Git LFS Large files in Git

START HERE

`git lfs install`
Enable LFS for your user

`git lfs status`
LFS view of pending changes

`git lfs track "*.psd"`
Store matching files in LFS

`git lfs pull`
Download real file contents

`git add .gitattributes`
Commit the tracking rule

`git lfs migrate info`
Find your largest files

`git lfs ls-files`
List files stored in LFS

`git lfs env`
Show config and endpoints

Large file in working tree → **`git add (clean)`** → **Pointer file** tiny text in Git → **`git push (pre-push)`** → **LFS server** real contents → **`checkout (smudge)`** → **Working tree** real file again

CONCEPTS

Track a pattern before you add the files: git lfs track only affects files added afterwards, and the rule lives in .gitattributes, which you must commit.

Git itself stores only a small pointer; the contents are uploaded on push and downloaded on checkout.

A file that shows a few lines starting with 'version https://git-lfs' is an unfetched pointer: run git lfs pull.

SETUP

`git lfs install`
Set up LFS filters for your user
Run once per machine; inside a repo it also adds the pre-push hook.

`git lfs install --local`
Set up LFS in this repo only

`git lfs install --skip-smudge`
Clone with pointers, no download
Handy for CI that does not need the big files; fetch later with git lfs pull.

`git lfs version`
Show the installed version

`git lfs env`
Show endpoint, paths and config

`git lfs update`
Reinstall the repo hooks

TRACKING

`git lfs track "<pattern>"`
Track files matching a pattern
Quote the pattern so the shell does not expand it.

`git lfs track`
List tracked patterns

`git lfs untrack "<pattern>"`
Stop tracking a pattern
Existing commits keep their pointers; use migrate export to convert them back.

`git lfs track --filename <file>`
Track one exact filename

TRACKING (cont.)

`git lfs track --dry-run "<pattern>"`
Preview without changing anything

.gitattributes entry

```
*.psd filter=lfs diff=lfs
merge=lfs -text
```

INSPECT

`git lfs ls-files`
List LFS files in this checkout
*A * after the OID means content is present, a - means pointer only.*

`git lfs ls-files -s`
Include object sizes

`git lfs ls-files -a`
Files from all history

`git lfs ls-files <ref>`
List LFS files at a ref

`git lfs status`
LFS files changed or not yet pushed

`git lfs status --porcelain`
Script-friendly status

`git lfs migrate info`
Size summary by file type
Looks at unpushed commits on the current branch unless you add --everything.

TRANSFER

`git lfs pull`
Fetch objects and update files

`git lfs fetch`
Download objects, keep files as is

`git lfs fetch --all`
Download every referenced object

`git lfs fetch --recent`
Also fetch recent branches

`git lfs pull -I "<paths>"`
Only fetch matching paths
-X excludes paths; both take comma-separated globs.

`git lfs checkout`
Write fetched content into files
Downloads nothing; uses the local LFS store.

TRANSFER (cont.)

`git lfs push origin <branch>`
Upload objects for a branch
A normal git push already does this through the pre-push hook.

`git lfs push --all origin`
Upload all local history
Use after switching or mirroring to a new remote.

`git lfs push --dry-run origin`
Show what would be uploaded

MIGRATE

`git lfs migrate import --include="<patterns>"`
Convert history to LFS
Rewrites commits; force-push needed afterwards.

`git lfs migrate import --include="<patterns>" --everything`
Convert all refs

`git lfs migrate import --above=<size>`
Convert files over a size
Size such as 1mb or 500kb.

`git lfs migrate import --no-rewrite <files>`
Convert without rewriting history
Adds one new commit; also writes .gitattributes.

`git lfs migrate export --include="<patterns>"`
Convert LFS files back to Git

`git lfs migrate import --fixup`
Repair files wrongly committed

Migrate import/export rewrite history and change commit hashes. Commit or stash first, check the result, then force-push and tell collaborators to re-clone.

CLEAN UP

`git lfs prune`
Delete old local LFS objects
Keeps the current checkout, unpushed and recent objects.

`git lfs prune --dry-run`
Preview what would be deleted

CLEAN UP (cont.)

`git lfs prune --verify-remote`
Delete only if the server has it

`git lfs fsck`
Check LFS objects for corruption

`git lfs dedup`
Save disk space on copy-on-write FS

LOCKING

`git lfs lock <path>`
Lock a file against other users
Useful for unmergeable binaries; the server must support locking.

`git lfs locks`
List current locks

`git lfs unlock <path>`
Release your lock

`git lfs unlock --force <path>`
Break someone else's lock

`git config lfs.locksverify true`
Verify locks on push

RECIPES

Start tracking a file type

```
git lfs install
git lfs track "*.psd"
git add .gitattributes
design.psd
git commit -m "Track PSD files
in LFS"
git push
```

Move existing history to LFS

```
git lfs migrate info --
everything
git lfs migrate import --
include="*.zip" --everything
git lfs ls-files
git push --force-with-lease --
all
```

Clone without big files, fetch later

```
GIT_LFS_SKIP_SMUDGE=1 git clone
<url>
cd <repo>
git lfs pull -I "assets/**"
```

Diagnose pointer files in the tree

```
git lfs ls-files   # '-' marks
pointers
git lfs env
git lfs pull
```