Configuring Git Large File Storage for your enterprise

In this article

Git Large File Storage (Git LFS) is an open source extension to Git that allows you to work with large files the same way as other text files.

About Git Large File Storage

Git Large File Storage (Git LFS) is an open source extension to Git that allows you to work with large files the same way as other text files. You can use Git Large File Storage with a single repository, all of your personal or organization repositories, or with every repository in your enterprise. Before you can enable Git LFS for specific repositories or organizations, you need to enable Git LFS for your enterprise.

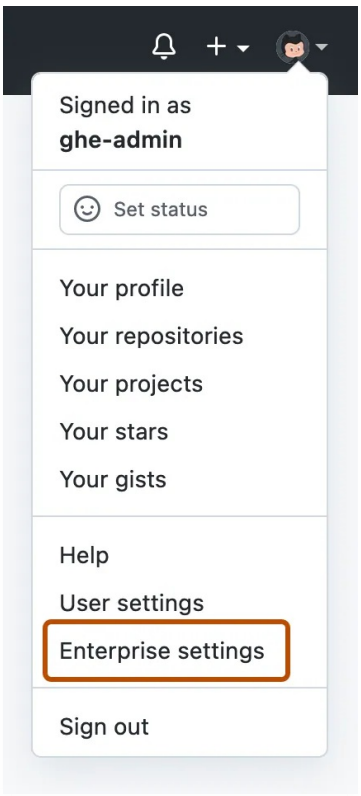
By default, the Git Large File Storage client stores large assets on the same server that hosts the Git repository. When Git LFS is enabled on your GitHub Enterprise Server instance, large assets are stored on the data partition in `/data/user/storage`.

Users cannot push Git LFS assets to GitHub Enterprise Server if Git LFS is disabled on the enterprise or repository.

For more information, see "About Git Large File Storage", "Managing large files," and the Git Large File Storage project site.

Configuring Git Large File Storage for your enterprise

1. In the top-right corner of GitHub Enterprise Server, click your profile photo, then click **Enterprise settings**.

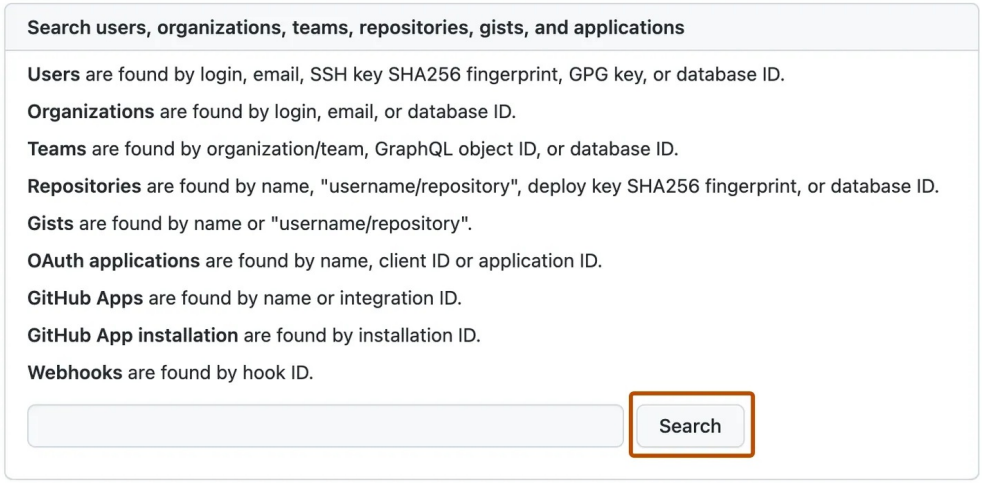


2. In the enterprise account sidebar, click ⚖ **Policies**.
3. Under ⚖ **Policies**, click **Options**.
4. Under "Git LFS access", select the drop-down menu, and click **Enabled** or **Disabled**.

Configuring Git Large File Storage for an individual repository

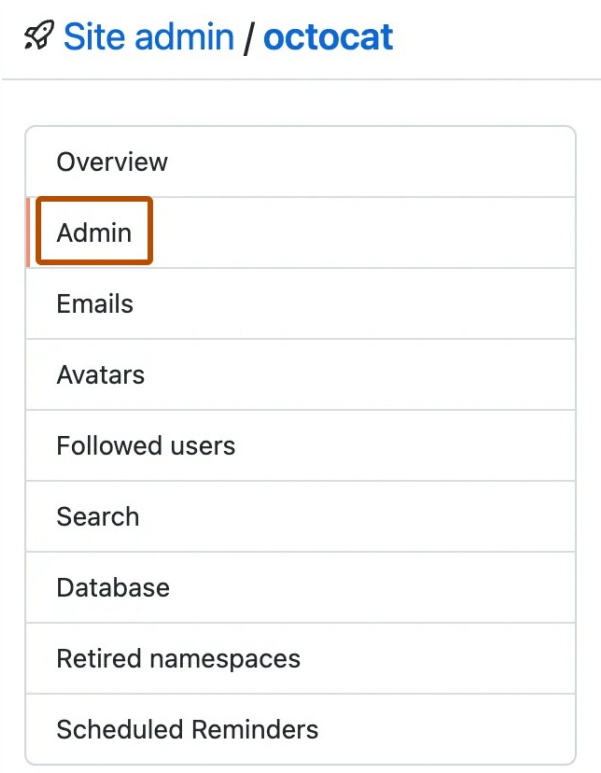
Note: Each repository automatically inherits default settings from the organization or user that owns it. You cannot override the default setting if the repository's owner has enforced the setting on all of their repositories.

1. From an administrative account on GitHub Enterprise Server, in the upper-right corner of any page, click 🚀.
2. If you're not already on the "Site admin" page, in the upper-left corner, click **Site admin**.
3. Under "Search users, organizations, teams, repositories, gists, and applications", type the name of the repository in the text field. Then to the right of the field, click **Search**.



4. Under "Search results – Repositories", click the name of the repository.
5. In the upper-right corner of the page, click **Admin**.

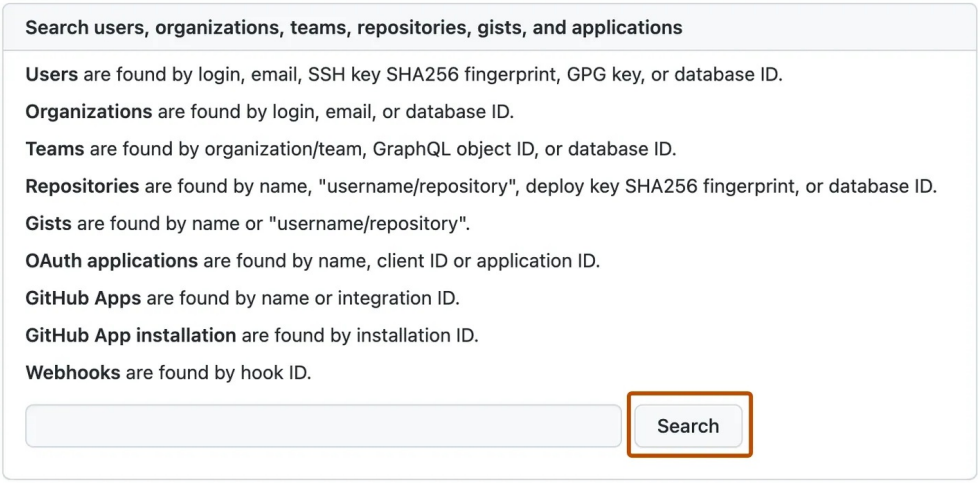
6. In the left sidebar, click **Admin**.



7. In the "Git LFS" section, next to "Toggle Git LFS access", click **Enable** or **Disable**.

Configuring Git Large File Storage for every repository owned by a user account or organization

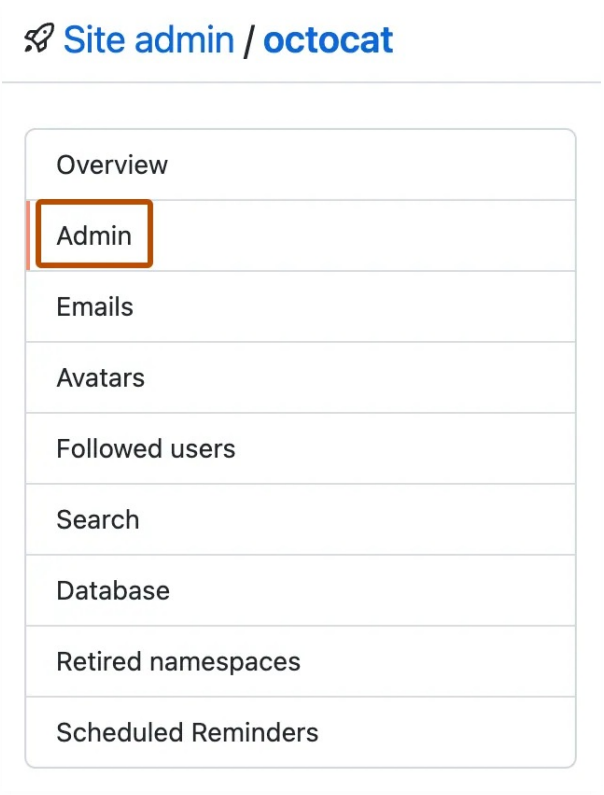
1. From an administrative account on GitHub Enterprise Server, in the upper-right corner of any page, click 🚀.
2. If you're not already on the "Site admin" page, in the upper-left corner, click **Site admin**.
3. Under "Search users, organizations, teams, repositories, gists, and applications", type the name of the user or organization in the text field. Then to the right of the field, click **Search**.



4. In the search results, click the name of the user or organization.

5. In the upper-right corner of the page, click ⚙ **Admin**.

6. In the left sidebar, click **Admin**.



7. In the "Git LFS" section, next to "Toggle Git LFS access", click **Enable** or **Disable**.

Configuring Git Large File Storage to use a third party server

By default, the Git Large File Storage client stores large assets on the same server that hosts the Git repository. When Git LFS is enabled on your GitHub Enterprise Server instance, large assets are stored on the data partition in `/data/user/storage`.

Users cannot push Git LFS assets to GitHub Enterprise Server if Git LFS is disabled on the enterprise or repository.

1. Disable Git LFS on your GitHub Enterprise Server instance. For more information, see "Configuring Git Large File Storage for your enterprise."
2. Create a Git LFS configuration file that points to the third party server.

```
# Show default configuration
$ git lfs env
> git-lfs/1.1.0 (GitHub; darwin amd64; go 1.5.1; git 94d356c)
> git version 2.7.4 (Apple Git-66)
&amp;nbsp;
> Endpoint=https://GITHUB-ENTERPRISE-HOST/path/to/repo/info/lfs (auth=basic)
&amp;nbsp;
# Create .lfsconfig that points to third party server.
$ git config -f .lfsconfig remote.origin.lfsurl https://THIRD-PARTY-LFS-
SERVER/path/to/repo
$ git lfs env
> git-lfs/1.1.0 (GitHub; darwin amd64; go 1.5.1; git 94d356c)
> git version 2.7.4 (Apple Git-66)
&amp;nbsp;
> Endpoint=https://THIRD-PARTY-LFS-SERVER/path/to/repo/info/lfs (auth=none)
&amp;nbsp;
# Show the contents of .lfsconfig
$ cat .lfsconfig
[remote "origin"]
lfsurl = https://THIRD-PARTY-LFS-SERVER/path/to/repo
```

3. To keep the same Git LFS configuration for each user, commit a custom `.lfsconfig` file to the repository.

```
git add .lfsconfig
git commit -m "Adding LFS config file"
```

4. Migrate any existing Git LFS assets. For more information, see "Migrating to a different Git Large File Storage server."

Migrating to a different Git Large File Storage server

Before migrating to a different Git Large File Storage server, you must configure Git LFS to use a third party server. For more information, see "Configuring Git Large File Storage to use a third party server."

1. Configure the repository with a second remote.

```
$ git remote add NEW-REMOTE https://NEW-REMOTE-HOSTNAME/path/to/repo
 
$ git lfs env
> git-lfs/1.1.0 (GitHub; darwin amd64; go 1.5.1; git 94d356c)
> git version 2.7.4 (Apple Git-66)
 
> Endpoint=https://GITHUB-ENTERPRISE-HOST/path/to/repo/info/lfs (auth=basic)
> Endpoint (NEW-REMOTE)=https://NEW-REMOTE-HOSTNAME/path/to/repo/info/lfs
(auth=none)
```

2. Fetch all objects from the old remote.

```
$ git lfs fetch origin --all
> Scanning for all objects ever referenced...
> ✔ 16 objects found
> Fetching objects...
> Git LFS: (16 of 16 files) 48.71 MB / 48.85 MB
```

3. Push all objects to the new remote.

```
$ git lfs push NEW-REMOTE --all
> Scanning for all objects ever referenced...
> ✔ 16 objects found
> Pushing objects...
> Git LFS: (16 of 16 files) 48.00 MB / 48.85 MB, 879.10 KB skipped
```

Further reading

- Git Large File Storage project site