Preparing to migrate data to GitHub Enterprise Server

In this article

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.

Preparing the migrated data for import to GitHub Enterprise Server

1. Using the `scp` command, copy the migration archive generated from your source instance or organization to your GitHub Enterprise Server target:

   ```
   scp -P 122 PATH-TO-MIGRATION-GUID.tar.gz admin@HOSTNAME:/home/admin/
   ```

2. As a site admin, SSH into your target GitHub Enterprise Server instance.

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

1. Use the `ghe-migrator prepare` command to prepare the archive for import on the target instance and generate a new Migration GUID for you to use in subsequent steps:

   ```
   ghe-migrator prepare /home/admin/MIGRATION-GUID.tar.gz
   ```

   - To start a new import attempt, run `ghe-migrator prepare` again and get a new Migration GUID.
   - To specify where migration files should be staged append the command with `--staging-path=/full/staging/path`. Defaults to `/data/user/tmp`.

Generating a list of migration conflicts

1. Using the `ghe-migrator conflicts` command with the Migration GUID, generate a *conflicts.csv* file:

```
ghe-migrator conflicts -g MIGRATION-GUID > conflicts.csv
```

- If no conflicts are reported, you can safely import the data by following the steps in "Migrating data to GitHub Enterprise Server".

2. If there are conflicts, using the `scp` command, copy *conflicts.csv* to your local computer:

```
scp -P 122 admin@HOSTNAME:conflicts.csv ~/Desktop
```

3. Continue to "Resolving migration conflicts or setting up custom mappings".

Reviewing migration conflicts

1. Using a text editor or CSV-compatible spreadsheet software, open *conflicts.csv*.
2. With guidance from the examples and reference tables below, review the *conflicts.csv* file to ensure that the proper actions will be taken upon import.

The *conflicts.csv* file contains a *migration map* of conflicts and recommended actions. A migration map lists out both what data is being migrated from the source, and how the data will be applied to the target.

`model_name`	`source_url`	`target_url`	`recommended_action`
`user`	`https://example-gh.source/octocat`	`https://example-gh.target/octocat`	`map`
`organization`	`https://example-gh.source/octo-org`	`https://example-gh.target/octo-org`	`map`
`repository`	`https://example-gh.source/octo-org/widgets`	`https://example-gh.target/octo-org/widgets`	`rename`
`team`	`https://example-gh.source/orgs/octo-org/teams/admins`	`https://example-gh.target/orgs/octo-org/teams/admins`	`merge`

Each row in *conflicts.csv* provides the following information:

Name	Description
`model_name`	The type of data being changed.
`source_url`	The source URL of the data.
`target_url`	The expected target URL of the data.
`recommended_action`	The preferred action `ghe-migrator` will take when importing the data.

Possible mappings for each record type

There are several different mapping actions that `ghe-migrator` can take when

transferring data:

`action`	Description	Applicable models
`import`	(default) Data from the source is imported to the target.	All record types
`map`	Instead of creating a new model based on the source data, an existing record in the target is used. Useful for importing a repository into an existing organization or mapping user identities in the target to user identities in the source.	Users, organizations
`rename`	Data from the source is renamed, then copied over to the target.	Users, organizations, repositories
`map_or_rename`	If the target exists, map to that target. Otherwise, rename the imported model.	Users
`merge`	Data from the source is combined with existing data on the target.	Teams

We strongly suggest you review the *conflicts.csv* file and use `ghe-migrator audit` to ensure that the proper actions are being taken. If everything looks good, you can continue to "Migrating data to GitHub Enterprise Server".

Resolving migration conflicts or setting up custom mappings

If you believe that `ghe-migrator` will perform an incorrect change, you can make corrections by changing the data in *conflicts.csv*. You can make changes to any of the rows in *conflicts.csv*.

For example, let's say you notice that the `octocat` user from the source is being mapped to `octocat` on the target.

`model_name`	`source_url`	`target_url`	`recommended_action`
`user`	`https://example-gh.source/octocat`	`https://example-gh.target/octocat`	`map`

You can choose to map the user to a different user on the target. Suppose you know that `octocat` should actually be `monalisa` on the target. You can change the `target_url` column in *conflicts.csv* to refer to `monalisa`.

`model_name`	`source_url`	`target_url`	`recommended_action`
`user`	`https://example-gh.source/octocat`	`https://example-gh.target/monalisa`	`map`

As another example, if you want to rename the `octo-org/widgets` repository to `octo-org/amazing-widgets` on the target instance, change the `target_url` to `octo-`

`org/amazing-widgets` and the `recommend_action` to `rename`.

model_name	source_url	target_url	recommended_action
repository	https://example-gh.source/octo-org/widgets	https://example-gh.target/octo-org/amazing-widgets	rename

Adding custom mappings

A common scenario during a migration is for migrated users to have different usernames on the target than they have on the source.

Given a list of usernames from the source and a list of usernames on the target, you can build a CSV file with custom mappings and then apply it to ensure each user's username and content is correctly attributed to them at the end of a migration.

You can quickly generate a CSV of users being migrated in the CSV format needed to apply custom mappings by using the `ghe-migrator audit` command:

```
ghe-migrator audit -m user -g MIGRATION-GUID > users.csv
```

Now, you can edit that CSV and enter the new URL for each user you would like to map or rename, and then update the fourth column to have `map` or `rename` as appropriate.

For example, to rename the user `octocat` to `monalisa` on the target `https://example-gh.target` you would create a row with the following content:

model_name	source_url	target_url	state
user	https://example-gh.source/octocat	https://example-gh.target/monalisa	rename

The same process can be used to create mappings for each record that supports custom mappings. For more information, see our table on the possible mappings for records.

Applying modified migration data

1. After making changes, use the `scp` command to apply your modified *conflicts.csv* (or any other mapping *.csv* file in the correct format) to the target instance:

   ```
   scp -P 122 ~/Desktop/conflicts.csv admin@HOSTNAME:/home/admin/
   ```

2. Re-map the migration data using the `ghe-migrator map` command, passing in the path to your modified *.csv* file and the Migration GUID:

   ```
   ghe-migrator map -i conflicts.csv  -g MIGRATION-GUID
   ```

3. If the `ghe-migrator map -i conflicts.csv -g MIGRATION-GUID` command reports that conflicts still exist, run through the migration conflict resolution process again.