

Migrating data to GitHub Enterprise Server

In this article

- Applying the imported data on GitHub Enterprise Server
- Reviewing migration data
- Record type filters
- Record state filters
- Filtering audited records
- Completing the import on GitHub Enterprise Server
- Unlocking repositories on the target instance
- Unlocking repositories on the source

After generating a migration archive, you can import the data to your target GitHub Enterprise Server instance. You'll be able to review changes for potential conflicts before permanently applying the changes to your target instance.

Applying the imported data on GitHub Enterprise Server [↗](#)

Before you can migrate data to your GitHub Enterprise Server, you must prepare the data and resolve any conflicts. For more information, see "[Preparing to migrate data to GitHub Enterprise Server](#)."

After you prepare the data and resolve conflicts, you can apply the imported data on GitHub Enterprise Cloud.

- 1 As a site admin, [SSH into your target GitHub Enterprise Server instance](#).

```
ssh -p 122 admin@HOSTNAME
```

- 1 Using the `ghe-migrator import` command, start the import process. You'll need:
 - Your Migration GUID. For more information, see "[Preparing to migrate data to GitHub Enterprise Server](#)."
 - Your personal access token for authentication. The personal access token that you use is only for authentication as a site administrator, and does not require any specific scope or permissions. For more information, see "[Managing your personal access tokens](#)."

```
$ ghe-migrator import /home/admin/MIGRATION-GUID.tar.gz -g MIGRATION-GUID -u USERNAME -p TOKEN
```

```
> Starting GitHub::Migrator
```

```
> Import 100% complete /
```

- To specify where migration files should be staged append the command with `--staging-path=/full/staging/path` . Defaults to `/data/user/tmp` .

Reviewing migration data [↗](#)

By default, `ghe-migrator audit` returns every record. It also allows you to filter records by:

- The types of records.
- The state of the records.

The record types match those found in the [migrated data](#).

Record type filters [↗](#)

Record type	Filter name
Users	<code>user</code>
Organizations	<code>organization</code>
Repositories	<code>repository</code>
Teams	<code>team</code>
Milestones	<code>milestone</code>
Project boards	<code>project</code>
Issues	<code>issue</code>
Issue comments	<code>issue_comment</code>
Pull requests	<code>pull_request</code>
Pull request reviews	<code>pull_request_review</code>
Commit comments	<code>commit_comment</code>
Pull request review comments	<code>pull_request_review_comment</code>
Releases	<code>release</code>
Actions taken on pull requests or issues	<code>issue_event</code>
Protected branches	<code>protected_branch</code>

Record state filters [↗](#)

Record state	Description
<code>export</code>	The record will be exported.
<code>import</code>	The record will be imported.

imported	The record will be imported.
map	The record will be mapped.
rename	The record will be renamed.
merge	The record will be merged.
exported	The record was successfully exported.
imported	The record was successfully imported.
mapped	The record was successfully mapped.
renamed	The record was successfully renamed.
merged	The record was successfully merged.
failed_export	The record failed to export.
failed_import	The record failed to be imported.
failed_map	The record failed to be mapped.
failed_rename	The record failed to be renamed.
failed_merge	The record failed to be merged.

Filtering audited records [↗](#)

With the `ghe-migrator audit` command, you can filter based on the record type using the `-m` flag. Similarly, you can filter on the import state using the `-s` flag. The command looks like this:

```
ghe-migrator audit -m RECORD_TYPE -s STATE -g MIGRATION-GUID
```

For example, to view every successfully imported organization and team, you would enter:

```
$ ghe-migrator audit -m organization,team -s mapped,renamed -g MIGRATION-GUID
> model_name,source_url,target_url,state
> organization,https://gh.source/octo-org/,https://ghe.target/octo-org/,renamed
```

We strongly recommend auditing every import that failed. To do that, you will enter:

```
$ ghe-migrator audit -s failed_import,failed_map,failed_rename,failed_merge -g MIGRATION-GUID
> model_name,source_url,target_url,state
> user,https://gh.source/octocat,https://gh.target/octocat,failed
> repository,https://gh.source/octo-org/octo-project,https://ghe.target/octo-org/octo-project,failed
```

If you have any concerns about failed imports, you can contact us by visiting [GitHub Enterprise Support](#).

Completing the import on GitHub Enterprise Server



After your migration is applied to your target instance and you have reviewed the migration, you'll unlock the repositories and delete them off the source. Before deleting your source data we recommend waiting around two weeks to ensure that everything is functioning as expected.

Unlocking repositories on the target instance

- 1 SSH into GitHub.com. If your instance comprises multiple nodes, for example if high availability or geo-replication are configured, SSH into the primary node. If you use a cluster, you can SSH into any node. For more information about SSH access, see "[Accessing the administrative shell \(SSH\)](#)."

```
ssh -p 122 admin@HOSTNAME
```

- 2 Unlock all the imported repositories with the `ghe-migrator unlock` command. You'll need your Migration GUID:

```
$ ghe-migrator unlock -g MIGRATION-GUID  
> Unlocked octo-org/octo-project
```

Warning: If your repository contains GitHub Actions workflows using the `schedule` trigger, the workflows will not run automatically after an import. To start the scheduled workflows once again, push a commit to the repository. For more information, see "[Events that trigger workflows](#)."

Unlocking repositories on the source

Unlocking repositories from an organization on GitHub.com

To unlock the repositories on a GitHub.com organization, you'll send a `DELETE` request to [the migration unlock endpoint](#). You'll need:

- Your access token for authentication
- The unique `id` of the migration
- The name of the repository to unlock

```
curl -H "Authorization: Bearer GITHUB_ACCESS_TOKEN" -X DELETE \  
-H "Accept: application/vnd.github.wyandotte-preview+json" \  
https://api.github.com/orgs/ORG-NAME/migrations/ID/repos/REPO_NAME/lock
```

Deleting repositories from an organization on GitHub.com

After unlocking the GitHub.com organization's repositories, you should delete every repository you previously migrated using [the repository delete endpoint](#). You'll need your access token for authentication:

```
curl -H "Authorization: Bearer GITHUB_ACCESS_TOKEN" -X DELETE \  
https://api.github.com/repos/ORG-NAME/REPO_NAME
```

Unlocking repositories from a GitHub Enterprise Server instance

- 1 SSH into GitHub.com. If your instance comprises multiple nodes, for example if high availability or geo-replication are configured, SSH into the primary node. If you use a cluster, you can SSH into any node. For more information about SSH access, see "[Accessing the administrative shell \(SSH\)](#)."

```
ssh -p 122 admin@HOSTNAME
```

- 2 Unlock all the imported repositories with the `ghe-migrator unlock` command. You'll need your Migration GUID:

```
$ ghe-migrator unlock -g MIGRATION-GUID  
> Unlocked octo-org/octo-project
```

Legal

© 2023 GitHub, Inc. [Terms](#) [Privacy](#) [Status](#) [Pricing](#) [Expert services](#) [Blog](#)