Objects

BLOB
- create: `hash-object -w <file>`
- show: `cat-file -p <hash>`

TREE
- create: `mktree`
  - see `ls-tree(1)`, "OUTPUT FORMAT"
- from idx: `write-tree`
- show: `ls-tree <tree>`

COMMIT
- create: `commit-tree <tree> < msg`
  - from idx: `commit -m <msg>`
- show: `log -1 <commit>`
  - or: `cat-file -p <commit>`

TAG
- create: `mktag < sig`
- show: `cat-file -p <tag>`

Refs

LOCATIONS (in `.git/`)
- branches: `refs/heads/<name>`
- remotes: `refs/remotes/<name>/*`
- tags: `refs/tags/<name>`
- *: `refs/*`
- HEAD: `HEAD`

MODIFY
- update-ref: write dst hash
  - set: `update-ref <path> <dst>`
  - multiple lines via `--stdin`
- symbolic-ref: write dst pointer
  - create: `symbolic-ref <path> <dst>`

PACKED
- pack all: `pack-refs --all`
- update: `pack-refs`

Transfer

ENDPOINTS
- fetch
  - client: `fetch-pack <remote>`
  - server: `upload-pack <dir>`
- push
  - client: `send-pack <host>:<dir>`
  - server: `receive-pack <dir>`
- archive
  - client: `archive --remote=<host>`
  - server: `upload-archive <dir>`

PROTOCOLS
- `<proto>::<addr>` uses `remote-<proto>`
  - `http::<url>` (curl)
  - `ftp::<url>` (curl)
  - `fd::<infd>[,<outfd>]`
  - `ext::<cmd> [arg...]`
- see `man gitremote-helpers`

FETCH/PUSH REFSPEC
- format: `[+]<src>[:<dst>]`
  - where: `<src>` is any revspec, `<dst>` is ref
    `+` is force push / overwriting fetch
- rm ref: `:<ref>`

CLONE OPTS
- info/alternates: `--reference <repo>`
- rm alternates: `--dissociate`
- specific branch: `-b <name>`
  - only fetch one: `--single-branch`
- shallow clone: `--depth <nr>`

OFFLINE
- binary: `bundle create <file> <revlist>`
  - extract: `bundle unbundle` or `clone` (!)
- text: `fast-export <revlist>`
  - pipe to: `fast-import`

Index (stage/cache)

SHOW
- show: `ls-files --stage`
- diff/tree: `diff-index --cached <tree>`
- diff/wd: `diff-files`

LOAD
- from tree: `read-tree <tree> [tree]...`
- from wd: `add [-u] <path>`
  - or: `update-index --{add|..}`
- raw: `update-index --cacheinfo`
  - or from stdin with `--index-info`

STORE
- to tree: `write-tree`
- to wd: `checkout-index`

Pack

SHOW
- idx file: `verify-pack -v file.idx`
  - raw: `show-index < file.idx`
- extra: `pack-redundant --all`

MAKE
- pack+idx: `pack-objects < objlist`
  - lost idx?: `index-pack <pack>`

EXTRACT
- into repo: `unpack-objects < pack`

Recovery

FINDING LOST
- reflog: `reflog -10`
- dangling: `fsck`
  - w/ stage: `fsck --cache`
  - extract: `fsck --lost-found`
    - into `.git/lost-found/`
- bad pack: `unpack-objects -r`
- read: git/howto/recover-corrupted-*

Extra

VARIABLES (just a few)
- `GIT_DIR`
  - main git metadata dir (ie. `.git/`)
- `GIT_EXEC_PATH`
  - where git(1) looks for subcommands
- `GIT_INDEX_FILE`
  - instead of `$GIT_DIR/index`
- `GIT_WORK_TREE`
  - for checkout, etc. - can point outside repo
- `GIT_OBJECT_DIRECTORY`
  - external actions on objects (ie. extracting packs)
- `GIT_ALTERNATE_OBJECT_DIRECTORIES`
  - for easy injections of extra objects into a repo
- `GIT_AUTHOR_{NAME,EMAIL,DATE}`
- `GIT_COMMITTER_{NAME,EMAIL,DATE}`
  - override config defaults for git-commit(1)
- `GIT_TRACE` or `GIT_TRACE_*`
  - enable git debugging (see `man git` for more)
- `GIT_NAMESPACE`
  - subst all `refs/*` for `refs/$GIT_NAMESPACE/*`

TIPS
- create permanently detached worktree with git commands working like in a normal one:
  - create file `.git` in tld, with
    - `gitdir: /absolute/path/to/git_dir`
- create url base aliases in local config
  - [url "git://actual-url/"] \ insteadOf = work:
  - `git clone work:somerepo // git-fetch(1)`
- search commits for
  - string in log message: `git log --grep=str`
  - string in modified code: `git log -Sstr`

Maintenance

EASY FIX
- do nothing - git has auto gc

~~LAZY~~ SIMPLE
- just: `gc`
  - thorough: `gc --aggressive`

MANUAL
- add pack: `repack`
- into one: `repack -a -d` or `-A -d`
  - c.ratio: `--window` and `--depth`
- rm loose: `prune [-n]`

MINIMUM SIZE (reasonably)
- rm reflog: `reflog expire \ --expire=now --all`
- gc, prune: `gc --aggressive --prune=all`

Merge

STRATEGIES
- `merge-resolve` / `merge-recursive`
  - recursive has additional `-X` opts
- `merge-octopus`
- `merge-ours` / `merge-theirs`
- `merge-subtree`

FLOW
- strategy: `merge-*`
- ancestor: `merge-base <commit> [commit]...`
- trivial: `read-tree -m`
- show: `ls-files --unmerged`
- external: `merge-index <prog> {-a|paths}`
- prog: `merge-one-file`
- uses: `merge-file <ours> <base> <theirs>`

CONFLICTS
- stages: `ls-files --unmerged`
- tofiles: `checkout-index --stage=all -a`
  - one: `checkout-index --stage=N <file>`
- markers: `merge-file <ours> <base> <theirs>`
  - if no base, touch empty file and use it as `<base>`

Documentation

MANPAGES
- per-command:
  - `man git-log`
  - `git log --help`
- `apropos -s7 ^git`
  - gitrevisions
  - gitglossary
  - gitcore-tutorial
  - ...

BOOKS
- Version Control with Git, 2nd Edition
  - reasonably thorough, in-depth
- ProGIT
  - progit.org
- (old) Git Community Book
  - schacon.github.io/gitbook

DOCS ONLINE
- kernel.org/pub/software/scm/git/docs

Version 1.0