

uploadpack

uploadpack.hideRefs

This variable is the same as `transfer.hideRefs`, but applies only to `upload-pack` (and so affects only fetches, not pushes). An attempt to fetch a hidden ref by `git fetch` will fail. See also `uploadpack.allowTipSHA1InWant`.

uploadpack.allowTipSHA1InWant

When `uploadpack.hideRefs` is in effect, allow `upload-pack` to accept a fetch request that asks for an object at the tip of a hidden ref (by default, such a request is rejected). See also `uploadpack.hideRefs`. Even if this is false, a client may be able to steal objects via the techniques described in the "SECURITY" section of the `linkgit:gitnamespaces[7]` man page; it's best to keep private data in a separate repository.

uploadpack.allowReachableSHA1InWant

Allow `upload-pack` to accept a fetch request that asks for an object that is reachable from any ref tip. However, note that calculating object reachability is computationally expensive. Defaults to `false`. Even if this is false, a client may be able to steal objects via the techniques described in the "SECURITY" section of the `linkgit:gitnamespaces[7]` man page; it's best to keep private data in a separate repository.

uploadpack.allowAnySHA1InWant

Allow `upload-pack` to accept a fetch request that asks for any object at all. It implies `uploadpack.allowTipSHA1InWant` and `uploadpack.allowReachableSHA1InWant`. If set to `true` it will enable both of them, if set to `false` it will disable both of them. By default not set.

uploadpack.keepAlive

When `upload-pack` has started `pack-objects`, there may be a quiet period while `pack-objects` prepares the pack. Normally it would output progress information, but if `--quiet` was used for the fetch, `pack-objects` will output nothing at all until the pack data begins. Some clients and networks may consider the server to be hung and give up. Setting this option instructs `upload-pack` to send an empty keepalive packet every `uploadpack.keepAlive` seconds. Setting this option to 0 disables keepalive packets entirely. The default is 5 seconds.

uploadpack.packObjectsHook

If this option is set, when `upload-pack` would run `git pack-objects` to create a packfile for a client, it will run this shell command instead. The `pack-objects` command and arguments it *would* have run (including the `git pack-objects` at the beginning) are appended to the shell command. The stdin and stdout of the hook are treated as if `pack-objects` itself was run. I.e., `upload-pack` will feed input intended for `pack-objects` to the hook, and expects a completed packfile on stdout.

Note that this configuration variable is only respected when it is specified in protected configuration (see [SCOPES](#)). This is a safety measure against fetching from untrusted repositories.

uploadpack.allowFilter

If this option is set, `upload-pack` will support partial clone and partial fetch object filtering.

uploadpackfilter.allow

Provides a default value for unspecified object filters (see: the below configuration variable). If set to `true`, this will also enable all filters which get added in the future. Defaults to `true`.

uploadpackfilter.<filter>.allow

Explicitly allow or ban the object filter corresponding to `<filter>`, where `<filter>` may be one of: `blob:none`, `blob:limit`, `object:type`, `tree`, `sparse:oid`, or `combine`. If using combined filters, both `combine` and all of the nested filter kinds must be allowed. Defaults to `uploadpackfilter.allow`.

uploadpackfilter.tree.maxDepth

Only allow `--filter=tree:<n>` when `<n>` is no more than the value of `uploadpackfilter.tree.maxDepth`. If set, this also implies `uploadpackfilter.tree.allow=true`, unless this configuration variable had already been set. Has no effect if unset.

uploadpack.allowRefInWant

If this option is set, `upload-pack` will support the `ref-in-want` feature of the protocol version 2 `fetch` command. This feature is intended for the benefit of load-balanced servers which may not have the same view of what OIDs their refs point to due to replication delay.

uploadpack.lazyFetchTrusted

A multi-valued configuration variable, each value of which specifies the absolute local path of a repository that `upload-pack` is allowed to lazily fetch missing objects for.

A repository is identified by its git directory, after following any `.git` file and resolving symbolic links. That is the repository itself if it is bare, the `.git` directory of a repository that has a worktree, or the directory that a `.git` file points to, for example when the repository was created with `--separate-git-dir` or for a linked worktree (see `linkgit:git-worktree[1]`). So a non-bare repository served as `/srv/repo` usually has to be allowlisted as `/srv/repo/.git`. Giving a path with `/*` appended to it will trust all repositories under the named directory. To trust all served repositories, set `uploadpack.lazyFetchTrusted` to the string `*`.

The value of this setting is interpolated, i.e., `~/<path>` expands to a path relative to the home directory and `%(<prefix>)/<path>` expands to a path relative to Git's (runtime) prefix.

By default, `upload-pack` refuses to lazily fetch (see the description of the `GIT_NO_LAZY_FETCH` environment variable in `linkgit:git-upload-pack[1]`), because doing so would run `git fetch`, which may execute arbitrary commands specified in the configuration and hooks of the served repository. Listing a repository here tells `upload-pack` that it is trusted, so lazy fetching from the promisor remotes configured in it is allowed. This is similar to setting `GIT_NO_LAZY_FETCH` to `0`, but only for the matching repositories: unlike that environment variable, the trust is not inherited by child

processes. So if a trusted repository lazily fetches from a promisor remote that is itself served by `upload-pack` on the same machine, for example through a local path or a `file://` URL, lazy fetching is allowed there only if that promisor remote is also listed here. An explicitly set `GIT_NO_LAZY_FETCH` takes precedence over this setting.

Note that this allows lazy fetching from any promisor remote configured in the served repository, not only from the promisor remotes that the client accepted using the "promisor-remote" protocol v2 capability (see `linkgit:gitprotocol-v2[5]`). The served repository is trusted as a whole, including its configuration, so the promisor remotes it configures are trusted too. It is the server operator's responsibility to make sure that the promisor remotes of a trusted repository are also trustworthy. In particular, a trusted repository should not be configured as its own promisor remote, as `upload-pack` would then try to lazily fetch missing objects from the repository itself, which is pointless.

As this is a multi-valued setting, you can add more than one repository via `git config (--global|--system) --add`. To reset the list of trusted repositories (e.g., to override any such repositories specified in the system config), add an `uploadpack.lazyFetchTrusted` entry with an empty value.

Note that this configuration variable is only respected when it is specified in protected configuration (see [SCOPES](#)). This prevents untrusted repositories from tampering with this value.

```
#2
[] [] [] 2026-10-31 20:09:09 UTC
[] [] [] 2026-10-31 20:09:11 UTC
```