Exporting migration data from GitHub.com

In this article

You can export migration data from an organization on GitHub.com by using the API to select repositories to migrate, then generating a migration archive that you can import into a GitHub Enterprise Server instance.

Preparing the source organization on GitHub

1. Ensure that you have owner permissions on the source organization's repositories.
2. Generate an access token with the `repo` and `admin:org` scopes on GitHub.com.
3. To minimize downtime, make a list of repositories you want to export from the source instance. You can add multiple repositories to an export at once using a text file that lists the URL of each repository on a separate line.

Exporting the organization's repositories

> **Note:** Fork relationships do not persist after a migration.

To export repository data from GitHub.com, use the Migrations API.

The Migrations API is currently in a preview period, which means that the endpoints and parameters may change in the future.

Generating a migration archive

> **Note:** Locking a repository prevents all write access to the repository. You cannot associate new teams or collaborators with a locked repository.
>
> If you're performing a trial run, you do not need to lock the repository. When you migrate data from a repository that's in use, GitHub strongly recommends locking the repository. For more information, see "About ghe-migrator."

1. Notify members of your organization that you'll be performing a migration. The export can take several minutes, depending on the number of repositories being exported. The full migration including import may take several hours so we

recommend doing a trial run in order to determine how long the full process will take. For more information, see "About ghe-migrator."

2. Start a migration by sending a `POST` request to the migration endpoint. You'll need:

   - Your access token for authentication.
   - A list of the repositories you want to migrate:

     ```
     curl -H "Authorization: Bearer GITHUB_ACCESS_TOKEN" \
     -X POST \
     -H "Accept: application/vnd.github+json" \
     -d'{"lock_repositories":true,"repositories":["ORG_NAME</em>/REPO_NAME",
     "ORG_NAME/REPO_NAME"]}' \
     https://api.github.com/orgs/ORG_NAME/migrations
     ```

   - If you want to lock the repositories before migrating them, make sure `lock_repositories` is set to `true`. This is highly recommended.
   - You can exclude file attachments by passing `exclude_attachments: true` to the endpoint. File attachments can be large and may needlessly bloat your final migration archive. The final archive size must be less than 20 GB.

   This request returns a unique `id` which represents your migration. You'll need it for subsequent calls to the Migrations API.

3. Send a `GET` request to the migration status endpoint to fetch the status of a migration. You'll need:

   - Your access token for authentication.
   - The unique `id` of the migration:

     ```
     curl -H "Authorization: Bearer GITHUB_ACCESS_TOKEN" \
     -H "Accept: application/vnd.github+json" \
     https://api.github.com/orgs/ORG_NAME/migrations/ID
     ```

   A migration can be in one of the following states:

   - `pending`, which means the migration hasn't started yet.
   - `exporting`, which means the migration is in progress.
   - `exported`, which means the migration finished successfully.
   - `failed`, which means the migration failed.

4. After your migration has exported, download the migration archive by sending a `GET` request to the migration download endpoint. You'll need:

   - Your access token for authentication.
   - The unique `id` of the migration:

     ```
     curl -H "Authorization: Bearer GITHUB_ACCESS_TOKEN" \
     -H "Accept: application/vnd.github+json" \
     -L -o migration_archive.tar.gz \
     https://api.github.com/orgs/ORG_NAME/migrations/ID/archive
     ```

5. The migration archive is automatically deleted after seven days. If you would prefer to delete it sooner, you can send a `DELETE` request to the migration archive delete endpoint. You'll need:

   - Your access token for authentication.
   - The unique `id` of the migration:

     ```
     curl -H "Authorization: Bearer GITHUB_ACCESS_TOKEN" \
     ```

```
-X DELETE \
-H "Accept: application/vnd.github+json" \
https://api.github.com/orgs/ORG_NAME/migrations/ID/archive
```

6. To prepare the archived migration data for import into a GitHub Enterprise Server instance, see "Preparing to migrate data to GitHub Enterprise Server".