Git: Commands and Documentation

INDIVIDUAL DEVELOPER (STANDALONE) top

A standalone individual developer does not exchange patches with other people, and works alone in a single repository, using the following commands.

- git-init(1) to create a new repository.
- git-log(1) to see what happened.
- git-switch(1) and git-branch(1) to switch branches.
- git-add(1) to manage the index file.
- git-diff(1) and git-status(1) to see what you are in the middle of doing.
- git-commit(1) to advance the current branch.
- git-restore(1) to undo changes.
- git-merge(1) to merge between local branches.
- git-rebase(1) to maintain topic branches.
- git-tag(1) to mark a known point.

INDIVIDUAL DEVELOPER (PARTICIPANT) top

A developer working as a participant in a group project needs to learn how to communicate with others, and uses these commands in addition to the ones needed by a standalone developer.

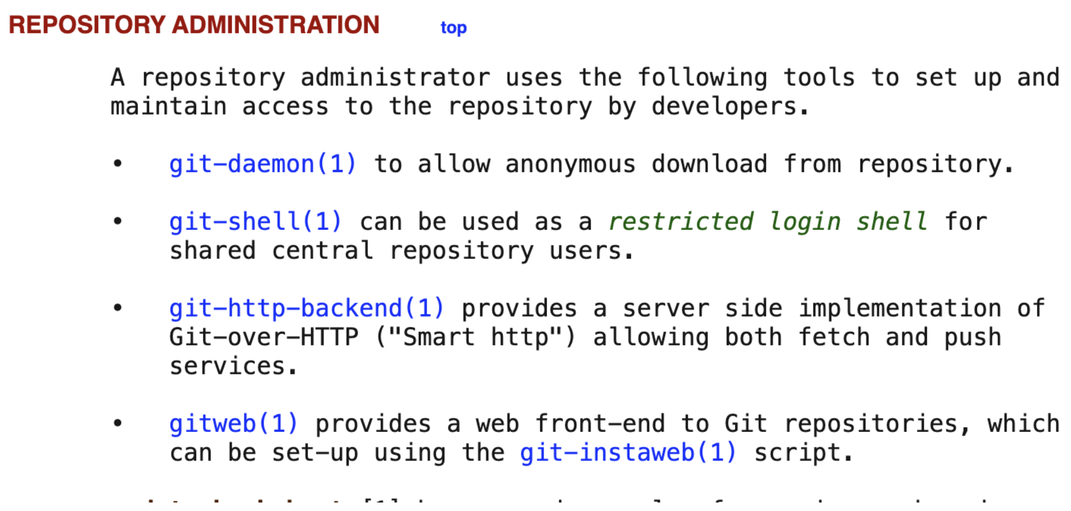
- git-clone(1) from the upstream to prime your local repository.
- git-pull(1) and git-fetch(1) from "origin" to keep up-to-date with the upstream.
- git-push(1) to shared repository, if you adopt CVS style shared repository workflow.
- git-format-patch(1) to prepare e-mail submission, if you adopt Linux kernel-style public forum workflow.
- git-send-email(1) to send your e-mail submission without corruption by your MUA.
- git-request-pull(1) to create a summary of changes for your upstream to pull.

INTEGRATOR top

A fairly central person acting as the integrator in a group project receives changes made by others, reviews and integrates them and publishes the result for others to use, using these commands in addition to the ones needed by participants.

This section can also be used by those who respond to **git request-pull** or pull-request on GitHub (www.github.com) to integrate the work of others into their history. A sub-area lieutenant for a repository will act both as a participant and as an integrator.

- git-am(1) to apply patches e-mailed in from your contributors.
- git-pull(1) to merge from your trusted lieutenants.
- git-format-patch(1) to prepare and send suggested alternative to contributors.
- git-revert(1) to undo botched commits.
- git-push(1) to publish the bleeding edge.



REPOSITORY ADMINISTRATION top

A repository administrator uses the following tools to set up and maintain access to the repository by developers.

- git-daemon(1) to allow anonymous download from repository.
- git-shell(1) can be used as a *restricted login shell* for shared central repository users.
- git-http-backend(1) provides a server side implementation of Git-over-HTTP ("Smart http") allowing both fetch and push services.
- gitweb(1) provides a web front-end to Git repositories, which can be set-up using the git-instaweb(1) script.

```
Git Basic Commands:

git pull

git push

git clone

git log         // shows history of commits



// ------ git commit ------ //

git commit

git commit -m "message" // the m flag will skip the need to ope

git commit -am "message" // this does ALL changed files



GitHub Commands:
```

```
Git Documentation:



tom@Toms-MacBook-Pro ~ % apropos git
git(1)                   - the stupid content tracker
git-add(1)               - Add file contents to the index
git-am(1)                - Apply a series of patches from a mail
git-annotate(1)          - Annotate file lines with commit infor
git-apply(1)             - Apply a patch to files and/or to the
git-archive(1)           - Create an archive of files from a nam
git-bisect(1)            - Use binary search to find the commit
git-blame(1)             - Show what revision and author last mo
git-branch(1)            - List, create, or delete branches
git-bugreport(1)         - Collect information for user to file
git-bundle(1)            - Move objects and refs by archive
git-cat-file(1)          - Provide content or type and size info
git-check-attr(1)        - Display gitattributes information
git-check-ignore(1)      - Debug gitignore / exclude files
git-check-mailmap(1)     - Show canonical names and email addres
git-check-ref-format(1)  - Ensures that a reference name is well
git-checkout(1)          - Switch branches or restore working tr
git-checkout-index(1)    - Copy files from the index to the work
git-cherry(1)            - Find commits yet to be applied to ups
git-cherry-pick(1)       - Apply the changes introduced by some
git-citool(1)            - Graphical alternative to git-commit
git-clean(1)             - Remove untracked files from the worki
git-clone(1)             - Clone a repository into a new directo
git-column(1)            - Display data in columns
git-commit(1)            - Record changes to the repository
git-commit-graph(1)      - Write and verify Git commit-graph fil
git-commit-tree(1)       - Create a new commit object
git-config(1)            - Get and set repository or global opti
```

```
git-count-objects(1)     - Count unpacked number of objects and
git-credential(1)        - Retrieve and store user credentials
git-credential-cache(1)  - Helper to temporarily store passwords
git-credential-cache--daemon(1) - Temporarily store user credent
git-credential-store(1)  - Helper to store credentials on disk
git-daemon(1)            - A really simple server for Git reposi
git-describe(1)          - Give an object a human readable name
git-diagnose(1)          - Generate a zip archive of diagnostic
git-diff(1)              - Show changes between commits, commit
git-diff-files(1)        - Compares files in the working tree ar
git-diff-index(1)        - Compare a tree to the working tree or
git-diff-tree(1)         - Compares the content and mode of blob
git-difftool(1)          - Show changes using common diff tools
git-fast-export(1)       - Git data exporter
git-fast-import(1)       - Backend for fast Git data importers
git-fetch(1)             - Download objects and refs from anothe
git-fetch-pack(1)        - Receive missing objects from another
git-filter-branch(1)     - Rewrite branches
git-fmt-merge-msg(1)     - Produce a merge commit message
git-for-each-ref(1)      - Output information on each ref
git-for-each-repo(1)     - Run a Git command on a list of reposi
git-format-patch(1)      - Prepare patches for e-mail submission
git-fsck(1)              - Verifies the connectivity and validit
git-fsck-objects(1)      - Verifies the connectivity and validit
git-fsmonitor--daemon(1) - A Built-in Filesystem Monitor
git-gc(1)                - Cleanup unnecessary files and optimiz
git-get-tar-commit-id(1) - Extract commit ID from an archive cre
git-grep(1)              - Print lines matching a pattern
git-hash-object(1)       - Compute object ID and optionally crea
git-help(1)              - Display help information about Git
git-hook(1)              - Run git hooks
git-http-backend(1)      - Server side implementation of Git ove
:
```