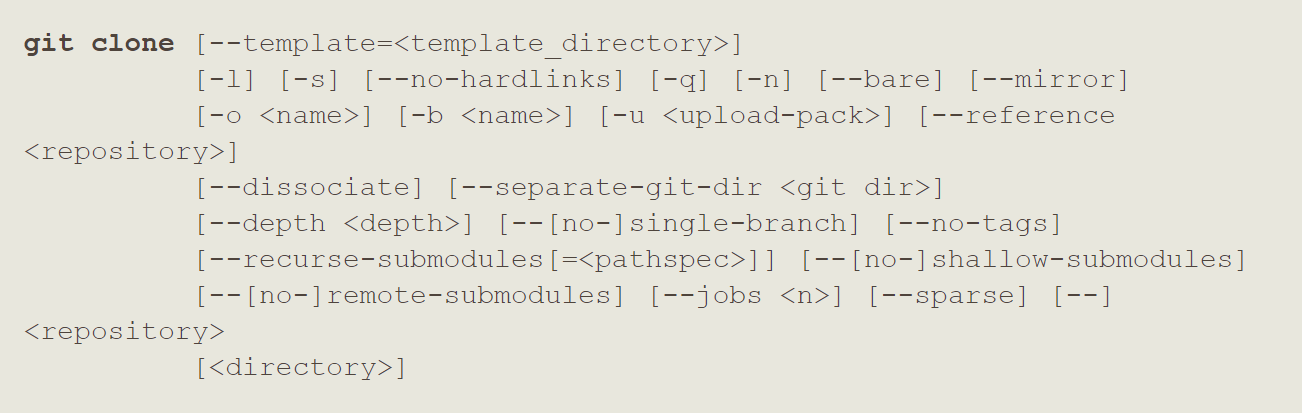
1. Getting and Creating Projects
   1. Init



**Creating a git repository:**

**$ git init**

* 1. Clone

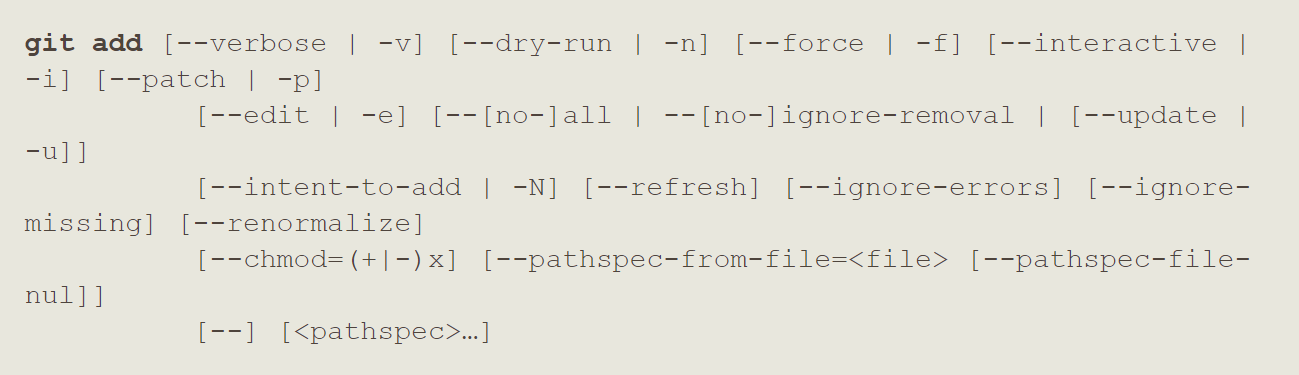


**Cloning an existing repository:**

**$ git clone <url>**

**$ git clone <url> <filename>**

1. Basic Snapshotting
   1. Add



**Tracking new files:**

**$ git add <filename>**

**$ git add . ~track all files**

* 1. Status



**Checking status of files:**

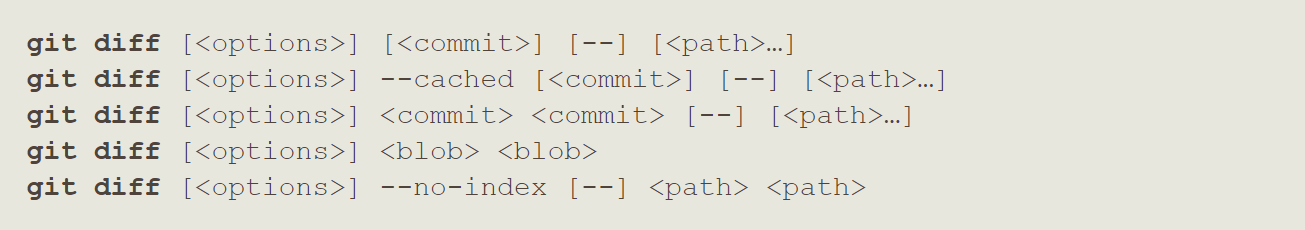
**$ git status**

**Short status:**

**$ git status -s**

**$ git status --short**

* 1. Diff



**See changes between modified & unstaged file:**

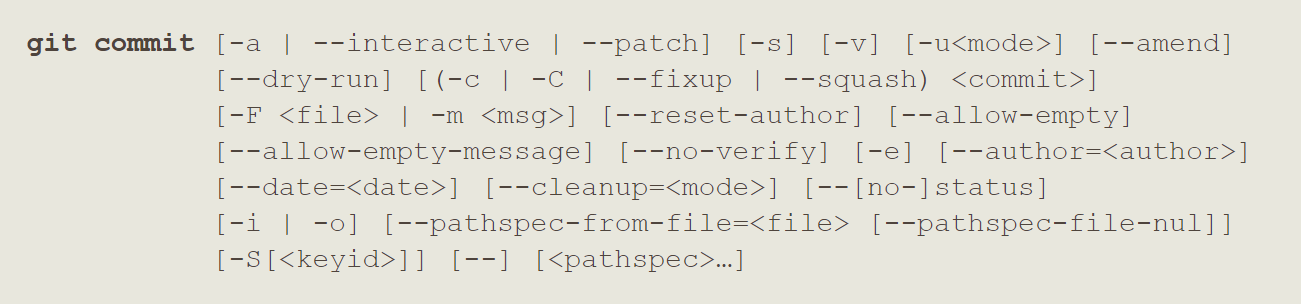
**$ git diff**

**See changes between staged & last commit:**

**$ git diff –-staged**

**$ git diff --cached**

* 1. Commit



**Simple commit:**

**$ git commit**

**Put the diffs in the commit:**

**$ git commit -v**

**Commit with message inline:**

**$ git commit -m “<message>”**

**Skip the staging area and direct commit:**

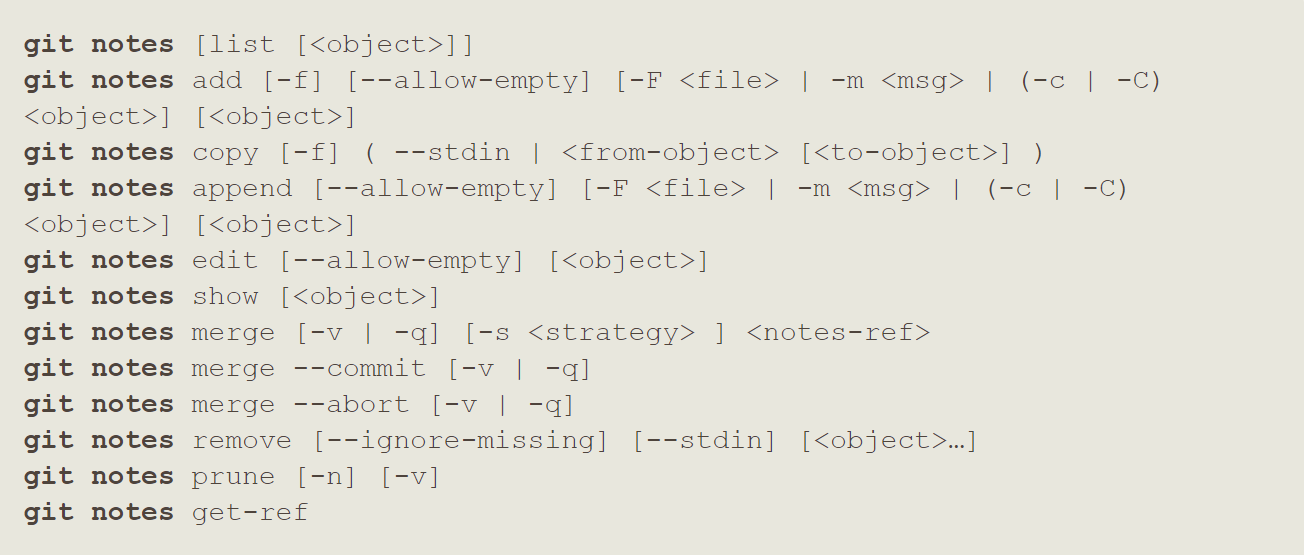
**$ git commit -a -m “<message>”**

**$ git commit -am “<message>”**

**If you want to redo that commit, make the additional changes you forgot, stage them, and commit again:**

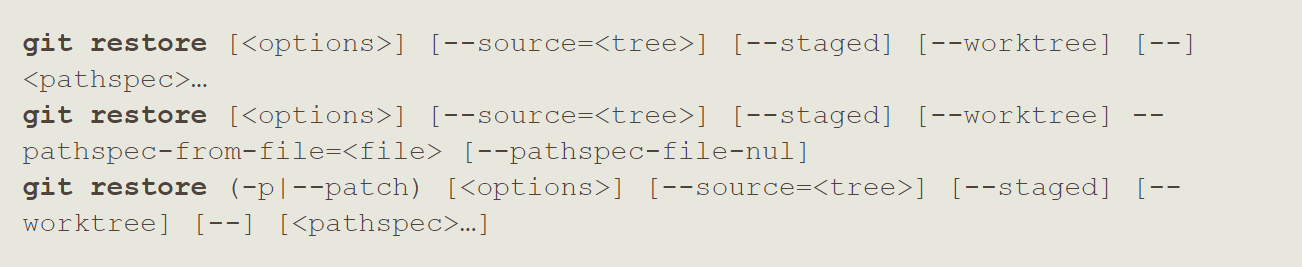
**$ git commit --amend**

* 1. Notes



**$**

* 1. Restore



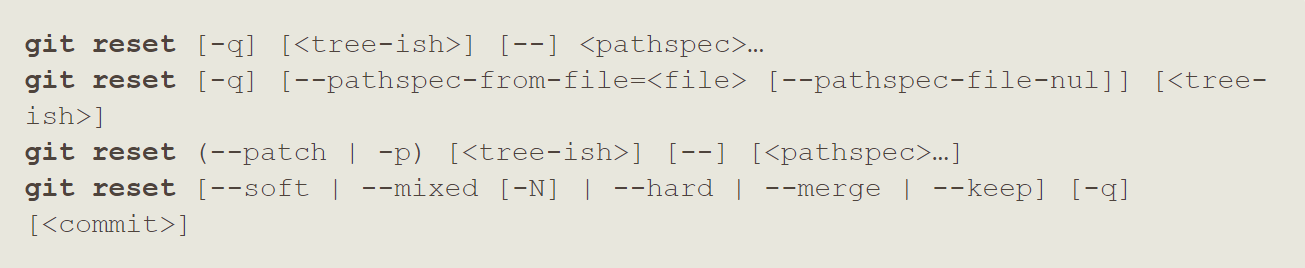
**Unstaging a Staged File:**

**$ git restore --staged <filename>**

**Unmodifying a Modified File:**

**$ git restore <filename>**

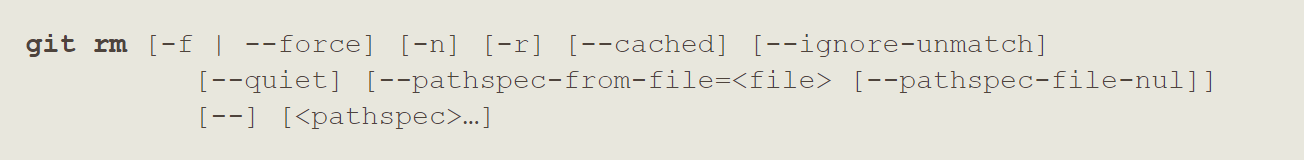
* 1. Reset



**Unstaging a Staged File:**

**$ git reset HEAD <filename>**

* 1. Rm



**Removing files:**

**$ git rm <filename>**

**$ git rm <directories> | <file-globe-patterns>**

**Force removal:**

**$ git rm -f <filename>**

**Untrack a file:**

**$ git rm –-cached <filename>**

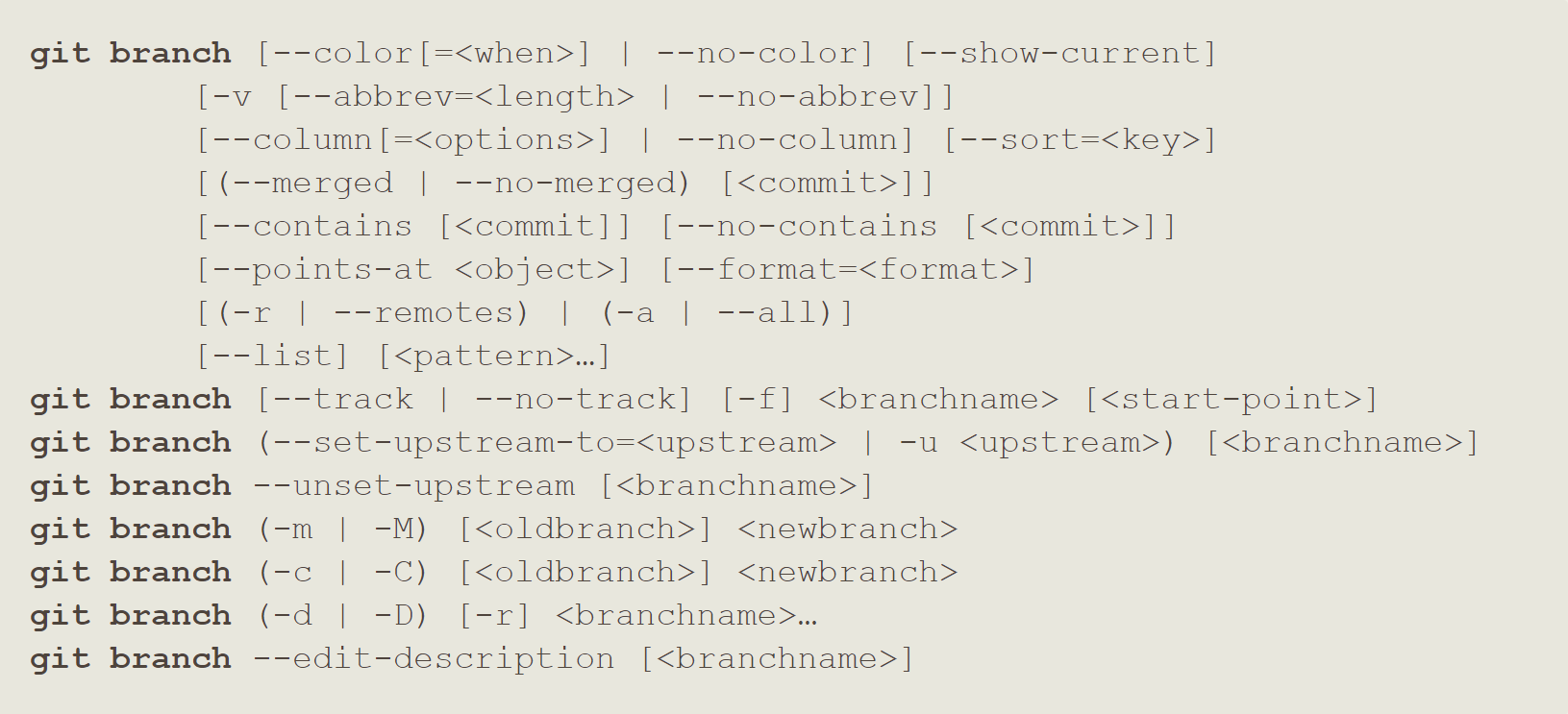
* 1. Mv



**Renaming a file:**

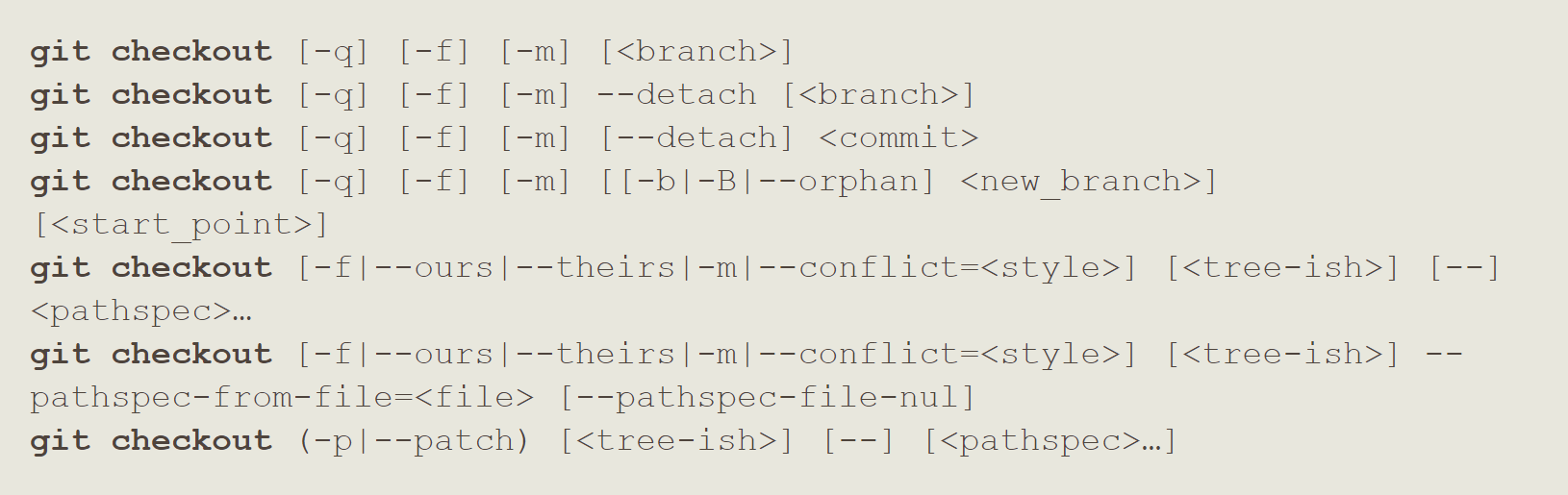
**$ git mv <old-filename> <new-filename>**

1. Branching and Merging
   1. Branch



**$**

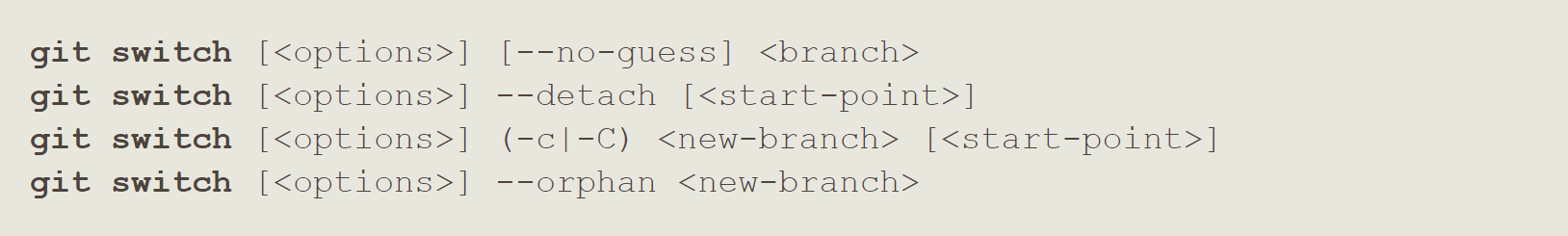
* 1. Checkout



**Unmodifying a Modified File:**

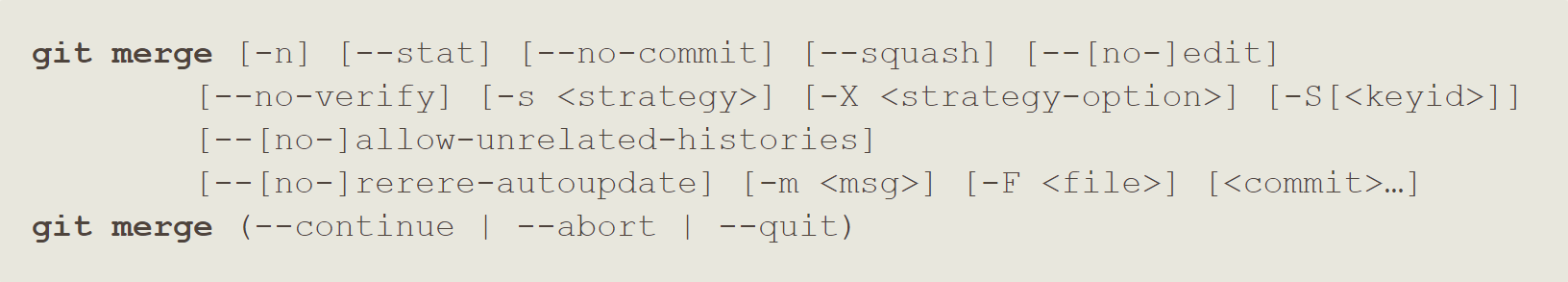
**$ git checkout -- <filename>**

* 1. Switch



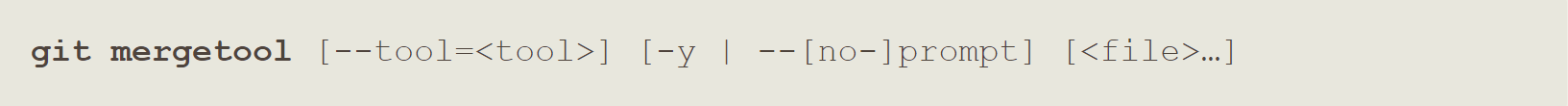
**$**

* 1. Merge



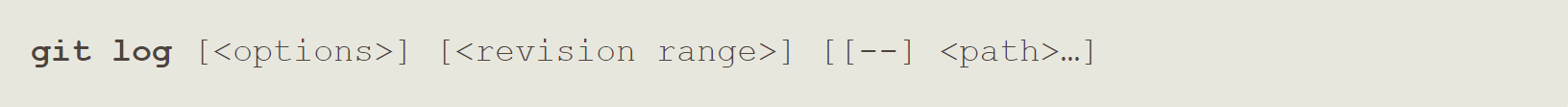
**$**

* 1. Mergetool



**$**

* 1. Log



**Viewing commit history:**

**$ git log**

**History with difference:**

**$ git log -p**

**$ git log –-patch**

**History with difference and limit:**

**$ git log -p -<n> ~ n: positive integer**

**History with stats & shortstats:**

**$ git log --stat**

**$ git log –-shortstat**

**Formatted History:**

**$ git log --pretty=<option>**

**~ option: oneline, short, full, fuller, format**

**$ git log --pretty=format:”format-options”**

**Formatted History with graph:**

**$ git log --pretty=<option> --graph**

**Show the list of files modified after the commit information:**

**$ git log --name-only**

**Show the list of files affected with added/ modified / deleted information as well:**

**$ git log --name-status**

**Show only the first few characters of the SHA-1 checksum instead of all 40:**

**$ git log --abbrev-commit**

**Display the date in a relative format (for example, “2 weeks ago”) instead of using the full date format:**

**$ git log --relative-date**

**Limiting log output:**

**$ git log --since=<n>.<days-weeks-years>**

**$ git log --until=<n>.<days-weeks-years>**

**$ git log --before=<n>.<days-weeks-years>**

**$ git log --after=<n>.<days-weeks-years>**

**$ git log --since=”<specific-date , relative date>”**

**Only show commits in which the author entry matches the specified string:**

**$ git log –author <author-name>**

**Only show commits in which the committer entry matches the specified string:**

**$ git log --committer <committer-name>**

**Only show commits with a commit message containing the string:**

**$ git log --grep “<string>”**

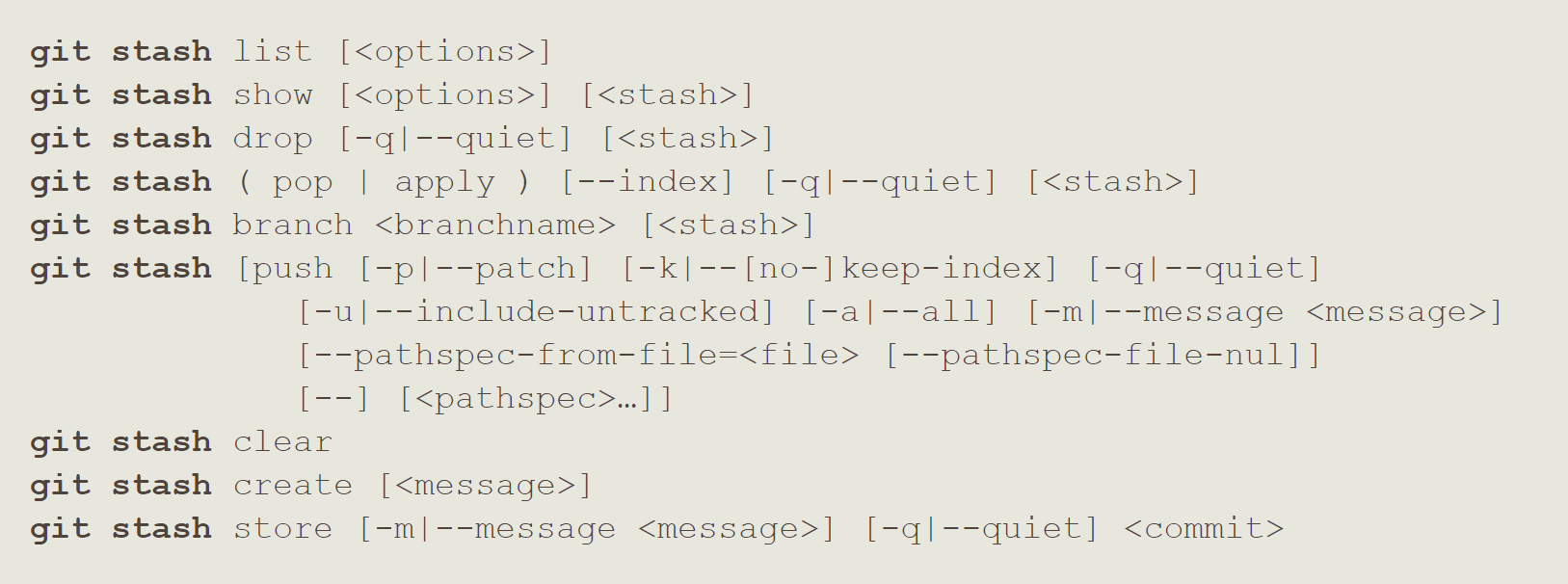
**Only show commits adding or removing code matching the string:**

**$ git log -S “<string>”**

**Prevent the display of merge commits:**

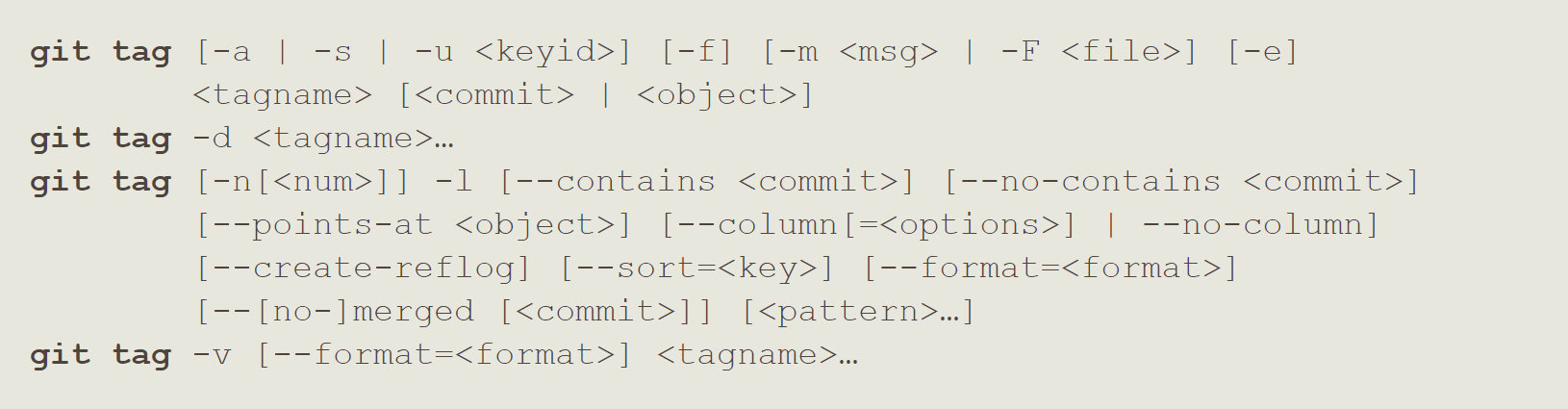
**$ git log --no-merges**

* 1. Stash



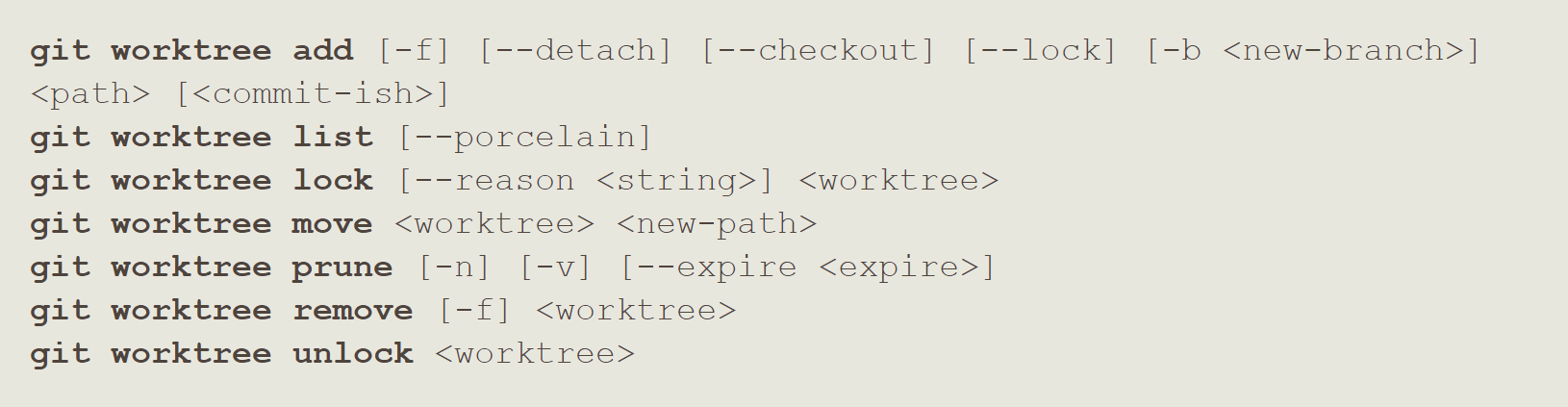
**$**

* 1. Tag



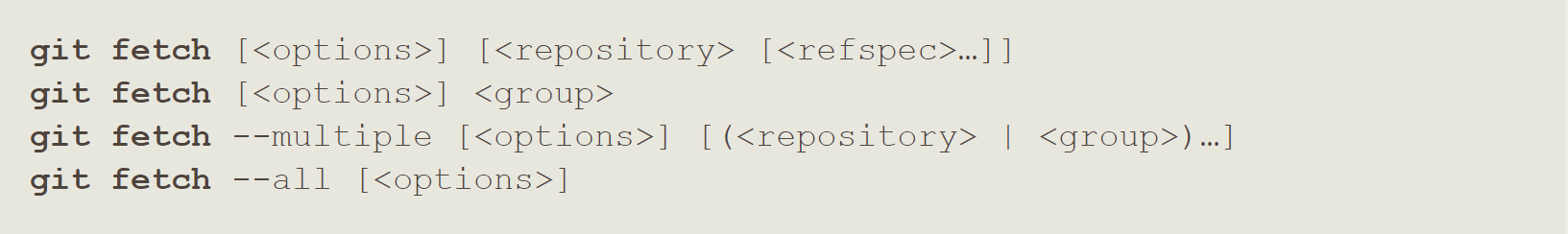
**$**

* 1. Worktree



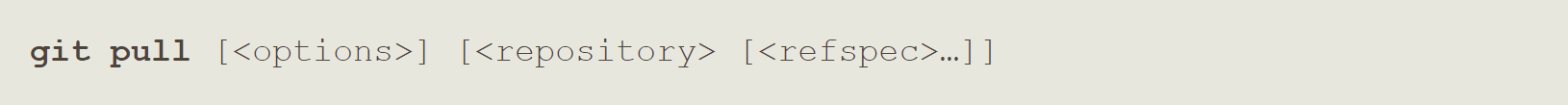
**$**

1. Sharing and Updating Projects
   1. Fetch



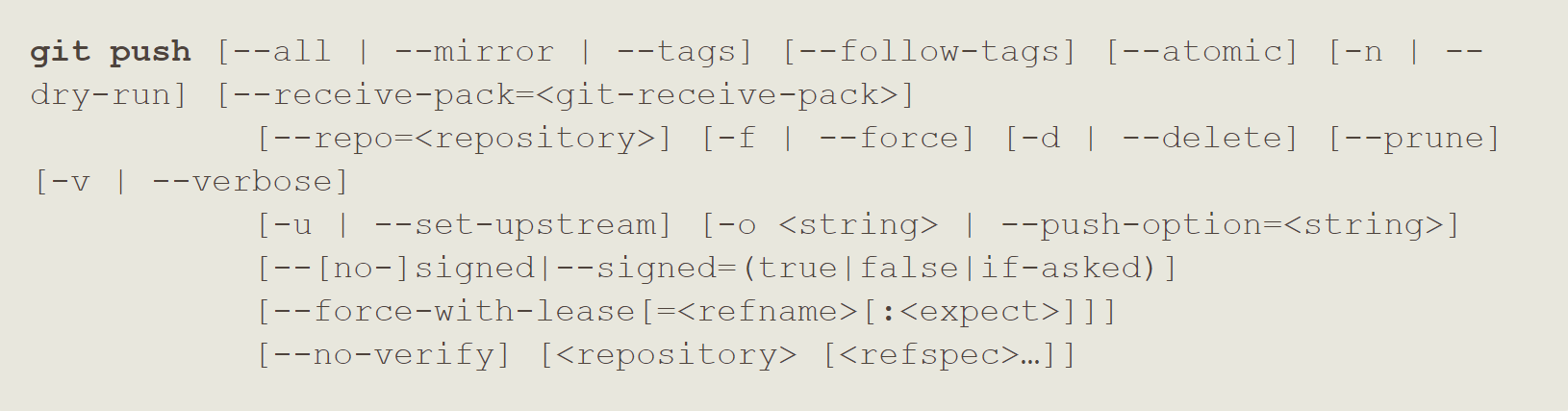
**$**

* 1. Pull



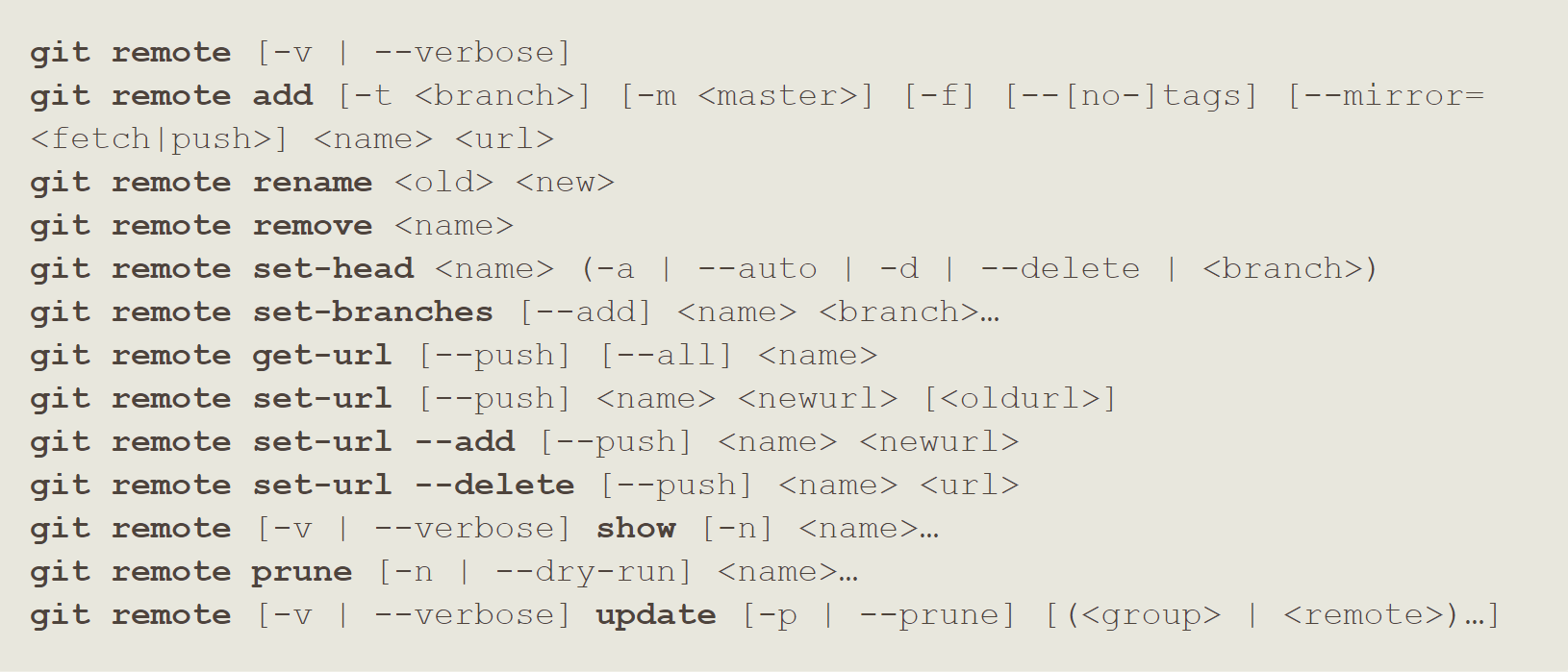
**$**

* 1. Push



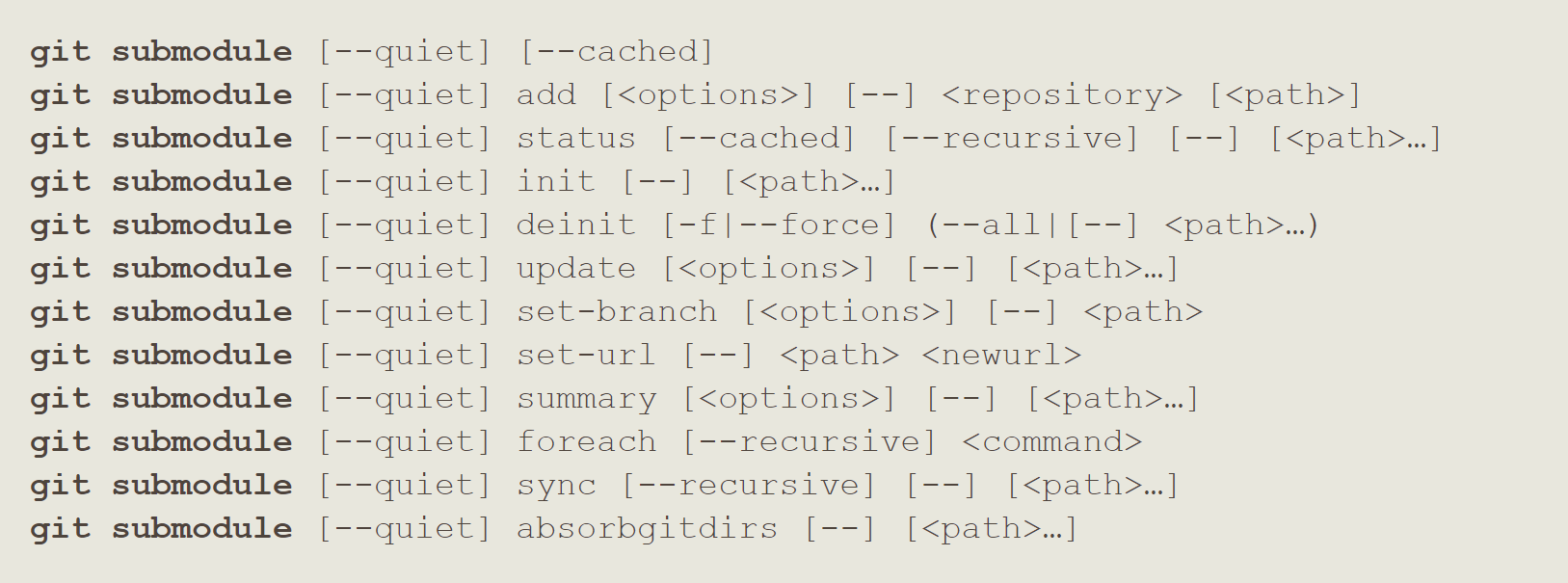
**$**

* 1. Remote



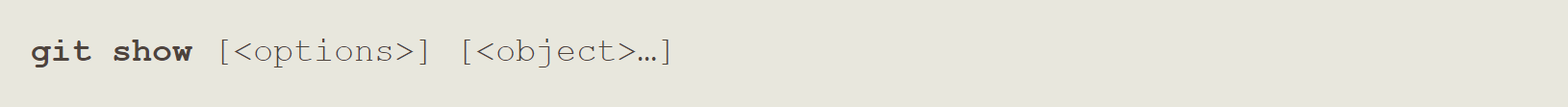
**$**

* 1. Submodule



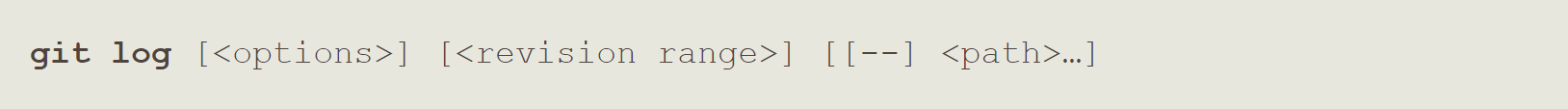
**$**

1. Inspection and Comparison
   1. Show



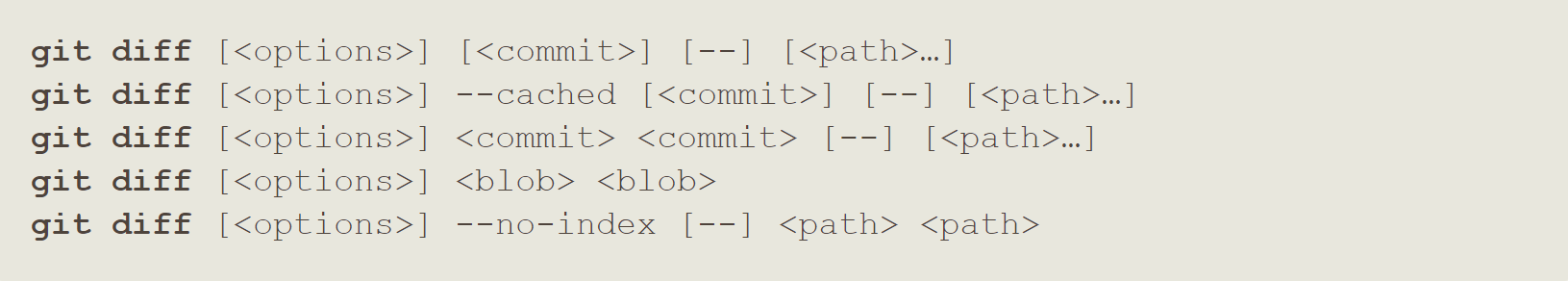
**$**

* 1. Log



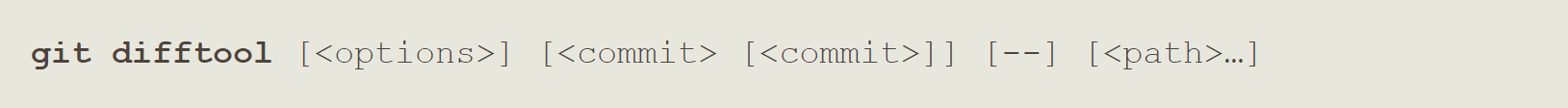
**$**

* 1. Diff



**$**

* 1. Difftool



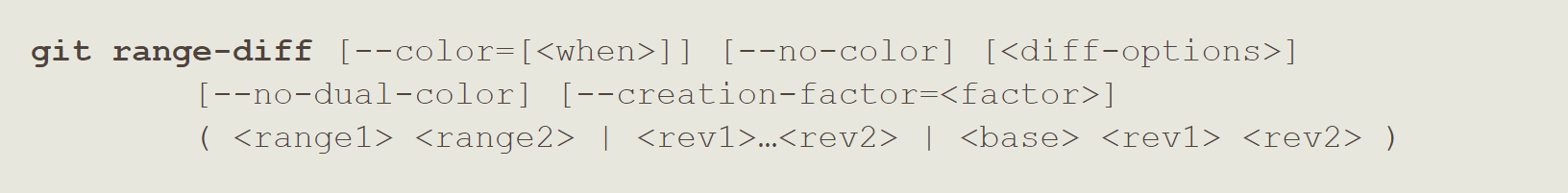
**See difference in an external diff showing program:**

**$ git difftool --tool=<tool>**

**To list the available diff showing programs:**

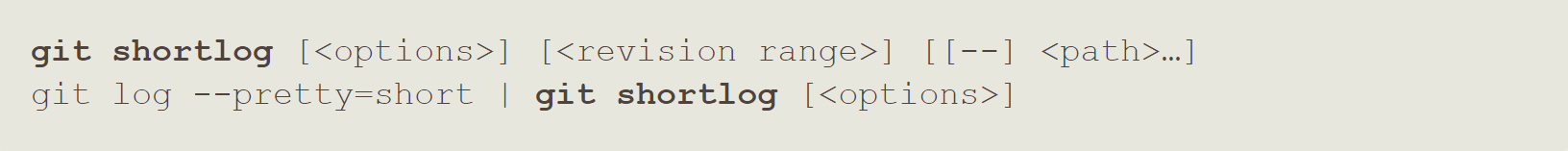
**$ git difftool --tool-help**

* 1. range-diff



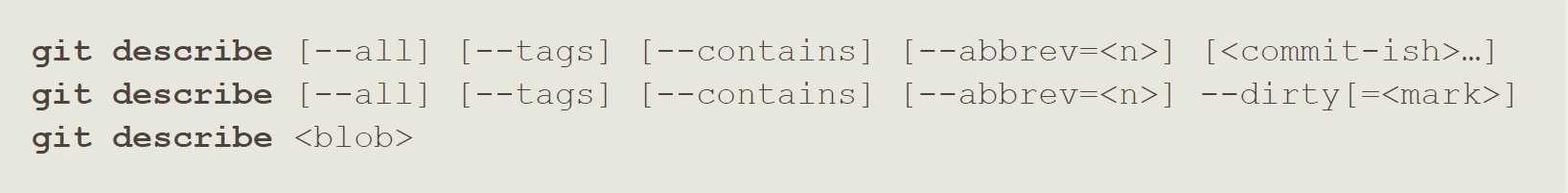
**$**

* 1. shortlog



**$**

* 1. describe



**$**

1. Gitignore
2. Patching
   1. apply
   2. cherry-pick
   3. diff
   4. rebase
   5. revert
3. Debugging
   1. bisect
   2. blame
   3. grep
4. Email
   1. am
   2. apply
   3. format-patch
   4. send-email
   5. request-pull
5. External Systems
   1. svn
   2. fast-import
6. Administration
   1. clean
   2. gc
   3. fsck
   4. reflog
   5. filter-branch
   6. instaweb
   7. archive
   8. bundle
7. Server Admin
   1. daemon
   2. update-server-info
8. Plumbing Commands
   1. cat-file
   2. check-ignore
   3. checkout-index
   4. commit-tree
   5. count-objects
   6. diff-index
   7. for-each-ref
   8. hash-object
   9. ls-files
   10. ls-tree
   11. merge-base
   12. read-tree
   13. rev-list
   14. rev-parse
   15. show-ref
   16. symbolic-ref
   17. update-index
   18. update-ref
   19. verify-pack
   20. write-tree