NAME

git−lfs−config − Configuration options for git−lfs

CONFIGURATION FILES

git−lfs reads its configuration from any file supported by **git config −l**, including all per−repository, per−user, and per−system Git configuration files.

Additionally, a small number of settings can be specified in a file called **.lfsconfig** at the root of the repository; see the "LFSCONFIG" section for more details. This configuration file is useful for setting options such as the LFS URL or access type for all users of a repository, especially when these differ from the default. The **.lfsconfig** file uses the same format as **.gitconfig**.

If the **.lfsconfig** file is missing, the index is checked for a version of the file, and that is used instead. If both are missing, **HEAD** is checked for the file. If the repository is bare, only **HEAD** is checked. This order may change for checkouts in the future to better match Git's behavior.

Settings from Git configuration files override the **.lfsconfig** file. This allows you to override settings like **lfs.url** in your local environment without having to modify the **.lfsconfig** file.

Most options regarding git−lfs are contained in the **[lfs]** section, meaning they are all named **lfs.foo** or similar, although occasionally an lfs option can be scoped inside the configuration for a remote.

LIST OF OPTIONS

General settings

- **lfs.url** / **remote.<remote>.lfsurl**

  The url used to call the Git LFS remote API. Default blank (derive from clone URL).

- **lfs.pushurl** / **remote.<remote>.lfspushurl**

  The url used to call the Git LFS remote API when pushing. Default blank (derive from either LFS non−push urls or clone url).

- **remote.lfsdefault**

  The remote used to find the Git LFS remote API. **lfs.url** and **branch.*.remote** for the current branch override this setting. If this setting is not specified and there is exactly one remote, that remote is picked; otherwise, the default is **origin**.

- **remote.lfspushdefault**

  The remote used to find the Git LFS remote API when pushing. **lfs.url** and **branch.*.pushremote** for the current branch override this setting. If this setting is not set, **remote.pushdefault** is used, or if that is not set, the order of selection is used as specified in the **remote.lfsdefault** above.

- **lfs.dialtimeout**

  Sets the maximum time, in seconds, that the HTTP client will wait to initiate a connection. This does not include the time to send a request and wait for a response. Default: 30 seconds

- **lfs.tlstimeout**

  Sets the maximum time, in seconds, that the HTTP client will wait for a TLS handshake. Default: 30 seconds.

- **lfs.activitytimeout** / **lfs.https://<host>.activitytimeout**

  Sets the maximum time, in seconds, that the HTTP client will wait for the next tcp read or write. If < 1, no activity timeout is used at all. Default: 30 seconds

- **lfs.keepalive**

  Sets the maximum time, in seconds, for the HTTP client to maintain keepalive connections. Default: 30 minutes.

- **lfs.ssh.automultiplex**

When using the pure SSH−based protocol, whether to multiplex requests over a single connection when possible. This option requires the use of OpenSSH or a compatible SSH client. Default: true.

- **lfs.ssh.retries**

  Specifies the number of times Git LFS will attempt to obtain authorization via SSH before aborting. Default: 5.

- **core.askpass**, GIT_ASKPASS

  Given as a program and its arguments, this is invoked when authentication is needed against the LFS API. The contents of stdout are interpreted as the password.

- **lfs.cachecredentials**

  Enables in−memory SSH and Git Credential caching for a single 'git lfs' command. Default: enabled.

- **lfs.storage**

  Allow override LFS storage directory. Non−absolute path is relativized to inside of Git repository directory (usually **.git**).

  Note: you should not run **git lfs prune** if you have different repositories sharing the same storage directory.

  Default: **lfs** in Git repository directory (usually **.git/lfs**).

- **lfs.largefilewarning**

  Warn when a file is 4 GiB or larger. Such files will be corrupted when using Windows (unless smudging is disabled) due to a limitation in Git. Default: true.

Transfer (upload / download) settings

These settings control how the upload and download of LFS content occurs.

- **lfs.concurrenttransfers**

  The number of concurrent uploads/downloads. Default 8.

- **lfs.basictransfersonly**

  If set to true, only basic HTTP upload/download transfers will be used, ignoring any more advanced transfers that the client/server may support. This is primarily to work around bugs or incompatibilities.

  The git−lfs client supports basic HTTP downloads, resumable HTTP downloads (using **Range** headers), and resumable uploads via tus.io protocol. Custom transfer methods can be added via **lfs.customtransfer** (see next section). However setting this value to true limits the client to simple HTTP.

- **lfs.tustransfers**

  If set to true, this enables resumable uploads of LFS objects through the tus.io API. Once this feature is finalized, this setting will be removed, and tus.io uploads will be available for all clients.

- **lfs.standalonetransferagent**

  Allows the specified custom transfer agent to be used directly for transferring files, without asking the server how the transfers should be made. The custom transfer agent has to be defined in a **lfs.customtransfer.<name>** settings group.

- **lfs.customtransfer.<name>.path**

  lfs.customtransfer.<name> is a settings group which defines a custom transfer hook which allows you to upload/download via an intermediate process, using any mechanism you like (rather than just HTTP). **path** should point to the process you wish to invoke. The protocol between the git−lfs client and the custom transfer process is documented at https://github.com/git−lfs/git−lfs/blob/main/docs/custom−transfers.md

name must be a unique identifier that the LFS server understands. When calling the LFS API the client will include a list of supported transfer types. If the server also supports this named transfer type, it will select it and actions returned from the API will be in relation to that transfer type (may not be traditional URLs for example). Only if the server accepts *name* as a transfer it supports will this custom transfer process be invoked.

- **lfs.customtransfer.<name>.args**

  If the custom transfer process requires any arguments, these can be provided here. This string will be expanded by the shell.

- **lfs.customtransfer.<name>.concurrent**

  If true (the default), git−lfs will invoke the custom transfer process multiple times in parallel, according to **lfs.concurrenttransfers**, splitting the transfer workload between the processes.

- **lfs.customtransfer.<name>.direction**

  Specifies which direction the custom transfer process supports, either "download", "upload", or "both". The default if unspecified is "both".

- **lfs.transfer.maxretries**

  Specifies how many retries LFS will attempt per OID before marking the transfer as failed. Must be an integer which is at least one. If the value is not an integer, is less than one, or is not given, a value of eight will be used instead.

- **lfs.transfer.maxretrydelay**

  Specifies the maximum time in seconds LFS will wait between each retry attempt. LFS uses exponential backoff for retries, doubling the time between each retry until reaching this limit. If a server requests a delay using the **Retry−After** header, the header value overrides the exponential delay for that attempt and is not limited by this option.

  Must be an integer which is not negative. Use zero to disable delays between retries unless requested by a server. If the value is not an integer, is negative, or is not given, a value of ten will be used instead.

- **lfs.transfer.maxverifies**

  Specifies how many verification requests LFS will attempt per OID before marking the transfer as failed, if the object has a verification action associated with it. Must be an integer which is at least one. If the value is not an integer, is less than one, or is not given, a default value of three will be used instead.

- **lfs.transfer.enablehrefrewrite**

  If set to true, this enables rewriting href of LFS objects using **url.*.insteadof/pushinsteadof** config. **pushinsteadof** is used only for uploading, and **insteadof** is used for downloading and for uploading when **pushinsteadof** is not set.

Push settings

- **lfs.allowincompletepush**

  When pushing, allow objects to be missing from the local cache without halting a Git push. Default: false.

Fetch settings

- **lfs.fetchinclude**

  When fetching, only download objects which match any entry on this comma−separated list of paths/filenames. Wildcard matching is as per git−ignore(1). See git−lfs−fetch(1) for examples.

- **lfs.fetchexclude**

  When fetching, do not download objects which match any item on this comma−separated list of paths/filenames. Wildcard matching is as per git−ignore(1). See git−lfs−fetch(1) for examples.

- **lfs.fetchrecentrefsdays**

  If non−zero, fetches refs which have commits within N days of the current date. Only local refs are included unless lfs.fetchrecentremoterefs is true. Also used as a basis for pruning old files. The default is 7 days.

- **lfs.fetchrecentremoterefs**

  If true, fetches remote refs (for the remote you're fetching) as well as local refs in the recent window. This is useful to fetch objects for remote branches you might want to check out later. The default is true; if you set this to false, fetching for those branches will only occur when you either check them out (losing the advantage of fetch −−recent), or create a tracking local branch separately then fetch again.

- **lfs.fetchrecentcommitsdays**

  In addition to fetching at refs, also fetches previous changes made within N days of the latest commit on the ref. This is useful if you're often reviewing recent changes. Also used as a basis for pruning old files. The default is 0 (no previous changes).

- **lfs.fetchrecentalways**

  Always operate as if −−recent was included in a **git lfs fetch** call. Default false.

Prune settings

- **lfs.pruneoffsetdays**

  The number of days added to the **lfs.fetchrecent*** settings to determine what can be pruned. Default is 3 days, i.e. that anything fetched at the very oldest edge of the 'recent window' is eligible for pruning 3 days later.

- **lfs.pruneremotetocheck**

  Set the remote that LFS files must have been pushed to in order for them to be considered eligible for local pruning. Also the remote which is called if −−verify−remote is enabled.

- **lfs.pruneverifyremotealways**

  Always run **git lfs prune** as if **−−verify−remote** was provided.

Extensions

- **lfs.extension.<name>.<setting>**

  Git LFS extensions enable the manipulation of files streams during smudge and clean. **name** groups the settings for a single extension, and the settings are: * **clean** The command which runs when files are added to the index * **smudge** The command which runs when files are written to the working copy * **priority** The order of this extension compared to others

Other settings

- **lfs.<url>.access**

  Note: this setting is normally set by LFS itself on receiving a 401 response (authentication required), you don't normally need to set it manually.

  If set to "basic" then credentials will be requested before making batch requests to this url, otherwise a public request will initially be attempted.

- **lfs.<url>.locksverify**

  Determines whether locks are checked before Git pushes. This prevents you from pushing changes to files that other users have locked. The Git LFS pre−push hook varies its behavior based on the value of this config key.

- **null** − In the absence of a value, Git LFS will attempt the call, and warn if it returns an error. If the response is valid, Git LFS will set the value to **true**, and will halt the push if the user attempts to update a file locked by another user. If the server returns a **501 Not Implemented** response, Git LFS will set the value to **false.**
- **true** − Git LFS will attempt to verify locks, halting the Git push if there are any server issues, or if the user attempts to update a file locked by another user.
- **false** − Git LFS will completely skip the lock check in the pre−push hook. You should set this if you're not using File Locking, or your Git server verifies locked files on pushes automatically.

Supports URL config lookup as described in: https://git−scm.com/docs/git−config#git−config−httplturlgt. To set this value per−host: **git config −−global lfs.https://github.com/.locksverify [true|false]**.

- **lfs.<url>.contenttype**

  Determines whether Git LFS should attempt to detect an appropriate HTTP **Content−Type** header when uploading using the 'basic' upload adapter. If set to false, the default header of **Content−Type: application/octet−stream** is chosen instead. Default: 'true'.

- **lfs.skipdownloaderrors**

  Causes Git LFS not to abort the smudge filter when a download error is encountered, which allows actions such as checkout to work when you are unable to download the LFS content. LFS files which could not download will contain pointer content instead.

  Note that this will result in git commands which call the smudge filter to report success even in cases when LFS downloads fail, which may affect scripts.

  You can also set the environment variable GIT_LFS_SKIP_DOWNLOAD_ERRORS=1 to get the same effect.

- **GIT_LFS_PROGRESS**

  This environment variable causes Git LFS to emit progress updates to an absolute file−path on disk when cleaning, smudging, or fetching.

  Progress is reported periodically in the form of a new line being appended to the end of the file. Each new line will take the following format:

  <direction> <current>/<total files> <downloaded>/<total> <name>

  Each field is described below: * **direction**: The direction of transfer, either "checkout", "download", or "upload". * **current** The index of the currently transferring file. * **total files** The estimated count of all files to be transferred. * **downloaded** The number of bytes already downloaded. * **total** The entire size of the file, in bytes. * **name** The name of the file.

- **GIT_LFS_FORCE_PROGRESS lfs.forceprogress**

  Controls whether Git LFS will suppress progress status when the standard output stream is not attached to a terminal. The default is **false** which makes Git LFS detect whether stdout is a terminal and suppress progress when it's not; you can disable this behaviour and force progress status even when standard output stream is not a terminal by setting either variable to 1, 'yes' or 'true'.

- **GIT_LFS_SKIP_SMUDGE**

  Sets whether or not Git LFS will skip attempting to convert pointers of files tracked into their corresponding objects when checked out into a working copy. If 'true', '1', 'on', or similar, Git LFS will skip the smudge process in both **git lfs smudge** and **git lfs filter−process**. If unset, or set to 'false',

’0’, ’off’, or similar, Git LFS will smudge files as normal.

- **GIT_LFS_SKIP_PUSH**

  Sets whether or not Git LFS will attempt to upload new Git LFS object in a pre−push hook. If ’true’, ’1’, ’on’, or similar, Git LFS will skip the pre−push hook, so no new Git LFS objects will be uploaded. If unset, or set to ’false’, ’0’, ’off’, or similar, Git LFS will proceed as normal.

- **GIT_LFS_SET_LOCKABLE_READONLY lfs.setlockablereadonly**

  These settings, the first an environment variable and the second a gitconfig setting, control whether files marked as ’lockable’ in **git lfs track** are made read−only in the working copy when not locked by the current user. The default is **true**; you can disable this behaviour and have all files writeable by setting either variable to 0, ’no’ or ’false’.

- **lfs.lockignoredfiles**

  This setting controls whether Git LFS will set ignored files that match the lockable pattern read only as well as tracked files. The default is **false**; you can enable this behavior by setting the variable to 1, ’yes’, or ’true’.

- **lfs.defaulttokenttl**

  This setting sets a default token TTL when git−lfs−authenticate does not include the TTL in the JSON response but still enforces it.

  Note that this is only necessary for larger repositories hosted on LFS servers that don’t include the TTL.

LFSCONFIG

The .lfsconfig file in a repository is read and interpreted in the same format as the file stored in .git/config. It allows a subset of keys to be used, including and limited to:

- lfs.allowincompletepush
- lfs.fetchexclude
- lfs.fetchinclude
- lfs.gitprotocol
- lfs.locksverify
- lfs.pushurl
- lfs.skipdownloaderrors
- lfs.url
- lfs.{*}.access
- remote.{name}.lfsurl

The set of keys allowed in this file is restricted for security reasons.

EXAMPLES

Configure a custom LFS endpoint for your repository:

git config −f .lfsconfig lfs.url https://lfs.example.com/foo/bar/info/lfs

SEE ALSO

git−config(1), git−lfs−install(1), gitattributes(5)

Part of the git−lfs(1) suite.