Importing a Mercurial repository

In this article

You can import a repository from Mercurial by converting the repository to Git, then pushing the Git repository to GitHub Enterprise Server.

Prerequisites

To follow these steps, you must use a macOS or Linux system and have the following tools installed:

- Mercurial
- Git
- Git Large File Storage (Git LFS) (see "Installing Git Large File Storage")
- Python, including the `pip` package manager

Importing a Mercurial repository

1. Create a new repository on your GitHub Enterprise Server instance. To avoid errors, do not initialize the new repository with README, license, or gitignore files. You can add these files after your project has been pushed to GitHub Enterprise Server. For more information, see "Creating a new repository."
2. To confirm that Mercurial is installed on your machine, run `hg --version`.

   The output should be similar to `Mercurial Distributed SCM (version 6.4)`.
3. To confirm that Git is installed on your machine, run `git --version.`

   The output should be similar to `git version 2.40.0`.
4. To confirm that Git LFS is installed on your machine, run `git lfs --version`.

   The output should be similar to `git-lfs/3.1.4 (GitHub; darwin arm64; go 1.18.1)`.
5. To confirm that `pip` is installed on your machine, run `pip --version`.

   The output should be similar to `pip 21.2.4`.
6. To install the `mercurial` Python package, run `pip install mercurial`.
7. Download the latest release of fast-export to your machine, then extract the archive.
8. Move into the extracted directory, then run `./hg-fast-export.sh --help`.

The output should start with `usage: hg-fast-export.sh`.

9. Clone your Mercurial repository.

   For example, to clone the source code of Mercurial itself to the `mercurial-repo` directory, run `hg clone https://www.mercurial-scm.org/repo/hg mercurial-repo`.

10. Create a new directory, move into the new directory, then initialize a fresh Git repository.

    For example, if you want to name your new repository `mercurial-git`, run `mkdir mercurial-git && cd mercurial-git && git init`.

11. Move into the directory for the newly-created Git repository.

12. To configure your new Git repository to handle the case of filenames in the same way as Mercurial, run `git config core.ignoreCase false`.

13. To get a list of committers in your Mercurial project and store the list in `committers.txt`, run the following script.

    Shell

    ```
    hg log --template "{author}\n" | sort | uniq > committers.txt
    ```

14. Update your `committers.txt` file, mapping the committer name used in the Mercurial repository to the name you want to use in your Git repository, with the following format:

    ```
    “The Octocat <octocato@gmail.com>”=”Octocat <octocat@github.com>”
    ```

15. In your initialized Git repository, run `hg-fast-export.sh`, passing in the path to your Mercurial repository and the path to your `committers.txt` file as arguments.

    For example, `../fast-export-221024/hg-fast-export.sh -r ../mercurial-repo -A ../mercurial-repo/committers.txt -M main`.

16. After the import finishes, to check out your newly-created Git repository, run `git checkout HEAD`.

17. To add your GitHub repository as a remote, run `git remote add origin URL`, replacing `URL` with the URL for the GitHub repository you created earlier, such as `https://github.com/octocat/example-repository.git`.

18. To push the repository to GitHub, run `git push --mirror origin`.

    If your repository contains any files that are larger than GitHub Enterprise Server's file size limit, your push may fail. Move the large files to Git LFS by running `git lfs import`, then try again.